یک الگوریتم ترکیبی مبتنی بر آتاماتاهای یادگیر و نظریه بازی‌ها برای بهینه‌سازی

مهدی رضاپور میرصالح، محمدرضا میبدی، محمد مهدی عبادزاده
دانشگاه صنعتی امیرکبیر – دانشکده مهندسی کامپیوتر و فناوری اطلاعات
mrezapoorm, mmeybodi, ebadzadeh @aut.ac.ir

چکیده

آتاماتای یادگیر ابزاری قوی است که در یک محیط تصادفی ناشناخته عمل کرده و به مرور زمان کارایی خود را از طریق یک فرایند یادگیری بهبود می‌دهد. آتاماتاهای یادگیر در حل مسائل بهینه سازی بسیار خوب عمل می‌کنند و یکی از ویژگی‌های بارز آنها قابلیت یادگیری می‌باشد. مساله بهینه سازی را می‌توان یافتن نقطه تعادل یک بازی دانست که در آن هر بازیکن یک مقدار از بعد متناظر خود در فضای جستجو را انتخاب می‌کند. در این مقاله یک الگوریتم ترکیبی تکاملی که از ترکیب آتاماتای یادگیر و مفاهیم نظریه بازیها حاصل می‌شود برای حل مسائل بهینه سازی پیشنهاد می‌گردد. آزمایش‌های انجام شده کارایی این روش را به خوبی نشان می‌دهد.

واژگان کلیدی

آتاماتای یادگیر، الگوریتم تکاملی، بهینه سازی، نظریه بازی‌ها

۱- مقدمه

مسایل بهینه سازی در فضاهای پیوسته در مسایل مختلف علوم ارتباطات، تجارت، طراحی مهندسی و ... به کار برده می‌شود. در دهه های گذشته روش‌های اکتشافی فراوانی پیشنهاد شده است که این روش‌ها نسبت به روش‌های سنتی و عددی دارای انعطاف پذیری بیشتری بوده و در حوزه‌های مختلف قابل استفاده است. از روش‌های اکتشافی در این زمینه می‌توان شبیه‌سازی حرارت، جستجوی تابو، الگوریتم ژنتیک، استراتژی تکامل بهینه سازی اجتماع ذرات، الگوریتم تکامل تفاضلی و سیستم ایمنی مصنوعی را نام برد. الگوریتم تکاملی، جستجوی خلاقانه‌ای است که در فضای جستجو به دنبال راه حل بهینه مسئله می‌گردد. این الگوریتم‌ها از مفهوم تکامل در موجودات زنده الهام گرفته شده‌اند. در این الگوریتم‌ها جمعیتی از کروموزوم‌های اولیه انتخاب شده و با تکامل این کروموزوم‌ها توسط عملگرهای تکاملی، کروموزوم‌ها که در برگیرنده راه‌حل‌های مسئله می‌باشند، به سمت راه حل بهینه حرکت می‌نمایند. کارایی بالای آتاماتای یادگیر در محیط‌های تصادفی و ناشناخته و همچنین قابلیت یادگیری آن باعث می‌شود که بتوان با ترکیب آتاماتای یادگیر و الگوریتم‌های تکاملی به الگوریتم‌های جدیدی دست پیدا کرد که کارایی قابل قبولی داشته باشند. در مساله بهینه‌سازی، هدف یافتن نقطه بهینه سراسری یک تابع است. در بعضی از توابع که علاوه بر می‌نیمم سراسری، تعدادی می‌نیمم محلی دیگر نیز وجود دارد، یافتن نقطه بهینه سراسری به آسانی امکان پذیر نیست. در چنین مسائلی می‌توان این مساله را مشابه با یافتن نقطه تعادل در بازی پیمایش سطح تابع توسط بازیکنانی دانست که هر بازیکن مقدار عددی بعد متناظر با خود را از فضای جستجو انتخاب می‌کند. سازماندهی ادامه این مقاله بدین صورت است: بخش ۲ شامل تعریف مساله و بررسی کلی مکانیزم انجام آن است. در بخش ۳ ساختار آتاماتای یادگیر مورد بررسی قرار می‌گیرد. در بخش ۴ الگوریتم جدید ترکیبی که مبتنی بر آتاماتای یادگیر می‌باشد، ارائه شده است. بخش ۵ به بررسی آزمایشات انجام شده اختصاص داده شده است و نهایتا در بخش ۶ به جمعبندی موارد مطرح شده پرداخته شده است.

