ارائه یک الگوریتم کلونی مورچه تطبیقی برای حل مسائل بهینه سازی پیوسته

کیان فر، سحر[۱]
میبدی، محمدرضا[۲]

چکیده : بسیاری از مسائل دنیای واقعی را می‌توان به شکل یک مسئله بهینه‌سازی پیوسته مدل کرد. برخی از این مسائل عبارتند از: بهینه سازی شبکه‌های عصبی، طراحی موشک، بیوتکنولوژی، کنترل، اقتصاد، مدیریت منابع. الگوریتم های ACO دسته ای ازالگوریتم های بهینه سازی هستند که از کاوش برخی گونه‌های مورچه‌ها برای غذا در طبیعت الهام گرفته شده اند. این مورچه‌ها دنباله فرمون بر روی زمین برجای می‌گذارند تا راه‌های مطلوبی را که می‌تواند توسط اعضای دیگر گروه دنبال شود علامت بزنند. بهینه‌سازی کلونی مورچه مکانیزم مشابهی را برای حل مسائل بهینه سازی به کار می‌گیرد. این دسته از الگوریتم ها با موفقیت در محدوده‌ی وسیعی از مسائل بهینه سازی به کار رفته اند. بیشترین محبوبیت ACO به دلیل موفقیت در حل مسائل بهینه سازی ترکیبی است. اما اخیراً نسخه هایی از ACO برای مواجه با مسائل بهینه سازی پیوسته توسعه داده شده اند. یکی از این الگوریتم ها که بیشترین تطابق را با ACO دارد، ACO_R نام دارد. این الگوریتم دارای پارامترهایی مثل اندازه آرشیو، q ونرخ تبخیر فرمون است که همگرایی و کارآیی الگوریتم به آن‌ها وابسته است. در این مقاله روشی مبتنی بر اتوماتای یادگیر برای تطبیق این پارامترها معرفی می‌گردد و با انجام آزمایشات کارآیی روش معرفی شده مورد بررسی قرار می‌گیرد.

واژه های کلیدی: الگوریتم کلونی مورچه، اتوماتای یادگیر، مسائل بهینه‌سازی پیوسته

[۱] کارشناسی ارشد- دانشکده مهندسی کامپیوتر و فناوری اطلاعات- دانشگاه صنعتی امیرکبیر- kianfar@aut.ac.ir
[۲] استاد تمام- دانشکده مهندسی کامپیوتر و فناوری اطلاعات- دانشگاه صنعتی امیرکبیر- meybodi@ce.aut.ac.ir

