

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

طیبه طاهری^۱، بابک نصیری^۲، محمدرضا میبیدی^۳

^۱ دانشکده برق، رایانه و فناوری اطلاعات، دانشگاه آزاد اسلامی، قزوین
t.taheri@qiau.ac.ir

^۲ دانشکده برق، رایانه و فناوری اطلاعات، دانشگاه آزاد اسلامی، قزوین
nasiri.babak@qiau.ac.ir

^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 بهینه‌سازی اشاره نمود. الگوریتم جستجوی هارمونی تا کنون بر روی 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 محدودیت‌های آن‌در طول زمان تغییر می‌کند که در نتیجه، راه حل بهینه بطور پیوسته در حال تغییر می‌باشد. از این‌رو لازم است که الگوریتم‌های بهینه‌سازی نیز متناسب با آن در حال 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

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 راه حل بهینه. در این مقاله از پویا بودن راه حل 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

الگوریتم جستجوی هارمونی^۱ (HS) یکی از الگوریتم‌های الهام گرفته شده از طبیعت است. این الگوریتم اولین بار توسط ژونگ‌هو جیم 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

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 پیاده‌سازی 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 شکل ۱ آورده شده است.

begin

Define objective function $f(x)$, $x=(x_1, x_2, \dots, x_d)^T$

Define harmony memory accepting rate (r_{accept})

Define pitch adjusting rate (r_{pa}) and other parameters

Generate Harmony Memory with random harmonies

while($t < \text{max number of iterations}$)

while($i \leq \text{number of variables}$)

if($\text{rand} < r_{accept}$),

Choose a value from HM for the variable i

if($\text{rand} < r_{pa}$),

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 [۲]

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. تعدادی از این راهکارها، تصادفی سازی در تمام یا بخشی از گروه را دربرمی‌گیرد [17]. تعدادی دیگر، از راه حل دفع کردن برای ایجاد تنوع استفاده می‌کند [18-20].

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. در صورتی که زمان انجام کار مهم نباشد، این یک گزینه مناسب است اما در بیشتر مواقع، زمان ارزش زیادی داشته و این راه حل 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 ($ps^{[8]}$) [۹]، الگوریتم ماهی‌های مصنوعی ($AFSA^{[10]}$) [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 قله‌های متحرک ($MPB^{[12]}$) که از معروفترین بنچمارکهای محیط پویاست ارزیابی شده و کارایی آن با شش الگوریتم معتبر دیگر بنام‌های، $mQSO^{[8]}$ ، $cellularps^{[13]}$ ، $fms^{[14]}$ ، $rps^{[15]}$ 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: بخش ۲ الگوریتم جستجوی هارمونی را به طور مختصر شرح می‌دهد، بخش ۳ توضیح مختصری در مورد کارهای انجام گرفته در زمینه محیط‌های پویا ارائه کرده، بخش ۴ الگوریتم پیشنهادی را توضیح می‌دهد. در بخش ۵ نتایج

