

## یک الگوریتم ترکیبی برای بهینه‌یابی در محیط پویا با استفاده از الگوریتم جستجوی هارمونی و الگوریتم گروه ذرات

طیبه طاهری<sup>۱</sup>، بابک نصیری<sup>۲</sup> و محمدرضا میبیدی<sup>۳</sup>

<sup>۱</sup> دانشکده برق، رایانه و فناوری اطلاعات، دانشگاه آزاد اسلامی، قزوین [t.taheri@qiau.ac.ir](mailto:t.taheri@qiau.ac.ir)

<sup>۲</sup> دانشکده برق، رایانه و فناوری اطلاعات، دانشگاه آزاد اسلامی، قزوین [nasiri.babak@qiau.ac.ir](mailto:nasiri.babak@qiau.ac.ir)

<sup>۳</sup> دانشکده مهندسی کامپیوتر، فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران [mmeybodi@aut.ac.ir](mailto:mmeybodi@aut.ac.ir)

### چکیده

بیشتر مسائل دنیای واقعی پویا بوده و تابع هدف، متغیرهای محیطی یا محدودیت‌های آن در 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 تا بتوانند الزامات محیط‌های پویا را برآورده سازند. در این مقاله از یک رهیافت جدید مبتنی بر ترکیب دو الگوریتم جستجوی هارمونی و گروه ذرات برای بهینه‌سازی در محیط پویا استفاده شده است. بدین منظور، از الگوریتم گروه ذرات به منظور جستجوی سراسری و از الگوریتم جستجوی هارمونی برای جستجوی محلی استفاده شده است. نتایج حاصل از رهیافت پیشنهادی بر روی معیار قله‌های متحرک که در حال حاضر شناخته شده‌ترین معیار برای ارزیابی در محیط‌های پویا می‌باشد، با نتایج حاصل از چندین الگوریتم معتبر مورد مقایسه قرار گرفته است. نتایج آزمایشات نشان دهنده صحت و کارایی روش ارائه شده می‌باشد.

**واژه‌های کلیدی** - الگوریتم جستجوی هارمونی<sup>۱</sup>، الگوریتم گروه ذرات<sup>۲</sup>، محیط پویا، معیار قله‌های متحرک.

دهد. برای ایجاد یا نگهداری تنوع دسته در محیط، روشهای زیادی پیشنهاد شده است.

### ۱- مقدمه

در نظر گرفتن هر تغییر به عنوان ورودی یک مسئله بهینه‌سازی 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. بکارگیری اطلاعات قبلی، یک تلاش طبیعی برای سرعت بخشیدن به فرآیند بهینه‌سازی بعد از یک تغییر است. 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)[2][3]، الگوریتم ماهی‌های مصنوعی (AFSA)[4][5]، و غیره استفاده شده

اکثر مسائل در 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 بتوانند با آن مقابله کنند. استفاده از روشهای تکاملی یکی از روشهای برخورد با عدم قطعیت است [1]. 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. دو مشکل عمده در روشهای پردازش تکاملی حافظه غیر معتبر و از دست دادن تنوع است. این دو مشکل کارایی روشهای پردازش تکاملی در محیط‌های 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.

کاربردهای بهینه‌سازی از قبیل: دسته‌بندی داده‌ها [12]، بهینه‌سازی ترکیبی [13][14][15] و غیره استفاده شده است. مراحل الگوریتم HS به ترتیب زیر می‌باشد:

1. تعریف مسئله و پارامترها. 2. مقدار دهی اولیه حافظه هارمونی.
3. تولید هارمونی جدید. 4. به روز رسانی حافظه هارمونی.
5. بررسی شرط پایان الگوریتم.