۱ مقدمه

هوش جمعی رویکرد نسبتاً جدیدی در حل مسائل بهینه سازی است که از رفتار اجتماعی حشرات و حیوانات دیگر الهام می‌گیرد. از جمله مورچه‌ها که برای متدها و تکنیک های زیادی مورد الهام قرار گرفته‌اند و بیشترین موفقیت را در تکنیک بهینه سازی که به عنوان بهینه‌سازی‌کلونی مورچه[1](ACO)شناخته شده است[۱]، به دست آورده‌اند. ACO از رفتار جستجوی غذای مورچه‌ها الهام گرفته شده است. این الگوریتم ابتدا برای حل مسائل بهینه‌سازی گسسته مانند مسئله فروشنده‌دوره‌گرد[۱-۳]، مسیریابی [۴-۶] و زمانبندی[۷, ۸] معرفی شد. در این مسایل متغییر با مجموعه محدود از مولفه‌ها مشخص می‌شود. ACO موفقیت زیادی را در زمینه بهینه سازی گسسته به دست آورد. باوجود این‌که ACO ابتدا برای مسائل بهینه سازی گسسته پیشنهاد شد، استفاده از آن برای حل مسائل بهینه‌سازی پیوسته مورد توجه قرار گرفت[۸, ۹]. در این دسته از مسائل بهینه‌سازی متغییر یک مقدار حقیقی از دامنه تعریف شده را می‌گیرد. اخیراً الگوریتم ACO_R [۱۰] برای حل مسائل بهینه سازی پیوسته پیشنهاد شده است که ساختارش تفاوت چندانی با ساختار اصلی ACO ندارد. این الگوریتم دارای پارامترهایی است که کارآیی الگوریتم را تحت تاثیر قرار می‌دهند. از جمله‌ی این پارامترها اندازه آرشیو راه حل(k)، نرخ همگرایی و q هستند. مقدار این پارامترها با توجه به نوع مسئله از طریق آزمایش و خطا تعیین می‌شود. در این مقاله سعی داریم که مقدار این پارامترها را با استفاده از اتوماتای یادگیر تطبیق دهیم. در[۱۱]مطالعاتی درباره اثرات پارامترهای الگوریتم انجام گرفته است. در این مقاله با انجام آزمایش‌های مختلف نشان می‌دهیم که پارامترهای ACO_R در بازه‌های خاصی نتایج بهتری تولید می‌کنند. در ادامه با استفاده از اتوماتاهای یادگیر پارامترهای q و ξ الگوریتم را به صورت تطبیقی تعیین می‌کنیم. این دو پارامتر در بهنگام سازی فرمون (آرشیو) و تصمیم گیری مورچه‌ها دخیل هستند. برخلاف نسخه اصلی الگوریتم که تمام مورچه‌ها پارامترهای یکسان دارند در الگوریتم پیشنهادی هر مورچه پارامترهای خود را دارد. در الگوریتم پیشنهادی که آن را $Adaptive\text{-}ACO_R$ می نامیم، هر مورچه به ۲ اتوماتای یادگیر مجهز می‌شود که هر کدام وظیفه‌ی تطبیق یکی از پارامترها را بر عهده دارد و مورچه برای تصمیم‌گیری در انتخاب گره‌ی بعدی و بهنگام سازی فرمون ازآن‌ها استفاده می‌کند. تطبیق پارامتر k که برای همه مورچه‌ها مشترک است به صورت افزایشی انجام می‌گیرد. قبلاً اتوماتای یادگیر برای تطبیق پارامترهای ACO به کار رفته است[۱۲]. روش پیشنهادی با دیگر الگوریتم‌های کلونی مورچه مقایسه می‌شود و نتایج شبیه سازی برتری الگوریتم پیشنهادی را نشان می‌دهد.

ادامه مقاله به صورت ذیل سازماندهی شده است. در بخش دوم و سوم مقاله به ترتیب به معرفی الگوریتم کلونی مورچه در محیط گسسته و پیوسته می‌پردازیم. در بخش چهارم اتوماتاهای یادگیر را معرفی می‌کنیم. در بخش پنجم الگوریتم پیشنهادی را ارائه می‌دهیم و در بخش ششم نتایج شبیه سازی ارائه می‌گردد. در نهایت در بخش هفتم نتیجه گیری را ارائه می‌نمایم.

۲ الگوریتم کلونی مورچه

الگوریتم کلونی مورچه‌ها برای اولین بار توسط دوریگو [2] به عنوان یک راه حل چند عامله[3] برای حل مسائل مشکل بهینه سازی مانند مسئله فروشنده دوره گرد (TSP)[4] ارائه شد[۱۳]. این الگوریتم از رفتار جستجوی غذا برخی گونه‌های مورچه در طبیعت الهام گرفته شده است. زمان جستجوی غذا ابتدا مورچه‌ها نواحی اطراف لانه را با یک رفتار تصادفی کاوش می‌کنند، به محض اینکه مورچه‌ای منبع غذایی را پیدا کرد آن را ارزیابی می‌کند و مقداری از آن را به سمت لانه برمی‌گرداند. در مسیر برگشت، مورچه دنباله‌ای از فرمون را بر روی زمین بر جای می‌گذارد. این فرمون برجای مانده که مقدار آن به کیفیت و اندازه منبع غذایی بستگی دارد، مورچه‌های دیگر را به سمت غذا هدایت می‌کند. ارتباط غیرمستقیم بین مورچه‌ها یعنی استفاده از دنباله فرمون، آن‌ها را قادر به پیدا کردن کوتاهترین مسیر بین لانه و غذا می‌کند.