۲- تعریف مساله

مسائل زیادی در علوم مهندسی، ریاضی و کاربردی منجر به بهینه سازی سراسری تابعی خاص می‌گردند. مساله بهینه سازی سراسری پیوسته را می‌توان به صورت زیر تعریف کرد.

$$\arg\max_{x \in D} f(x) \qquad (1)$$

به گونه ای که در آن

$$x = (x_1, x_2, \ldots, x_n) \in R^n, x_i \in [x_i^l, x_i^u], i = 1, 2, \ldots, n$$

$f(x)$ تابع هدف است و $D = \prod[x_i^l, x_i^u] \subseteq R^n$ فضای جستجو را نشان می‌دهد.

در این مساله هدف یافتن x^* به عنوان جواب بهینه مساله (1) است به گونه ای که $f(x^*) = \max\{f(x) | x \in D\}$

در روش ارائه شده، مقدار عددی هر بعد از بردار n بعدی x، با یک رشته به طول n_x نمایش داده می‌شود که با به هم پیوستن رشته‌های مربوط به همه ابعاد، رشته ای با طول $n * n_x$ که با s نشان داده می‌شود، ایجاد می‌شود. هر عنصر این رشته را می‌توان از یک مجموعه مقادیر قابل قبول که دارای m عنصر است انتخاب نمود. در حقیقت با این روش بازنمایی می‌توان هر نقطه از فضای پیوسته را با یک رشته نمایش داد.

به عنوان مثال تابع زیر را در نظر می‌گیریم.

$$f(x_1, x_2) = x_1 + x_2 \qquad (2)$$

که در آن $x_i \in [-2.048, 2.048], i = 1, 2$

با در نظر گرفتن n_x=10 و m=2 (با فرض اینکه مجموعه عناصر قابل قبول برای تولید رشته بازنمایی برابر {0,1} باشد) می‌توان مقدار عددی هر بعد را با یک رشته ۱۰ بیتی نشان داد. به عنوان مثال اگر رشته متناظر با x_1 و x_2 برابر

$$s: \underbrace{0000110111}_{x_1}\ \underbrace{1101110001}_{x_2}$$

باشد مقادیر عددی هر بعد طبق رابطه

$$x_i = x_i^l + desimal(s_i) * (x_i^u - x_i^l)/(m^{n_x} - 1) \qquad (3)$$

که در آن $i = 1, 2, \ldots, n$ و s_i رشته مربوط به بعد x_i می‌باشد، برابر است با:

$$x_1 = 2.048 + desimal(0000110111_2) * (4.096)/(2^{10} - 1) = -1.827785$$

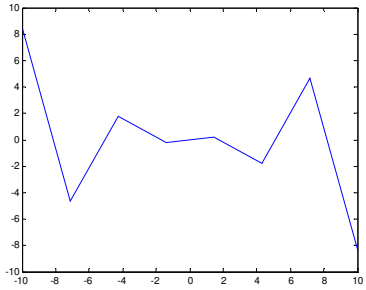
$$x_2 = -2.048 + desimal(1101110001_2) * (4.096)/(2^{10} - 1) = 1.479444$$

و در نتیجه مقدار تابع هدف برابر $f(x_1, x_2) = -0.348341$ می‌باشد.