شبه کد الگوریتم HS در شکل 1 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 1: شبه کدی از الگوریتم جستجوی هارمونی استاندارد [10]

### 3- الگوریتم گروه ذرات

بهینه‌سازی گروه ذرات (PSO) تکنیکی مبتنی بر جمعیت است و از رفتار اجتماعی گروه پرندگان الهام گرفته شده است. این الگوریتم اولین بار توسط کندی و آبرهات در سال 1995 طراحی شد [16]. هر ذره در گروه با توجه به سرعت و موقعیتی که دارد راه‌حلی را پیشنهاد می‌کند و PSO سعی می‌کند راه‌حل بهینه را از بین راه‌حلهای موجود پیدا کند. در گام بعدی همه ذرات سرعت و موقعیت خود را با استفاده از فرمولهای (1) و (2) به روز می‌کنند.

$$V_i(t+1) = V_i(t) + c_1 r_1 (p_i^{\text{best}} - p_i(t)) + c_2 r_2 (l_{\text{best}} - p_i(t)) \quad (1)$$

$$p_i(t+1) = p_i(t) + V_i(t+1) \quad (2)$$

که در آن  $c_1$  و  $c_2$  دو مقدار ثابتی هستند که جاذبه نسبی را در بهینه سراسری و بهترین راه‌حل شخصی هر ذره را کنترل می‌کنند.  $r_1$  و  $r_2$  دو مقدار تصادفی در بازه  $[0,1]$  هستند،  $p_i^{\text{best}}$  بهترین موقعیت شخصی ذره  $i$  و  $l_{\text{best}}$  بهترین موقعیت محلی است که از همه ذرات همسایه  $i$  بدست آمده است.

شبه کد PSO در شکل 2 نشان داده شده است.

است. در این الگوریتم‌ها سعی شده با بهبود برخی از ویژگی‌های فوق نتایج بهتری حاصل شود.

در این مقاله یک الگوریتم بهینه‌سازی مبتنی بر ترکیب دو الگوریتم HS و PSO پیشنهاد می‌گردد که برای عمل در محیط‌های پویا طراحی شده است و تمام الزامات محیط‌های پویا را برآورده می‌کند. در رهیافت پیشنهادی مکانیزم HS استاندارد برای جستجوی محلی بکار گرفته شده است با این تفاوت که بعد از هر بار تکرار، از یک روش مبتنی بر شعاع همگرایی بهترین هارمونی استفاده شده است. الگوریتم PSO استاندارد نیز در قسمت جستجوی سراسری مورد استفاده قرار گرفت، چون کل فضای جستجو باید سریع بررسی گردد تا همه قله‌ها کشف شوند بنابراین از PSO که سرعت جستجوی بالایی دارد بکار گرفته شد. الگوریتم HS محدود شده در یک محیط دایره نیز با دقت به جستجوی محلی می‌پردازد، در نتیجه ترکیب این دو الگوریتم در کنار هم باعث افزایش کارایی مدل پیشنهادی گردید. مدل پیشنهادی بر روی بنچمارک قله‌های متحرک  $(\text{MPB}^4)$  [6] که از معروفترین بنچمارک‌های محیط پویاست ارزیابی شده و کارایی آن با چهار الگوریتم معتبر دیگر بنام‌های، mQSO [2]، cellular-PSO [7]، fmso [8]، rpsso [9] مقایسه شده است. معیار مقایسه خطای برون خطی<sup>5</sup> 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: بخش‌های 2 و 3 الگوریتم جستجوی هارمونی و الگوریتم گروه ذرات را به طور مختصر شرح می‌دهند، بخش 4 توضیح مختصری در مورد کارهای انجام گرفته در زمینه محیط‌های پویا ارائه کرده، بخش 5 الگوریتم پیشنهادی را توضیح می‌دهد. در بخش 6 نتایج آزمایشات مورد بررسی قرار می‌گیرد و بخش 7 به بیان نتیجه‌گیری می‌پردازد.

### 2- الگوریتم جستجوی هارمونی

الگوریتم جستجوی هارمونی (HS) یکی از الگوریتم‌های الهام گرفته شده از طبیعت است. این الگوریتم اولین بار توسط ژونگ‌هو جیم در سال 2001 ارائه گردید [10]. الگوریتم جستجوی هارمونی برگرفته شده از فعالیت موسیقی موسیقی‌دانان است که به دنبال پیدا کردن یک هارمونی کامل و بی نقص هستند [11]. از جمله مزایای آن می‌توان به غیر حساس بودن به مقادیر اولیه، جایگزین کردن جستجوی تصادفی بجای جستجوی گرایانی و استفاده کمتر از ریاضیات اشاره کرد [11]. این الگوریتم در

شده و همچنین این دسته کل فضای جستجو را بررسی می کند. گروه-های دیگر گروه فرزندان را تشکیل می دهند که با استفاده از الگوریتم fpos [23] به جستجوی محلی می پردازند و بهینه های محلی را کشف می کنند.