شبه‌کد الگوریتم بهینه‌سازی کلونی مورچه[5] در شکل ۱ نشان داده شده است[۱۰]. پس از تعیین پارامترها و مقدار دهی اولیه فرمون، الگوریتم در یک حلقه اصلی تکرار می‌شود، در هر تکرار همه‌ی مورچه‌ها راه‌حل‌ها را می‌سازند، این راه حل ها در یک جستجوی محلی تقویت می‌شوند(این گزینه‌ها اختیاری است) و درنهایت مقدار فرمون بروزرسانی می‌شود.

---

[1] Ant Colony Optimization

[2] Dorigo

[3] Multi Agent

[4] Traveling Sales Person

[5] Ant Colony Optimization

```
Ant Colony Optimization metaheuristic
while termination conditions not met do
    ScheduleActivities
        AntBasedSolutionConstruction( )
        PheromoneUpdate( )
        DaemonActions( ) {optional}
    end ScheduleActivities
endwhile
```

شکل ۱. شبه کد الگوریتم کلونی مورچه

ساخت راه حل: در الگوریتم بهینه‌سازی، یک مورچه‌ی مصنوعی یک راه حل را با پیمودن ساختار گراف کاملاً متصل $G_c(I$ می‌سازد که Vمجموعه‌ی راس‌ها و E مجموعه‌ی یال‌ها است. این گراف از مجموعه‌ی مولفه‌های راه حل C به دست می‌آید. مورچه‌های مصنوعی از رأسی به رأس دیگر در امتداد یال‌ها حرکت می‌کنند، تا به تدریج یک راه‌حل جزئی را بسازند. بعلاوه، یک مقدار مشخصی از فرمون بر روی مولفه‌ها برجای می‌گذارند. که با توجه به نحوه‌ی پیمایش گراف می‌تواند برروی رأس ها یا یال های گراف باشد. در مرحله ساخت راه‌حل توسط مورچه به صورت جزئی، هر حرکت از مولفه i به j با استفاده از یک رویکرد احتمالاتی انجام میگیرد. این احتمال با استفاده از رابطه ۱ بیان می‌شود.

$$p(c_{ij}|s^p) = \frac{\tau_{ij}^{\alpha}.\eta(c_{ij})^{\beta}}{\sum_{c_{il}\in N(s^p)} \tau_{il}^{\alpha}.\eta(c_{il})^{\beta}}, \forall c_{ij} \in N_i^k \qquad (۱)$$

که در آن تابع وزنی است که در هر مرحله از ساخت، یک مقدار هیوریستیک به هر یک از مولفه‌های راه‌حل موجود $\forall c_{ij} \in$ تخصیص می‌دهد. و پارامترهای مثبتی هستند که اهمیت اطلاعات هیوریستیک و فرمون را مشخص می‌کنند. یک مجموعه محدود از عناصر است که مورچه k هنوز آن‌ها را انتخاب نکرده و مقدار فرمون تخصیص یافته به مولفه است.

اعمال جستجوی محلی[1]: زمانی که راه‌حل ها ساخته شدند، قبل از به روزرسانی فرمون، راه حل های به دست آمده توسط مورچه‌ها تقویت می‌شود این فاز به مشخصات مسئله وابسته است و اختیاری است گرچه الگوریتم‌های بهینه سازی کلونی مورچه جدید شامل این فاز هستند.

به روزرسانی فرمون[2] : هدف از به روزرسانی فرمون افزایش فرمون راه حل‌های خوب و امیدبخش و کاهش فرمون راه‌حل های بد است. معمولاً این هدف با ۱) کاهش مقادیر فرمون با تبخیر و ۲) افزایش سطح فرمون یک مجموعه از راه حل‌های خوب قابل دستیابی است. (رابطه۲)

$$\tau_{ij}^{t+1} \leftarrow \begin{cases} (1-\rho).\tau_{ij}^{t} + \Delta\tau_{ij}^{t,k} & \text{if } \tau_{ij} \in \text{good solution} \\ (1-\rho).\tau_{ij}^{t} & \text{otherwise} \end{cases} \qquad (۲)$$

در این رابطه $0 < \rho$ ، نرخ تبخیر است، این پارامتر برای اجتناب از همگرایی سریع به کار می‌رود و الگوریتم را قادر به فراموش کردن تصمیمات بد قبلی می‌کند. مقدار فرمون برجای گذاشته شده توسط مورچه k روی عنصر ij است و به کیفیت راه‌حل پیدا شده بستگی دارد.