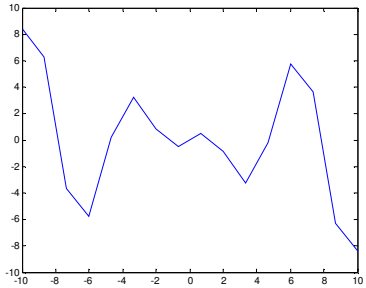
میزان دقت این روش بازنمایی وابسته به طول رشته در نظر گرفته شده(n_x)، حداقل و حداکثر محدوده مجاز برای هر بعد و تعداد عناصر قابل قبول جهت تولید رشته (m) می‌باشد. به عبارت دیگر میزان دقت این بازنمایی برابر است با:

$$(x_i^u - x_i^l)/(m^{n_x} - 1) \qquad (4)$$

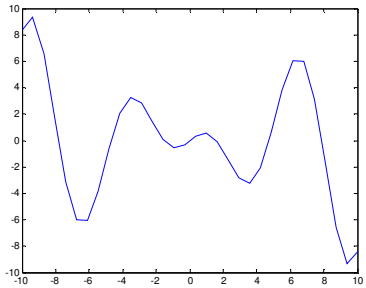
به عنوان مثال تابع $f(x)=x*cos(x)$ را در نظر بگیرید. اگر فرض کنیم که محدوده تغییر x بین [-10,10] باشد، بازنمایی این تابع با مقادیر مختلف n_x و m در شکل‌های ۱ تا ۴ نشان داده شده است.



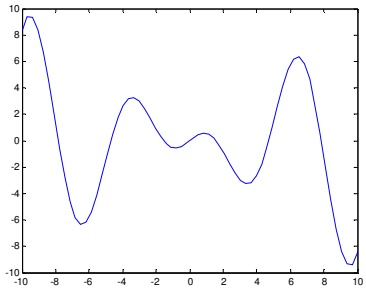
شکل ۱-نمایش تابع $f(x)=x*cos(x)$ با $m=2$ و



شکل ۲-نمایش تابع $f(x)=x*cos(x)$ با $m=2$ و



شکل ۳- نمایش تابع $f(x)=x*cos(x)$ با $m=2$ و



شکل ۴- نمایش تابع $f(x)=x*cos(x)$ با $m=2$ و

۳- آتاماتای یادگیر

آتاماتای یادگیر ماشینی است که می‌تواند تعداد محدودی عمل را انجام دهد؛ هرگاه این ماشین عملی را انتخاب می‌کند، عمل انتخاب شده توسط محیط ارزیابی شده و نتیجه آن به صورت یک سیگنال بازخوردی مثبت (در صورت مناسب بودن عمل) یا منفی (در صورت نامناسب بودن عمل) به آتاماتا بازگردانده می‌شود. مقدار این سیگنال در انتخاب اعمال بعدی تاثیر می‌گذارد. هدف این فرایند این است که آتاماتا بعد از گذشت مدتی به سمت مناسب ترین عمل خود در محیط میل کرده و یا به عبارت دیگر یاد می‌گیرد که کدام عمل بهترین عمل است. نحوه تعامل آتاماتای یادگیر و محیط در شکل ۵ نشان داده شده است. همانطور که در شکل مشاهده می‌شود، محیط یک آتاماتای یادگیر همانند یک تابع رفتار می‌کند؛ ورودی این تابع عمل انتخاب شده توسط آتاماتا و خروجی آن سیگنال بازخوردی است.

شکل ۵- تعامل آتاماتای یادگیر با محیط

مدل ریاضی محیط بصورت سه‌تایی $\{\underline{\alpha},\underline{c},\underline{\beta}\}$ تعریف می‌شود که در آن $\underline{\alpha}=\{\alpha_1,...,\alpha_r\}$ مجموعه محدود ورودی‌ها، $\underline{\beta}=\{\beta_1,...,\beta_m\}$ مجموعه محدود یا نامحدود خروجی‌های محیط و $\underline{c}=\{c_1,...,c_r\}$ مجموعه محدودی از احتمالهای تنبیه می‌باشد. هر c_i با α_i رابطه دارد و در واقع c_i ها مشخصات و رفتار محیط را تعریف می‌کنند.

برحسب نحوه تعریف مجموعه $\underline{\beta}$، مدلهای مختلفی از محیط تعریف می‌گردد، اما اغلب پاسخ محیط به عمل ورودی به یکی از دو صورت مثبت یا منفی (عمل مطلوب یا نامطلوب) در نظر گرفته می‌شود که در این حالت مجموعه $\underline{\beta}$ بصورت یک مجموعه دو عضوی $\underline{\beta}=\{\beta_1,\beta_2\}$ تعریف می‌گردد و معمولاً با دو علامت $\underline{\beta}=\{0,1\}$ نمایش داده می‌شود. در آتاماتاهای یادگیر، معمولاً 0 به عنوان پاسخ مطلوب و 1 به عنوان پاسخ نامطلوب در نظر گرفته می‌شود.