#### 5- الگوریتم پیشنهادی

در این الگوریتم همانند [2] از چند دستگی استفاده شده است. بدین صورت که کل هارمونی های موجود را به دسته های از پیش تعیین شده ای تقسیم کرده است. ارتباط بین دسته ها به دوصورت محلی براساس  $T_{exle}$  و سراسری بر اساس  $T_{conv}$  می باشد [2]. در واقع هنگامی که تمام دسته ها همگرا شده و تعداد قله ها بیش از تعداد دسته ها باشد، دسته ای که کمترین مقدار شایستگی را داشته باشد دوباره مقدار دهی اولیه می شود. همچنین اگر دو دسته در محدوده  $T_{exle}$  باشند دسته با کمترین مقدار شایستگی برای مقدار دهی اولیه انتخاب می شود. این کار بدین منظور است که تمام قله ها تحت پوشش قرار گیرند. وقتی که تعداد قله ها بیش از تعداد دسته ها باشد بعضی از قله ها هیچ وقت شناسایی نمی شوند، بنابراین همانطور که در [2] پیشنهاد شده در چنین مواقعی دسته ای که کمترین مقدار شایستگی را دارد، برای مقداردهی اولیه انتخاب می شود تا شانس پیدا کردن قله های دیگر که احتمالاً 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 هارمونی های موجود را به دسته های از پیش تعیین شده ای تقسیم می کند و مقدار شایستگی هر هارمونی محاسبه می شود. سپس بهترین هارمونی هر دسته مشخص شده و سایر هارمونی های آن دسته را به اندازه شعاع همگرایی دسته طبق رابطه 3 حرکت می دهد.

$$X_i(t+1) = x_i(t) + (R_{cloud} \times rand) \quad (3)$$

که در رابطه (3)  $x_i$  بردار هارمونی ام و  $R_{cloud}$  شعاع همگرایی دسته، rand یک عدد تصادفی در فاصله [-1,1] است. با این روند هارمونی های هر دسته در یک دایره به مرکزیت بهترین هارمونی و شعاع همگرایی آن دسته قرار می گیرند. این عمل باعث می شود تا هارمونی های 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 می شود. مقدار شایستگی هارمونی جدید هر دسته با بدترین هارمونی آن دسته مقایسه می شود و در صورتی که از آن بهتر بود جایگزین بدترین هارمونی

For each particle:

Initialize particle

DO:

1: for each particle

Calculate fitness value

If the fitness value is better than the best fitness value (pBest) in history

Set current value as the new pbest

End

2: for each particle

Find in the particle neighborhood, the particle with the best fitness

Calculate particle velocity according to the velocity equation(1)

Apply the velocity constriction

Update particle position according to the position equation(2)

Apply the position constriction

End

While maximum iteration or minimum error criteria is not attained

شکل 2: شبه کدی از الگوریتم گروه ذرات [16]

#### 4- کارهای گذشته

از بین چالش های موجود در محیط های پویا مهمترین چالش مسئله از دست دادن تنوع می باشد. برای حل این مشکل راهکارهایی ارائه شده است. تعدادی از این راهکارها، تصادفی سازی در تمام یا بخشی از گروه را دربرمی گیرد [17]. تعدادی دیگر، از راه حل دفع کردن برای ایجاد تنوع استفاده می کنند [18-20]. تنظیم ساختار اشتراک اطلاعات، راه دیگری است که می تواند با هدف کاهش موقتی انگیزه برای جابجایی به سوی بهترین موقعیت سراسری باشد و از این راه تنوع جمعیت را افزایش دهد [21، 22]. در بسیاری دیگر برای این منظور از مدل چند گروهی استفاده شده است. مدل mpsو تنوع را در دو سطح حفظ می کند. در الگوریتم spso تمام ذرات بر اساس ترتیب نزولی مقدار برازندگی بهترین های شخصی هر ذره مرتب می شود.