۳ الگوریتم کلونی مورچه برای مسائل بهینه سازی پیوسته

ACO_R یک الگوریتم بهینه سازی مبتنی بر کلونی مورچه است که توسط سوچا[3] و دوریگو در سال ۲۰۰۶ پیشنهاد شده است [۱۰]. ACO_Rسعی دارد متاهیوریستیک ACO را دنبال کند. ساختار آن به کاربر اجازه حل مسائل بهینه‌سازی ترکیبی(گسسته-پیوسته) را می‌دهد. قانون ACO_R در انتخاب مقادیر متغییر در فضای جستجو با توزیع احتمال پیوسته بیان می‌شود. همانطور که در بخش ۲ اشاره شد ACO در هر مرحله از ساخت یک انتخاب احتمالاتی مطابق (۱ انجام می‌دهد. احتمال تخصیص یافته، یک توزیع احتمال گسسته را می‌سازد. ایده‌ی اصلی ACO_R استفاده از یک توزیع احتمال پیوسته با استفاده از یک تابع چگالی احتمال (PDF)[4]، به جای استفاده از یک توزیع احتمال گسسته است. برای این منظور، الگوریتم تابع گاوسی را به کار می‌گیرد. اما یک تابع گاوسی به تنهایی نمی‌تواند دو نقطه از فضا را که کاندید بهینه هستند نشان دهد، بنابراین مولفین یک کرنل گاوسی را که مجموع وزندار از چندین تابع گاوسی واحد g می‌باشد، را تعریف می‌کنند و آن را با (x) نشان می دهند [۱۰]

[1] ApplyLocalSearch

[2] UpdatePheromones

[3] Socha

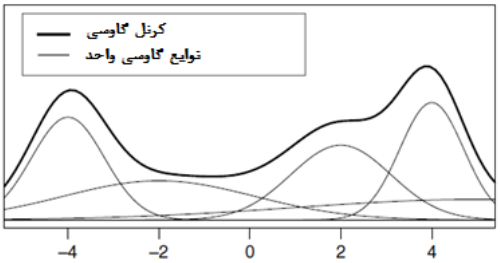
[4] Probability Density Function

$$G^l(x) = \sum_{l=1}^{k} w_l g_l^l(x) = \sum_{l=1}^{k} w_l \frac{1}{\sigma_l^l\sqrt{2\pi}} e^{-\frac{(x-\mu_l^l)^2}{2\sigma_l^{l^2}}}$$

(۳)

در این رابطه w بردار وزن توابع گاوسی، بردار میانگین و واریانس تابع گاوسی lام هستند.

چنین PDF ای ضمن این که امکان نمونه برداری ساده را فراهم می آورد، در مقایسه با یک تابع گاوسی واحد انعطاف پذیری بیشتری دارد. شکل ۲ یک نمونه تابع چگالی احتمال کرنل گاوسی را به همراه توابع گاوسی واحد نشان می‌دهد.



شکل ۲. نمونه ای از توابع واحد گاوسی و تابع کرنل گاوسی

راه حل‌ها در ACO_R در آرشیو ذخیره می‌شوند. در شروع الگوریتم آرشیو، با تولید k راه حل تصادفی مقدار دهی می‌شود. در آرشیو k راه‌حل $\{s_1, s_2, \dots, s_l, \dots\}$ به همراه مقادیر تابع هدف‌شان $\{f(s_1), f(s_2), \dots, f\}$ نگهداری می‌شوند. iامین متغییر از lامین راه حل را با نشان می‌دهیم. راه حل‌ها در آرشیو به ترتیب کیفیت‌شان نگهداری می‌شوند، بنابراین برای یک مسئله کمینه سازی داریم:

$$f(s_1) \le f(s_2) \le \dots \le f(s_l) \le \dots \le f(s_k)$$

و هر راه حل وزنی متناسب با کیفیتش دارد یعنی:

$$w_1 \ge w_2 \ge \dots \ge w_l \ge \dots \ge w_k$$

وزن راه حل با استفاده از (۴ به دست می‌آید.

$$w_l = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}}$$

(۴)

مطابق رابطه ۴ وزن مقدار تابع گاوسی با میانگین ۱و واریانسqk است و q پارامتر الگوریتم است. زمانی که q کوچک است راه حل-هایی با رتبه بالاتر ترجیح بیشتری دارند.