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 قله-های 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.

```
// initialize swarms
for each swarm i
  for each harmony j
    initialize xij
    pbestij = xij
  end
  Gbest = arg min f(pbestij)
end
// initialize swarm finder
for each harmony finder
  initialize xij
end
repeat:
  for each swarm i
    for each harmony j
      update xij 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

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 صورتیکه مقدار R_{cloud} کوچک باشد توانایی جستجوی محلی الگوریتم بسیار بالا می‌رود، زیرا

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 Mpsو 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^۶ استفاده شده است. بدین صورت که کل هارمونی‌های موجود را به دسته‌های از پیش تعیین شده‌ای تقسیم کرده است. ارتباط بین دسته‌ها به دو صورت محلی براساس T_{exle} و سراسری بر اساس T_{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 مقداردهی اولیه می‌شود. همچنین اگر دو دسته در محدوده T_{exle} باشند دسته با کمترین مقدار شایستگی برای مقداردهی 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 احتمالاً بلندتر هستند نیز بوجود آید. در الگوریتم پیشنهادی نیز از این دو تکنیک به ترتیب به نامهای ضدگرایی^۷ و محروم‌سازی^۸ 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.

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

که در رابطه (۱) X_i بردار هارمونی نام 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

قله‌هاست. در جدول (۲)، بهترین نتیجه هر سطر 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. با توجه به جدول مشاهده می‌شود، زمانی که تعداد قله‌ها کمتر از تعداد دسته‌هاست چون ممکن است چند دسته بر روی 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.

جدول ۱: پارامترهای MPB

پارامتر	مقدار
M: تعداد قله‌ها	بین ۱ تا ۲۰۰
فرکانس تغییر	۵۰۰ و ۱۰۰۰ و ۵۰۰۰
میزان تغییر ارتفاع	۷
میزان تغییر عرض	۱
شکل قله	cone
تابع اولیه	ندارد
S: طول جابجایی	۱
N: تعداد ابعاد	۵
محدوده مکانی قله‌ها	[۰ و ۱۰۰ و ۰]
محدوده پارامتر ارتفاع	[۳ و ۷۰]
محدوده پارامتر عرض	[۱ و ۱۲]
مقدار ارتفاع اولیه قله‌ها	۵۰

همانطور که جدول ۲ نشان می‌دهد در هر سه فرکانس زمانی که تعداد قله‌ها ۲۰ و ۱۰ بوده الگوریتم پیشنهادی اختلاف چشمگیری با سایرین داشته است. همچنین در فرکانس‌های ۵۰۰ و ۵۰۰۰ با تعداد قله‌های ۵۰ نتیجه الگوریتم پیشنهادی بهتر از سایر الگوریتم‌ها بوده. در فرکانس ۱۰۰۰ با تعداد دسته‌های ۱۰، ۲۰، ۳۰ و ۴۰ مکانیزم استفاده شده بهتر از دیگر الگوریتم‌ها عمل می‌کند. اما همانطور