آتاماتاهای یادگیر به دو خانواده آتاماتاهای یادگیر با ساختار ثابت و آتاماتاهای یادگیر با ساختار متغیر دسته بندی می‌شوند. آتاماتاهای کرینسکی و کرایلو مثال‌هایی از آتاماتاهای با ساختار ثابت هستند. یک آتاماتای یادگیر با ساختار متغیر را می‌توان با یک پنج تایی $<\underline{\alpha},\underline{\Phi},\underline{\beta},\underline{F},\underline{G}>$ نشان داد. که $\underline{\alpha}=\{\alpha_1,...,\alpha_r\}$ مجموعه اقدامهای مجاز برای آتاماتای یادگیر، $\underline{\Phi}=\{\Phi_1,...,\Phi_s\}$ مجموعه وضعیت‌های آتاماتا، $\underline{\beta}=\{0,1\}$ مجموعه ورودی‌ها (در این مجموعه 1 نمایانگر شکست و 0 نمایانگر موفقیت می‌باشد)، $F:\underline{\Phi}\times\underline{\beta}\to\underline{\Phi}$ تابع نگاشت وضعیت‌ها و $G:\underline{\Phi}\to\underline{\alpha}$ تابع نگاشت خروجی می‌باشد. اقدام یک آتاماتا به عنوان ورودی به محیط داده می‌شود و محیط بعد از اعمال اقدام داده شده توسط آتاماتا یک پاسخ تصادفی که می‌تواند بیانگر موفقیت یا شکست اقدام اعمال شده باشد را تولید می‌کند که به عنوان ورودی به آتاماتا داده می‌شود. آتاماتا با توجه به پاسخ محیط این اقدام را جریمه می‌کند و یا به آن پاداش می‌دهد. اگر احتمال تغییر وضعیت‌ها در آتاماتا ثابت باشد آن را آتاماتای یادگیر با ساختار ثابت، و در غیر این‌صورت آن را آتاماتای یادگیر با ساختار متغیر می‌نامند.

۴- الگوریتم جدید ترکیبی برای حل مساله بهینه سازی

در الگوریتم ارائه شده در هر بعد از یک آتاماتای یادگیر استفاده شده است. هر آتاماتا دارای n_x اقدام است که می‌تواند یکی از m مقدار قابل قبول را به خود اختصاص دهد. در واقع هر اقدام آتاماتا معادل یک بازیکن می‌باشد که می‌تواند استراتژی خود را از میان مجموعه [0,m-1] انتخاب کند. به عبارت دیگر می‌توان گفت: اقدام $LA_1.\alpha_1$ (اقدام اول آتاماتای شماره ۱) معادل استراتژی انتخاب شده توسط بازیکن شماره ۱ برای بیت اول بعد x_1 است. لذا می‌توان گفت که در یک فضای جستجوی n بعدی، مجموعه بازیکنان برابر با $I=1,2,...,n*n_x$ خواهد بود و استراتژی خالص[1] هر بازیکن برابر با $s_i\in\{0,1,...,m-1\}$ می‌باشد. همچنین عبارت $(LA_1.\alpha_1,...,LA_1.\alpha_{n_x})$ پروفایل استراتژی بازیکنان بعد x_1 را نشان می‌دهد. لذا پروفایل استراتژی همه بازیکنان با عبارت زیر نمایش داده می‌شود.

$$([LA_1.\alpha_1,...,LA_1.\alpha_{n_x}),...,(LA_n.\alpha_1,...,LA_n.\alpha_{n_x}))$$

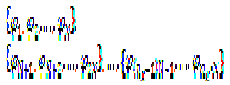
می‌توان از تابع هدف به عنوان تابع بهره[2] استفاده کرد. به عبارت دیگر

$$\mu((LA_1.\alpha_1,...,LA_1.\alpha_{n_x}),...,(LA_n.\alpha_1,...,LA_n.\alpha_{n_x}))\equiv f(x_1,...,x_n)$$