بردار دوم از کرنل گاوسی که باید تعیین شود، بردار میانگین است. برای هر G مقادیر iامین متغیر از راه حل‌های موجود در آرشیو به عنوان بردار میانگین انتخاب می‌شوند.

$$\mu^l = \{\mu_1^l, \dots, \mu_k^l\} = \{s_1^l, \dots, s_k^l\}$$

(۵)

برای تعیین بردار سوم یعنی واریانس ، فرض می‌کنیم که یک مورچه یکی از راه حلهای موجود در آرشیو را بر اساس یک متد احتمالی مانند چرخه رولت[۱۴] انتخاب می‌کند، بنابراین راه حلی که رتبه بالاتری دارد شانس بیشتری برای انتخاب شدن توسط مورچه دارد. پس واریانس بین iامین مقادیر متغییر انتخابی متناسب با متغییر محاسبه خواهد شد. از این روکل بردار واریانس محاسبه نمی‌شود. بلکه فقط که مورد نیاز است محاسبه می‌شود. به منظور محاسبه‌ی میانگین فاصله‌ی راه حل انتخابی تا دیگر راه حل های موجود در آرشیو را به دست می‌آوریم و آن را در پارامتر ضرب می کنیم.

$$\sigma_l^l = \xi \sum_{e=1}^{k} \frac{|s_e^l - s_l^l|}{k-1}$$

(۶)

پارامتر ξ برای همه‌ی ابعاد یکسان است وتاثیری مشابه نرخ تبخیر فرمون درACO دارد. مقادیر بالاتر سرعت همگرایی پایین الگوریتم را به همراه دارد.

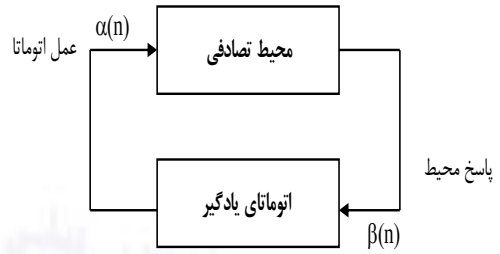
برای هر مورچه، با استفاده از یک تولید کننده تصادفی نرمال مانند متد Box-Muller [۱۵]با میانگین و انحراف معیار مقدار iام متغییر تصمیم مشخص خواهد شد، این پروسه تا انتخاب nامین متغییر تصمیم ادامه خواهد داشت و در انتها مقدار تابع هدف محاسبه می‌شود. سرانجام به تعداد مورچه ها راه حل جدید ساخته می‌شود و به آرشیو راه حل اضافه می‌شود. در مرحله به روز رسانی فرمون kراه حل در آرشیو باقی می‌مانند و مابقی حذف می-شوند

۴ اتوماتای یادگیر

اتوماتای یادگیر[۱][۱۶]، ماشینی است که می‌تواند تعدادی متناهی عمل را انجام دهد. هر عمل انتخاب شده توسط یک محیط احتمالی ارزیابی می‌شود و نتیجه ارزیابی در قالب سیگنالی مثبت یا منفی به اتوماتا داده می‌شود و اتوماتا از این پاسخ در انتخاب عمل بعدی تاثیر می‌گیرد. هدف نهایی این است که اتوماتا یاد بگیرد تا از بین اعمال خود بهترین عمل را انتخاب کند. بهترین عمل، عملی

4 Learning Automata

است که احتمال دریافت پاداش از محیط را به حداکثر برساند. کارکرد اتوماتای یادگیر در تعامل با محیط، در شکل ۳مشاهده می‌شود.