که ملاحظه می‌شود در هر سه فرکانس زمانی که تعداد قله‌ها کمتر از تعداد دسته‌ها می‌شود این الگوریتم نمی‌تواند خوب عمل کند، مخصوصاً زمانی که تعداد قله‌ها ۱ باشد چون همه دسته‌ها روی یک قله با هم رقابت می‌کنند و هر دسته سعی می‌کند که دیگر دسته‌ها را از قله دور کند بنابراین زمان زیادی صرف دور کردن همدیگر می‌کنند و نتیجه بدست آمده مطلوب نخواهد بود. این روند زمانی که تعداد قله‌ها خیلی بیشتر از تعداد دسته‌هاست نیز ممکن است اتفاق بیفتد و خیلی از قله‌ها بدون پوشش باقی می‌ماند. با توجه به جدول نتایج و همچنین شکل ۳ می‌توان نتیجه گرفت که اگر بتوان الگوریتم را طوری طراحی کرد که تعداد دسته‌ها با در نظر گرفتن تعداد قله‌ها هماهنگ شوند، مطمئناً نتایج حاصله بهبود چشمگیری خواهد داشت.

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 می‌شود 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 دوبار 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.

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. معیار ارزیابی خطای برون خطی است که میانگین حرکت بین بهترین راه‌حل شناخته شده توسط الگوریتم و برازندگی بهینه سراسری است که فرمول آن در رابطه ۲ آورده شده است.

$$Offline_error = \frac{1}{FES} \sum_{t=1}^{FES} (fitness(gbest(t)) - fitness(global - optimum(t))) \quad (2)$$

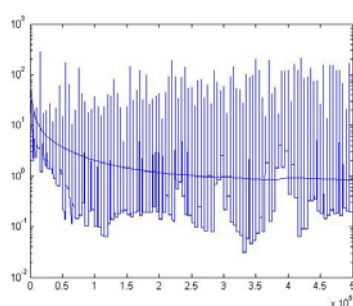
که در آن، FES بیشترین مقدار برازندگی، $gbest(t)$ و $global - optimum$ به ترتیب بهترین موقعیت شناخته شده توسط الگوریتم و بهینه سراسری روی t امین مقدار ارزیابی است.

الگوریتم پیشنهادی با نرم‌افزار متلب نوشته شده و برای بررسی کارایی آن، نتایج بدست آمده از این الگوریتم با نتایج چهار الگوریتم شناخته شده دیگر به نامهای $mQSO$ [۸]، $cellular\ pso$ [۱۳]، $fmso$ [۱۴]، $rpsso$ [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 در نظر گرفته شده است.

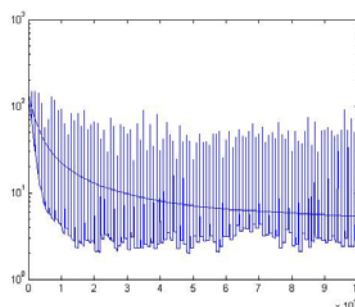
نتایج بدست آمده از خطای برون خطی و خطای استاندارد بر روی قله‌های مختلف در جدول (۲) نشان داده شده است. خطای استاندارد درون پراکنش نوشته شده است. در این جدول p نشان دهنده تعداد

جدول (۲): خطای برون خطی و خطای استاندارد الگوریتم‌ها در فرکانس‌های تغییرات ۵۰۰ و ۱۰۰۰ و ۵۰۰۰ با تعداد قله‌های مختلف

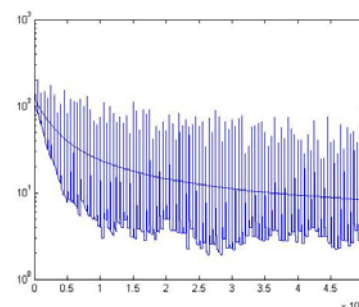
الگوریتم پیشنهادی	FMSO	mQSO	rPSO	Cellular PSO	P	Frequency
۵۰۰	۱۸.۶۹(۱.۰۴)	۳۳.۶۷(۳.۴)	۴.۲۷(۰.۱۷)	۱۳.۴(۰.۷۴)	۱	۱
	۹.۸۳(۰.۶۲)	۱۱.۹۱(۰.۷۶)	۱۶.۱۹(۰.۴۲)	۹.۶۳(۰.۴۹)	۵	۵
	۶.۹۰(۰.۱۷)	۹.۶۲(۰.۳۴)	۱۷.۳۴(۰.۲۷)	۹.۴۲(۰.۲۱)	۱۰	۱۰
	۷.۲۱(۰.۲۱)	۸.۷۹(۰.۲۳)	۱۷.۰۶(۰.۳۷)	۸.۸۴(۰.۲۸)	۲۰	۲۰
	۷.۹۰(۰.۲۱)	۸.۸۰(۰.۲۱)	۱۶.۹۸(۰.۲۷)	۷.۸۸(۰.۲۳)	۳۰	۳۰