در cellular pso که ترکیبی از توانایی pso با آتاماتای سلولی است با جاسازی آتاماتی سلولی در فضای جستجو، تراکم ذرات را در هر سلول به حداکثر میزان معین شده نگه می دارد و زمانی که تراکم ذرات در یک سلول به بیش از میزان تعیین شده می رسد، گروهی از ذرات این سلول پرجمع، انتخاب شده و مقداردهی اولیه می گردند. در این الگوریتم با انجام تعاملات محلی بین سلولها، ذرات هر سلول به همراه ذرات سلولهای همسایه باهم به جستجوی یک قله می پردازند.

مدل mqso یک سیستمی بر مبنای ذرات کوانتوم می باشد. هدف از این روش، استفاده از جمعیت فراوانی از ذرات بر روی قله های مختلف است که این هدف با استفاده از افزودن عملگر ضد همگرایی تامین می گردد. دلیل عمده توانایی این مدل و برتری آن نسبت به روش pso استاندارد، استفاده از ذرات موازی و وجود خاصیت مقداردهی اولیه مجدد است.

در fmso از چندین دسته استفاده می شود تا تنوع را در طول اجرا حفظ کند. در این روش از یک دسته والد برای نگهداری تنوع استفاده

```
// initialize swarms
for each swarm i
    for each harmony j
        initialize  $x_{ij}$ 
         $pbest_{ij} = x_{ij}$ 
    end
     $Gbest = \arg \min f(pbest_{ij})$ 
end
// initialize swarm finder
for each harmony finder
    initialize  $x_{ij}$ 
end
repeat:
    for each swarm i
        for each harmony j
            update  $x_{ij}$  based on equation 1
        end
        call HS Algorithm
    end
    call pso 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 3: شبه کد الگوریتم پیشنهادی

## 6- نتایج آزمایشات

اکثر شبیه سازی تغییرات محیط پویا در زمان، روی MPB که یکی از شناخته شده ترین معیارها برای ارزیابی است، صورت می گیرد. محیط آن شامل تعداد مشخصی قله با محل، ارتفاع و عرضهای مختلف می باشد. برازندگی محیط برابر ماکزیمم مقدار در میان توابع قله تعریف می شود. معیار ارزیابی خطای برون خطی است که میانگین حرکت بین بهترین راه حل شناخته شده توسط الگوریتم و برازندگی بهینه سراسری است که فرمول آن در رابطه 4 آورده شده است.

$$Offline\_error = \frac{1}{FEs} \sum_{t=1}^{FEs} (fitness(gbest(t)) - fitness(global - optimum(t))) \quad (4)$$

که در آن، 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. برای ارزیابی الگوریتم پیشنهادی از سیستمی با پردازنده 1.6 گیگاهرتز، حافظه 1 گیگابایت و سیستم عامل xp استفاده شده است. شبیه سازی در برنامه matlab(2012) پیاده سازی شده است و برای بررسی کارایی آن، نتایج بدست آمده از این الگوریتم با نتایج چهار الگوریتم شناخته شده دیگر به نامهای mQSO [2]، cellular pso [7]، fms [8]،

موجود می شود. الگوریتم HS تضمین می کند که در هر تکرار هارمونی بهتری در صورت وجود جایگزین شود. این روند تا جایی ادامه پیدا می کند که دسته مورد نظر به یک قله همگرا شود. زمانی که دسته به یک قله همگرا شد از یک تکنیک به نام خواباندن دسته استفاده می کنیم. بدین معنی که آن دسته را از چرخه جستجو خارج کرده تا تعداد مقایسه ها کاهش یابد و عمل جستجو روی سایر دسته ها متمرکز شود. در الگوریتم پیشنهادی دسته دیگری به عنوان دسته ناظر ایجاد کرده تا کل محیط را جستجو کند که اگر قله ای وجود داشت که توسط هیچ دسته ای شناسایی نشده بود دسته ای را که هیچ قله ای را پیدا نکرده به آن معرفی کند. و خود دوباره به محیط باز گشته و به جستجوی قله های ناشناس می پردازد. دسته ناظر از الگوریتم pso معرفی شده در بخش 3 برای بهبود هارمونی های خود استفاده می کند تا بتواند کل محیط را با سرعت بالایی بررسی کند. استفاده از pso با سرعت جستجوی بالا به همراه HS با دقت بالا در بهینه یابی محلی، باعث افزایش کارایی رویکرد پیشنهادی می شود.