شکل ۳. ارتباط بین اتوماتای یادگیر و محیط

محیط را می‌توان توسط سه تایی $E \equiv \{\alpha, \beta, c\}$ نشان داد که در آن $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه ورودی ها، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_m\}$ مجموعه خروجی‌ها و $c \equiv \{c_1, c_2, ..., c_r\}$ مجموعه احتمال‌های جریمه می‌باشد. هرگاه β مجموعه دو عضوی باشد، محیط از نوع P می‌باشد. در چنین محیطی $\beta_1 = 1$ به عنوان جریمه و $\beta_2 = 0$ به عنوان پاداش در نظر گرفته می‌شود. در محیط از نوع Q، $\beta(n)$ می‌تواند به طور گسسته یک مقدار از مقادیر محدود در فاصله [۰,۱] و در محیط از نوع S، $\beta(n)$ متغیر تصادفی در فاصله [۰,۱] است. c_i احتمال اینکه عمل α_i نتیجه نامطلوب[۱] داشته باشد، می‌باشد. در محیط ایستا[۲] مقادیر c_i بدون تغییر می‌مانند، حال آنکه در محیط غیر ایستا[۳] این مقادیر در طی زمان تغییر می‌کنند.

اتوماتای یادگیر با ساختار ثابت توسط پنج-تایی $\{\alpha, \beta, F, G, \phi\}$ نشان داده می‌شود که در آن $\alpha \equiv \{\alpha_1, \alpha_2, \cdots, \alpha_r\}$ مجموعه عمل‌های اتوماتا، $\beta \equiv \{\beta_1, \beta_2, \cdots, \beta_m\}$ مجموعه ورودی‌های اتوماتا، $\phi \equiv \{\phi_1, \phi_2, \cdots, \phi_s\}$ مجموعه وضعیت‌های داخلی اتوماتا، $F : \phi \times \beta \to \phi$ تابع تولید وضعیت جدید اتوماتا و $G : \phi \to \alpha$ تابع خروجی می‌باشد که وضعیت کنونی اتوماتا را به خروجی بعدی می‌نگارد.

اتوماتای یادگیر با ساختار متغیر را میتوان توسط چهارتایی $\{\alpha, \beta, p, T\}$ نشان داد که $\alpha = \{\alpha_1, ..., \alpha_r\}$ مجموعه عملهای اتوماتا، $\beta = \{\beta_1, ..., \beta_m\}$ مجموعه ورودیهای اتوماتا، $p = \{p_1, ..., p_r\}$ بردار احتمال انتخاب هریک از عملها و $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ الگوریتم یادگیری می باشد. الگوریتم زیریک نمونه از الگوریتمهای یادگیری خطی است.

فرض کنید عمل α_i در مرحله nام انتخاب شود.

- پاسخ مطلوب

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)]$$
$$p_j(n+1) = (1-a)p_j(n) \qquad \forall j \quad j \neq i$$

(۷)

- پاسخ نامطلوب

$$p_i(n+1) = (1-b)p_i(n)$$
$$p_j(n+1) = (b/r - 1) + (1-b)p_j(n) \quad \forall j \quad j \neq i$$

(۸)

در روابط ۷و ۸ ، a پارامتر پاداش و b پارامتر جریمه می باشند. با توجه به مقادیر a و b سه حالت زیر را می توان در نظر گرفت. زمانی‌که a و b با هم برابر باشند، الگوریتم را L_{RP} می نامیم، زمانیکه b از a خیلی کوچکتر باشد، الگوریتم را $L_{R\varepsilon P}$ می نامیم. و زمانیکه b مساوی صفر باشد الگوریتم را L_{RI} مینامیم[۱۷].