یا به طور خلاصه خواهیم داشت:

$$\mu(LA_1,...,LA_n)\equiv f(x)$$

$\varphi=\{\varphi_1,\varphi_2,...,\varphi_{n_x*N}\}$ مجموعه وضعیت‌ها و N عمق حافظه برای آتاماتا می‌باشد. مجموعه وضعیت‌های این آتاماتا به n_x (تعداد اقدامها یا تعداد بازیکنان در هر بعد) زیر مجموعه زیر افراز می‌شود.



به عبارت دیگر هر اقدام(بازیکن) N وضعیت داخلی دارد. در مجموعه وضعیت‌های اقدام(بازیکن) i، وضعیت $\Phi_{(j-1)N+1}$ را وضعیت داخلی و وضعیت Φ_{jN} را وضعیت مرزی می‌نامیم. اقدامی(بازیکنی) که در وضعیت $\Phi_{(j-1)N+1}$ قرار دارد نشان دهنده بیش‌ترین اهمیت برای استراتژی است، که انتخاب کرده است. و اقدامی(بازیکنی) که در وضعیت Φ_{jN} قرار دارد دارای کمترین اهمیت برای استراتژی اختصاص یافته به آن است.

$\beta = \{0,1\}$ مجموعه ورودی‌های آتاماتا می‌باشد. در این مجموعه 1 شکست و 0 موفقیت را نشان می‌دهد.

با استفاده از تابع $F : \Phi \times \beta \to \Phi$ میزان اهمیت مقدار اختصاص یافته به هر اقدام(بازیکن) تغییر می‌کند. این تابع از روی وضعیت فعلی و ورودی آتاماتا، وضعیت بعدی آن را تولید می‌نماید. این تابع برای آتاماتاهای مختلف، متفاوت می‌باشد.

$G : \Phi \to \alpha$ تابع نگاشت خروجی می‌باشد. با کمک این تابع استراتژی که برای یک بازیکن در نظر گرفته شده است به عنوان مقدار مناسب بر ای بیت مربوطه در x_i قرار می‌گیرد. به عنوان مثال اگر رابطه ۲ را در نظر بگیریم دو آتاماتای LA_1, LA_2 متناظر با x_1, x_2 وجود خواهد داشت که هر آتاماتا دارای ۱۰ اقدام(بازیکن) است و هر بازیکن می‌تواند یکی از دو مقدار مجموعه {0,1} را به عنوان استراتژی خود انتخاب کند. در الگوریتم ارائه شده ابتدا همه بازیکنان به صورت تصادفی یک استراتژی را انتخاب می‌کنند. سپس در هر تکرار از الگوریتم تکاملی ابتدا با احتمال p_d استراتژی انتخاب شده هر بازیکن تغییر می‌یابد و استراتژی جدیدی از مجموعه [0,m-1]، برای آن بازیکن در نظر گرفته می‌شود. این عمل باعث می‌شود تا از قرار گرفتن در مینیمم‌های محلی اجتناب گردد. سپس در فاز بعد به بررسی اقدام‌های آتاماتاها پرداخته می‌شود. در این مرحله ابتدا از روی هر آتاماتا یک آتاماتای دیگری ایجاد می‌شود که با احتمال p_d در یک اقدام(بازیکن) دارای استراتژی مشابه با والد خود بوده و با احتمال $1-p_d$ در آن استراتژی متفاوت است. در صورتی که استراتژی انتخاب شده برای هر اقدام(بازیکن) بهترین مقدار ممکن باشد (حداکثر افزایش را در تابع هدف داشته باشد) به آن اقدام(بازیکن) پاداش داده می‌شود و در غیر اینصورت آن اقدام(بازیکن) جریمه می‌شود. نهایتا بایستی میزان تاثیر آتاماتای جدید را در کل مجموعه محاسبه شود. اگر وجود آتاماتای جدید به همراه سایر آتاماتاها منجر به افزایش تابع هدف شود، آتاماتای جدید را جایگزین آتاماتای والد می‌کنیم و در غیر این صورت از آتاماتای جدید صرف نظر خواهیم کرد و از همان آتاماتای قبلی جهت ادامه کار استفاده میکنیم. بررسی اقدام‌های آتاماتاها را تا تعداد تکرار مشخص (R) یا عدم بهبود تابع هدف در چند تکرار متوالی (d_r) ادامه داده می‌شود. جزئیات این روش در شکل ۶ نشان داده شده است.

```
procedure LAEGA
  t=0;
  Randomly initialize actions of LA_1^(t), LA_2^(t), ..., LA_n^(t)
  From [0,m-1] as current solution;
  while termination condition is not satisfied do
    for each LA_i, i = 1,2,...,n do
      for each action j of LA_i, j = 1,2,...,n_a do
        if rand(0,1) < p_d then
          LA_i.a_j = randint[0,m − 1]
        end if
      end for
    end for
    r = 0;
    while r < R or function value not improved for d_r rounds do
      for each LA_i, i = 1,2,...,n do
        for each action j of LA_i, j = 1,2,...,n_a do
          if rand(0,1) < p_d then
            LA_i^'.a_j = randint[0,m − 1]
          else
            LA_i^'.a_j = LA_i.a_j
          end if
        end for
        for each action j of LA_i, j = 1,2,...,n_a do
          a = choose one of [0,m-1] for action j
          that maximize function value;
          if LA_i^'.a_j == a then
            Reward (LA_i^', a_j);
          else
            Penalize (LA_i^', a_j);
          end if
        end for
        if f(LA_1, ..., LA_i^', ..., LA_n) > f(LA_1, ..., LA_i, ..., LA_n) then
          LA_i = LA_i^';
        end if
      end for
      r = r + 1;
    end while
    if f(LA_1, ..., LA_n) > f(LA_1^(t), ..., LA_n^(t)) then
      for each LA_i, i = 1,2,...,n do
        LA_i^(t+1) = LA_i;
      end for
    else
      for each LA_i, i = 1,2,...,n do
        LA_i^(t+1) = LA_i^(t);
      end for
    end if
    t = t + 1;
  end while
end procedure
```