	۸.۲۳(۰.۶۵)	۸.۵۵(۰.۲۱)	۱۶.۴۶(۰.۲۸)	۷.۸۳(۰.۲۱)	۴۰	۴۰
	۸.۵۵(۰.۲۴)	۸.۵۶(۰.۲)	۱۵.۷۷(۰.۱۸)	۸.۶۲(۰.۲۳)	۵۰	۵۰
	۱۰.۰۲(۰.۳۲)	۸.۵۴(۰.۱۶)	۱۴.۵۵(۰.۲۲)	۱۱.۳۸(۰.۲۳)	۱۰۰	۱۰۰
	۱۲.۸۷(۰.۳۸)	۸.۱۹(۰.۱۷)	۱۳.۴۰(۰.۱۳)	۱۱.۳۴(۰.۲۷)	۲۰۰	۲۰۰
۱۰۰۰	۱۰.۳۲(۰.۰۸)	۱۸.۶(۱.۶)	۱.۹۴(۰.۰۸)	۶.۷۷(۰.۱۲)	۱	۱
	۷.۹۴(۰.۰۸)	۶.۵۶(۰.۳۸)	۱۳.۷۷(۰.۴۷)	۵.۳۰(۰.۳۲)	۵	۵
	۴.۶۰(۰.۱۹)	۵.۷۱(۰.۲۲)	۱۵.۵۵(۰.۵۱)	۵.۱۵(۰.۱۳)	۱۰	۱۰
	۴.۸۹(۰.۰۹)	۵.۹۰(۰.۱۷)	۱۵.۵۴(۰.۴۵)	۵.۳۲(۰.۱۸)	۲۰	۲۰
	۵.۱۰(۰.۰۹)	۶.۰۲(۰.۱۴)	۱۴.۳۸(۰.۳۸)	۵.۹۳(۰.۲۰)	۳۰	۳۰
	۵.۱۱(۰.۱۰)	۶.۲۲(۰.۱۶)	۱۴.۱۱(۰.۳۳)	۵.۹۴(۰.۱۸)	۴۰	۴۰
	۶.۲۱(۰.۰۹)	۹.۸۵(۰.۱۱)	۱۳.۷۵(۰.۲۹)	۵.۵۵(۰.۱۴)	۵۰	۵۰
	۶.۷۷(۰.۱۱)	۸.۷۷(۰.۰۹)	۱۲.۲۷(۰.۲۳)	۶.۲۷(۰.۱۴)	۱۰۰	۱۰۰
	۶.۸۰(۰.۱۰)	۸.۰۶(۰.۰۷)	۱۱.۳۲(۰.۱۶)	۶.۰۱(۰.۱۱)	۲۰۰	۲۰۰
۵۰۰۰	۰.۹۵(۰.۰۷)	۳.۴۴(۰.۱۱)	۵.۰۷(۰.۱۷)	۲.۵۵(۰.۱۲)	۱	۱
	۱.۸۶(۰.۰۸)	۲.۹۴(۰.۰۷)	۱.۸۱(۰.۰۷)	۱.۶۸(۰.۱۱)	۵	۵
	۱.۷۲(۰.۰۸)	۳.۱۱(۰.۰۶)	۱.۷۵(۰.۰۶)	۱.۷۸(۰.۰۵)	۱۰	۱۰
	۲.۰۷(۰.۱۲)	۳.۳۶(۰.۰۶)	۲.۷۴(۰.۰۷)	۲.۶۰(۰.۰۷)	۲۰	۲۰
	۲.۶۹(۰.۱۰)	۳.۲۸(۰.۰۵)	۳.۲۷(۰.۱۱)	۲.۹۳(۰.۰۸)	۳۰	۳۰
	۲.۸۷(۰.۱۰)	۳.۲۶(۰.۰۴)	۳.۶۰(۰.۰۸)	۳.۱۴(۰.۰۸)	۴۰	۴۰
	۲.۹۰(۰.۱۰)	۳.۲۲(۰.۰۵)	۳.۶۵(۰.۱۱)	۳.۲۶(۰.۰۸)	۵۰	۵۰
	۳.۱۴(۰.۱۱)	۳.۰۶(۰.۰۴)	۳.۹۳(۰.۰۸)	۳.۴۱(۰.۰۷)	۱۰۰	۱۰۰
	۳.۹۳(۰.۱۲)	۲.۸۴(۰.۰۳)	۳.۸۶(۰.۰۷)	۳.۴۰(۰.۰۶)	۲۰۰	۲۰۰



ج



ب



الف

شکل (۳): نمونه‌ای از نمودارهای الگوریتم پیشنهادی با تعداد قله‌های ۱۰ در فرکانس‌های ۵۰۰ و ۱۰۰۰ و ۵۰۰۰

۶- نتیجه گیری

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 مخصوصاً 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 نتیجه بدتر می‌شود. بنابراین اگر بتوان از ابتدا تعداد دسته‌ها 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

- [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 Intelligence, Lecture Notes in Computer Science, vol. 5821, pp. 422-433, (2009).
- [14] Li, C., Yang, S.: Fast Multi-Swarm Optimization for Dynamic Optimization Problems. In: Fourth International Conference on Natural Computation, Jinan, Shandong, China, vol. 7, pp. 624-628 (2008)
- [15] D. Parrott and X. Li, "Locating and Tracking Multiple Dynamic Optima by A Particle Swarm Model Using Speciation," in IEEE Transaction on Evolutionary Computation, vol. 10, No. 4, pp. 440-458, (2006).
- [16] S. Janson and M. Middendorf. "A hierarchical 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] 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. HaghiriChehreghani, 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 International Conference on Telecommunication (IST2010), Tehran, Iran, (2010).