۵ الگوریتم پیشنهادی

همانطور که ذکر شد الگوریتم ACO_R دارای پارامترهای اندازه آرشیو راه حل(k)، نرخ همگرایی و q است که کارآیی الگوریتم را تحت تاثیر قرار می‌دهند. در مقالات مقدار این پارامترها با توجه به نوع مسئله از طریق آزمایش و خطا تعیین می‌شود. در این بخش مقدار این پارامترها را با استفاده از اتوماتای یادگیر تطبیق دهیم. این بخش شامل دو قسمت است در قسمت اول با استفاده از اتوماتاهای یادگیر پارامترهای q و ξ الگوریتم را به صورت تطبیقی تعیین می‌کنیم. هر مورچه به ۲ اتوماتای یادگیر مجهز می‌شود که هر کدام وظیفه‌ی تطبیق یکی از پارامترها را بر عهده دارد و مورچه برای تصمیم‌گیری در انتخاب گره‌ی بعدی و بهنگام سازی فرمون ازآن‌ها استفاده می‌کنند. در قسمت دوم برای تطبیق پارامتر k که مشترک بین تمام مورچه‌هاست یک روش افزایشی معرفی می‌کنیم.

---

5 Unfavorable
6 Stationary
7 Non-Stationary

k بزرگ تنوع را در فضای جستجو بالا می‌برد در حالی که مقدار کوچک آن راه حل‌های موجود را تقویت می‌کند.

۵-۱- تطبیق پارامترهای ACO_R با استفاده از اتوماتای یادگیر

در این بخش متدی مبتنی بر اتوماتای یادگیر برای تطبیق پارامترهای q و ξ الگوریتم ACO_R ارایه می‌گردد. این دو پارامتر در بهنگام سازی فرمون (آرشیو) و تصمیم گیری مورچه‌ها دخیل هستند. در رابطه(۴) ترم qk واریانس توزیع گاوسی است که برای محاسبه وزن راه‌حل موجود در آرشیو به کار می‌رود. زمانی که q کوچک است راه‌حل‌هایی با بهترین رتبه قویاً ترجیح داده می‌شوند و زمانی که q بزرگ است احتمال به طور یکنواخت توزیع می‌شود. تاثیر این پارامتر در ACO_R مشابه متعادل کردن توازن بین به‌روز رسانی فرمون با بهترین در تکرار[1] و بهترین تاکنون[2] به کاررفته در ACO است. پارامتر ξ برای همه‌ی ابعاد یکسان است وتاثیری مشابه نرخ تبخیر فرمون در ACO دارد(رابطه ۶). مقادیر بالاتر سرعت همگرایی پایین الگوریتم را به همراه دارد. برخلاف نسخه اصلی الگوریتم که تمام مورچه‌ها پارامترهای یکسان دارند در الگوریتم پیشنهادی هر مورچه پارامترهای خود را دارد.

در الگوریتم پیشنهادی که آن را $Adaptive\text{-}ACO_R$ می نامیم هر مورچه به ۲ اتوماتای یادگیر مجهز شده است که هر کدام وظیفه‌ی تطبیق یکی از پارامترها را بر عهده دارد و مورچه برای تصمیم‌گیری در انتخاب گره‌ی بعدی و بهنگام سازی فرمون ازآن‌ها استفاده می‌کند. با توجه به نتایج ارائه شده در [۱۰] برای هر پارامتر مجموعه‌ای از اعداد تعریف می‌کنیم که پارامتر می‌تواند اختیار کند:

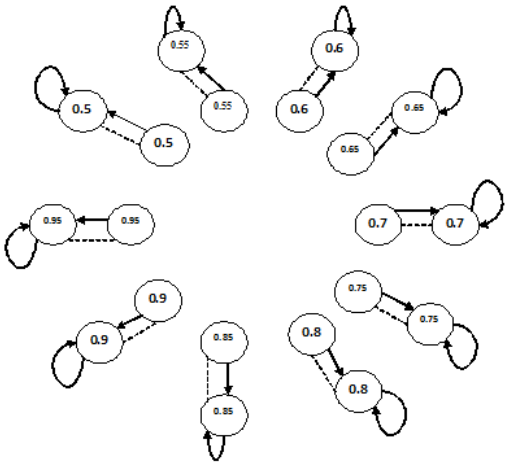
$$q \in \{10e-7, 10e-6, 10e-5, 10e-4, 10e-3, 10e-2, 10e-1, 1\}$$