همانطور که از رابطه 3 مشاهده می شود مقدار شعاع  $R_{cloud}$  در تعیین موقعیت سایر هارمونی ها 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 می رود، زیرا هارمونی های جدید در محدوده کوچکی از فضا ایجاد خواهد شد و شباهت زیادی به سایر هارمونی ها خواهد داشت اما باعث کاهش طول گام حرکت در هر تکرار و در نتیجه کاهش سرعت همگرایی الگوریتم می شود و از توانایی تنوع دسته نیز کاسته می شود. در صورتی که مقدار این شعاع بزرگ در نظر گرفته شود، احتمال ایجاد هارمونی های بهتر کاهش می یابد و توانایی جستجوی محلی نیز کم می شود. در واقع مقدار این پارامتر باید به گونه ای 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 قوی تر به نتایج بهتری دست یابد. در الگوریتم پیشنهادی برای دستیابی به این هدف مقدار  $R_{cloud}$  را دوبرابر طول گام حرکت قله ها در نظر گرفته سپس با استفاده از یک رهیافت خود تطبیقی مقدار این پارامتر کاهش می یابد.

شبه کد الگوریتم پیشنهادی در شکل 3 نشان داده شده است.

بدتری دارد بازنده شده و مجبور است دوباره مقدار دهی شود و این باعث می شود نتایج بدست آمده با حالتی که تعداد قله ها برابرند اختلاف چشمگیری داشته باشد.

جدول ۱: پارامترهای MPB

پارامتر	مقدار
M تعداد قله ها	بین ۱ تا ۲۰۰
فرکانس تغییر	۵۰۰، 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 قله ها	[۰، ۱۰۰، ۰]
محدوده پارامتر ارتفاع	[۳۰، ۷۰]
محدوده پارامتر عرض	[۱، ۲]
مقدار ارتفاع اولیه قله ها	۵۰

rpso [9] مقایسه شده است. نتایج با توجه به پارامترهای جدول ۱ که با نام سناریوی ۲ در مسئله MPB معروف است، بدست آمده اند. تنها تفاوت آن با سناریوی ۲، در تعداد قله ها است که بین ۱ قله تا ۲۰۰ قله در فرکانس های تغییر ۵۰۰، ۱۰۰۰، ۵۰۰۰ در نظر گرفته شده است. نتایج بدست آمده از خطای برون خطی و خطای استاندارد بر روی قله های مختلف در جدول (۲) نشان داده شده است. خطای استاندارد درون پراکنش نوشته شده است. در این جدول p نشان دهنده تعداد قله ها است. در جدول (۲)، بهترین نتیجه هر سطر بصورت پررنگ مشخص شده است. همانطور که مشاهده می شود، الگوریتم پیشنهادی نتایج قابل قبولی را ارائه کرده است. علی الخصوص زمانی که تعداد قله ها با تعداد دسته ها برابر است. با توجه به جدول مشاهده می شود، زمانی که تعداد قله ها کمتر از تعداد دسته ها است چون ممکن است چند دسته بر روی یک قله با هم رقابت کنند، در نتیجه دسته ای که شایستگی

جدول (۲): خطای برون خطی و خطای استاندارد الگوریتم ها در فرکانس های تغییرات ۵۰۰ و ۱۰۰۰ و ۵۰۰۰ با تعداد قله های مختلف

frequency	P	Cellular PSO	rPSO	mQSO	FMSO	الگوریتم پیشنهادی
500	1	13.4(0.74)	<b>4.27(0.17)</b>	33.67(3.4)	27.58(0.94)	15.09(0.74)
	5	9.63(0.49)	16.19(0.42)	11.91(0.76)	19.45(0.45)	<b>9.13(0.62)</b>
	10	9.42(0.21)	17.34(0.27)	9.62(0.34)	18.26(0.32)	<b>5.95(0.57)</b>
	20	8.84(0.28)	17.06(0.37)	8.79(0.23)	17.34(0.30)	<b>8.21(0.41)</b>