شکل ۶- الگوریتم ترکیبی

همچنین جزئیات مربوط به نحوه جریمه و پاداش اقدام آتاماتا در شکل ۷ و شکل ۸ نمایش داده شده است.

```
procedure Reward(LA, u)
  if (LA.state(u)-1) mod N <> 0 then
    Dec(LA.state(u));
  end if
end procedure
```

شکل ۷-عملگر پاداش یک اقدام

```
Procedure Penalize(LA, u)
  if (LA.state(u)) mod N <> 0 then
    Inc(LA.state(u));
  else
    a = choose one of [0,m-1] for action u
    that maximize function value;
    LA.u = a;
  end if
end procedure
```

شکل ۸- عملگر جریمه یک اقدام

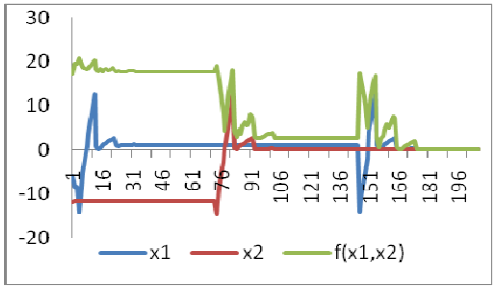
۵- نتایج آزمایشات

برای بررسی عملکرد روش ارائه شده از سه تابع استاندارد Griewangk و Ackley و Rastrigin استفاده شده است. در آزمایشات انجام شده، شرط خاتمه الگوریتم تکاملی، تعداد تکرار ۵۰۰ دور (T=500) یا رسیدن به خطای ۰،۰۰۰۱ در نظر گرفته شده است. همچنین احتمال تغییر مقدار اختصاصی هر اقدام را ۰،۷ ($p_c = 0.7$) و تعداد اقدامها را ۷ ($n_x = 7$) و تعداد مقادیر قابل انتخاب برای هر اقدام ۲ ($m = 2$) در نظر گرفته شده است. نتایج نشان داده شده، میانگین ۱۰ بار اجرای هر آزمایش است. نتایج حاصل با الگوریتم BGA[1] یک الگوریتم ژنتیکی است مقایسه شده است.