	30	<b>7.88(0.23)</b>	16.98(0.27)	8.80(0.21)	16.39(0.48)	8.90(0.41)
	40	<b>7.83(0.21)</b>	16.46(0.28)	8.55(0.21)	15.34(0.45)	8.83(0.65)
	50	8.62(0.23)	15.77(0.18)	8.56(0.2)	15.54(0.26)	<b>8.55(0.24)</b>
	100	11.38 (0.23)	14.55(0.22)	<b>8.54(0.16)</b>	12.87(0.60)	9.12(0.32)
	200	11.34(0.27)	13.40(0.13)	<b>8.19(0.17)</b>	11.52(0.16)	12.07(0.38)
1000	1	<b>6.77(0.12)</b>	1.94(0.08)	18.6(1.6)	14.42(0.48)	11.32(0.78)
	5	<b>5.30(0.32)</b>	13.77(0.47)	6.56(0.38)	10.59(0.24)	7.98(0.58)
	10	5.15(0.13)	15.55(0.51)	5.71(0.22)	10.40(0.17)	<b>4.05(0.19)</b>
	20	<b>5.32(0.18)</b>	15.54(0.45)	5.90(0.17)	10.33(0.13)	5.89(0.09)
	30	5.93(0.20)	14.38(0.38)	6.02(0.14)	10.06(0.14)	<b>5.08(0.09)</b>
	40	5.94(0.18)	14.11(0.33)	6.22(0.16)	9.85(0.11)	<b>4.79(0.10)</b>
	50	<b>5.55(0.14)</b>	13.75(0.29)	5.90(0.17)	9.54(0.11)	6.01(0.09)
	100	<b>6.27(0.14)</b>	12.27(0.23)	6.48(0.15)	8.77(0.09)	6.77(0.11)
	200	<b>6.01(0.11)</b>	11.32(0.16)	6.18(0.13)	8.06(0.07)	6.83(0.10)
5000	1	2.55(0.12)	<b>0.56(0.04)</b>	5.07(0.17)	3.44(0.11)	1.95(0.37)
	5	<b>1.68(0.11)</b>	12.22(0.76)	1.81(0.07)	2.94(0.07)	1.82(0.08)
	10	1.78(0.05)	12.98(0.48)	1.75(0.06)	3.11(0.06)	<b>1.71(0.08)</b>
	20	2.60(0.07)	12.79(0.54)	2.74(0.07)	3.36(0.06)	<b>2.07(0.10)</b>
	30	2.93(0.08)	12.35(0.62)	3.27(0.11)	3.28(0.05)	<b>2.61(0.10)</b>
	40	3.14(0.08)	11.37(0.41)	3.60(0.08)	3.26(0.04)	<b>2.89(0.10)</b>
	50	3.26(0.08)	11.34(0.29)	3.65(0.11)	3.22(0.05)	<b>2.99(0.10)</b>
	100	3.41(0.07)	9.73(0.28)	3.93(0.08)	<b>3.06(0.04)</b>	3.14(0.11)
	200	3.40(0.06)	8.90(0.19)	3.86(0.07)	<b>2.84(0.03)</b>	3.90(0.10)

کند. در این فرکانس زمانی که تعداد قله ها با تعداد دسته ها برابر است مدل پیشنهادی اختلاف چشمگیری با بقیه الگوریتم ها داشته. اما زمانی

در فرکانس ۵۰۰ زمانی که تعداد قله ها ۵، ۱۰، ۲۰، ۵۰ است مدل پیشنهادی بهتر از سایر الگوریتم های ذکر شده در جدول ۲ عمل می -



- [4] 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.
- [5] 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.
- [6] <http://www.aifb.unikarlsruhe.de/~jbr/MovPeaks/>
- [7] 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.
- [8] H. Richter, "Memory Design for Constrained Dynamic Optimization Problems," Applications of Evolutionary Computation, pp. 552-561, 2010.
- [9] D. Parrott and X. Li, "Locating and Tracking Multiple Dynamic Optima by A Particle Swarm Model Using pectiation," in IEEE Transaction on Evolutionary Computation, vol. 10, No. 4, pp. 440-458, 2006.
- [10] Geem, Z.W., Kim, J.H., Loganathan, G.V., "A new heuristic optimization algorithm: Harmony search". Simulation 76, 60-68, 2001.
- [11] Xin-She Yang, "Harmony Search as a Metaheuristic Algorithm", Springer-Verlag Berlin Heidelberg, vol. 191, 1-14, 2009.
- [12] M. Mahdavi, M. Haghiri Chehreghani, H. Abolhassani, R. Forsati, "Novel meta-heuristic algorithms for clustering web documents", Applied Mathematics and Computation 201, 441-451, 2008.