همانگونه که از جدول ۱ مشاهده می‌شود، در همه توابع الگوریتم LAEA با تعداد فراخوانی‌های کمتری از توابع مذکور به شرط خاتمه رسیده است.

	Function evaluation					
	Ackley		Griewangk		Rastrigin	
n	LAEGA	BGA	LAEGA	BGA	LAEGA	BGA
100	19178	53860	6974	361722	13566	25040
200	38348	107800	27756	748300	27906	52948
400	76498	220820	55766	1630000	55776	112634

جدول ۱- مقایسه تعداد فراخوانی‌های تابع در الگوریتم LAEGA و الگوریتم BGA



شکل ۹- فرآیند نزدیک شدن به نقطه می نیمم سراسری

شکل ۹ نحوه نزدیک شدن به نقطه می‌نیمم سراسری را نشان می‌دهد. نمودار آبی رنگ تغییرات بعد x1 و نمودار قرمز رنگ تغییرات بعد x2 را نشان می‌دهد. این نمودار از آزمایش تابع Ackely دو بعدی حاصل شده است. همانگونه که مشخص است در ۷۰ فراخوانی اول، ابتدا بعد x1 به می نیمم محلی نزدیک می‌شود و در ۷۰ فراخوانی بعدی بعد x2 به می‌نیمم سراسری نزدیک شده است. و این فرآیند تا رسیدن به می‌نیمم سراسری تکرار شده است. نمودار سبزرنگ تغییرات $f(x_1,x_2)$ را نشان می دهد.

۶- نتیجه گیری

در این مقاله یک الگوریتم ترکیبی جدید مبتنی برآتاماتای یادگیر برای حل مساله بهینه سازی ارائه شد. در این روش فضای جستجوی مساله که یک فضای پیوسته است به فضایی گسسته نگاشت شد که در آن در هر بعد از فضای جستجوی گسسته از یک آتاماتای یادگیر که شاما تعدادی اقدام(بازیکن) است، جهت یافتن نقطه بهینه استفاده شد. استفاده از عملگرهای جریمه و پاداش و انتخاب همیشه‌ی بهترین استراتژی برای هر بازیکن در مقابل استراتژی سایر بازیکنان باعث افزایش کارایی روش ارائه شده است. از مهمترین پارامترهای این روش میزان گسستگی فضای جدید است. به نظر نویسنده می‌توان به عنوان پژوهش‌های بعدی، در مراحل ابتدایی الگوریتم فضای گسسته را با دقت کمتری ایجاد نمود تا الگوریتم با سرعت بیشتری به نقطه می-نیمم سراسری نزدیک شود و در ادامه با کوچک کردن ابعاد فضای جستجو به اطراف نقطه می‌نیمم سراسری و افزایش دقت فضای گسسته به جستجوی دقیق نقطه می‌نیمم سراسری پرداخت.

مراجع

[1] Thathachar, M.A.L. and Sastry, P.S., "Varieties of Learning Automata: An Overview", IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics, Vol. 32, No. 6, pp. 711-722, 2002.

[2] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization", IEEE Transactions on Evolutionary Computation, 2008.

[3] Potter, M. A., De Jong, K. A., "A Cooperative Coevolutionary Approach to Function Optimization". Proceedings of the International Conference on Evolutionary Computation, The 3rd Conference on Parallel Problem Solving from Nature, Springer Verlag, pp.249-257, 1994.

[4] H.Mühlenbein, D. Schlierkamp-Voosen, Predictive models for the Breeder genetic algorithm, Evolutionary Computation 1 (1) (1993) 25–49.

[5] Rezapoor Mirsaleh, M. and Meybodi, M. R., "A Hybrid Algorithm for Solving Graph Isomorphism Problem", Proceedings of the Second International Conference on Information and Knowledge Technology (IKT2005), Tehran, Iran, May 24-26, 2005.

پانویس‌ها

[1] Pure Strategy
[2] Utility Payoff