- [13] 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.
- [14] 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.
- [15] 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.
- [16] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization", in IEEE International Conference on Neural Networks, Piscataway, NJ, pp. 1942-194, 1995.
- [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.
- [22] S. Janson and M. Middendorf. "A hierarchical particle swarm optimizer for dynamic optimization problems," In:

که تعداد قله‌ها 1 و 100 و 200 شد این الگوریتم کمی ضعیف عمل می‌کند. در فرکانس 1000 مدل پیشنهادی روی تعداد قله‌های 10 و 30 و 40 خوب جواب داد. اما باز هم در این فرکانس روی تعداد قله‌های 1 و 5 و 10 و 200 نتیجه مطلوبی نداشت. در فرکانس 5000 این الگوریتم بهتر از فرکانس‌های 1000 و 500 عمل کرده و روی تعداد قله‌های 10 و 20 و 30 و 40 و 50 نتیجه قابل توجه‌تری نسبت به بقیه داشته است.

باتوجه به موارد ذکر شده می‌توان نتیجه گرفت که زمانی که تعداد قله‌ها و تعداد دسته‌ها باهم یکسان هستند چون هر دسته مسئول یافتن یک قله می‌شود و همه قله‌ها تحت پوشش قرار می‌گیرند نتیجه بدست آمده رضایت بخش خواهد بود. در فرکانس‌های بالا چون محیط خیلی دیرتر تغییر می‌کند بنابراین الگوریتم فرصت بیشتری دارد تا بتواند محیط را جستجو کرده و قله‌ها را شناسایی کند. بنابراین برای کارایی بالاتر در محیط‌های پویا با استفاده از چند دستگی، اگر بتوان بین تعداد قله‌ها و تعداد دسته‌ها هماهنگی ایجاد کرد و تعداد دسته‌ها را مطابق تعداد قله‌ها انتخاب کرد نتیجه بدست آمده مطمئناً رضایت بخش خواهد بود.

## 7- نتیجه‌گیری

در این مقاله از الگوریتم جستجوی هارمونی محدود شده به شعاع همگرایی و همچنین مکانیزم خواباندن دسته برای جستجوی محلی در کنار الگوریتم گروه ذرات به منظور جستجوی سراسری استفاده شده و نتایج آنرا که روی تابع بنچمارک قله‌های متحرک اعمال شده با نتایج چندین روش شناخته شده دیگر مقایسه گردید. نتایج آزمایشات نشان داد که روش پیشنهادی کارایی قابل قبولی مخصوصاً زمانی که تعداد قله‌ها نزدیک به تعداد دسته‌ها باشد، دارد. بهترین نتیجه مربوط به سطری است که تعداد قله‌ها و دسته‌ها برابر است. بنابراین اگر بتوان از ابتدا تعداد دسته‌ها را نیز مطابق تعداد قله‌ها انتخاب کرد 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] 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.
- [2] 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.
- [3] T. Blackwell and J. Branke, "Particle Swarms for Dynamic Optimization Problems", in Swarm Intelligence, pp.193-217, 2008.



G. R. Raidl, editor, APPLications of Evolutionary Computing, volume 3005 of Lecture Notes in Computer Science. Springer, Berlin, Germany, pp.513–524, 2004.

- [23] C. Li and S. Yang, "Fast Multi-Swarm optimization for Dynamic Optimization Problems" in Fourth International Conference on Natural Computation, Jinan, Shandong, China, 624-628, 2008.

---

<sup>1</sup>Harmony Search Algorithm

<sup>2</sup>Particle Swarm Optimization

<sup>3</sup>Artificial Fish Swarm Algorithm

<sup>4</sup>Moving Peak Benchmark

<sup>5</sup>Offline Error