$$\xi \epsilon \{0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95\}$$

هر عمل در اتوماتای یادگیر متناظر با یک مقدار در مجموعه‌ی تعریف شده‌است. اتوماتای یادگیری که q را تطبیق می‌دهد دارای ۸ عمل و اتوماتای یادگیر ξ دارای ۱۰ عمل است. که هر عمل معرف یکی از مقادیر موجود در مجموعه است. شکل ۴و ۶ اتوماتای تطبیق ξ را نشان می‌دهند. شکل۵ تغییر حالت اتوماتای یادگیر زمانی که اتوماتا پاداش می‌گیرد و شکل ۶ تغییر حالات اتوماتای یادگیر زمانی که اتوماتا جریمه می‌شود را نشان می‌دهند. اتوماتای یادگیر برای q مشابه این اتوماتاست. کیفیت جواب به دست آمده در هر مرحله از الگوریتم برای دادن پاداش یا جریمه به اتوماتا به کار می‌رود. در واقع مسئله نقش محیط را برای اتوماتا بازی می‌کند. از این طریق هر مورچه در حین اجرای الگوریتم یاد می‌گیرد که بهترین پارامترها برای حل مسئله کدام است. در شروع الگوریتم پارامترهای مورچه بصورت تصادفی انتخاب می‌شوند.

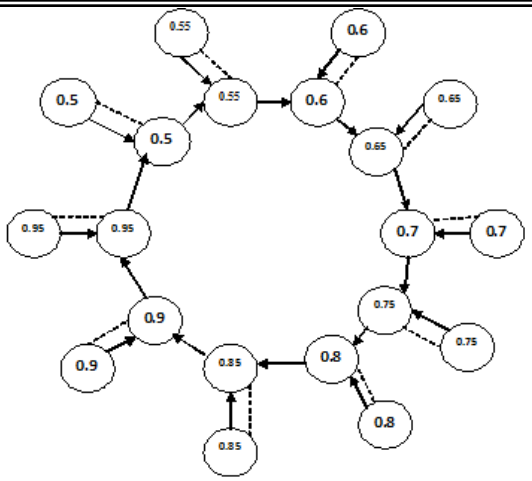
۵-۲- تطبیق افزایشی k

راه حل‌های موجود در آرشیو به عنوان راهنمایی برای تولید راه‌حل‌های جدید به کار می‌روند. در الگوریتم پیشنهادی سایز آرشیو در طول زمان افزایش پیدا می‌کند. این اصلاح بر اساس فریم ورک یادگیری اجتماعی افزایشی است[۱۸, ۱۹]. پارامتر Growth نرخ رشد آرشیو را کنترل می‌کند. نرخ بالاتر تنوع را در فضای جستجو بالا می‌برد در حالی که مقدار یک برای آن راه حل‌های موجود را تقویت می‌کند. پروسه‌ی بهینه‌سازی با یک آرشیو کوچک شروع می‌شود که سایز این آرشیو با پارامتر InitArchiveSize تعریف می‌شود. یک راه حل جدید در هر Growth تکرار یک بار به آرشیو افزوده می‌شود تا زمانی که به ماکزیمم اندازه آرشیو یا MaxArchiveSize برسیم. بعد از رسیدن به این سایز آرشیو مطابق با الگوریتم اصلی به روز رسانی می‌شود.



شکل ۴.عمل برای تطبیق پارامتر ξ،نمودار تغییر وضعیت اتوماتای یادگیر برای پاسخ مطلوب

[1] iteration-best
[2] best-so-far



شکل ۵ .عمل برای تطبیق پارامتر ξ،نمودار تغییر وضعیت اتوماتای یادگیر برای پاسخ نامطلوب

۵-۳- الگوریتم Adaptive-ACO_R

این الگوریتم مشابه ACO_R عمل می کند و از m مورچه برای ساخت راه حل استفاده می‌کند. هر مورچه به ۲ عدد اتوماتای یادگیر مجهز شده است. در شروع الگوریتم عمل انتخابی اتوماتا که همان پارامتر متناظر اتوماتای یادگیر است بصورت تصادفی انتخاب می‌شود. در هرتکرار بعد از این که مورچه ها با استفاده از پارامترهای خود راه حل را ساختند و مقادیر فرمون موجود در آرشیو ذخیره را با استفاده از مقادیر پارامترهای فعلی‌شان بهنگام نمودند، الگوریتم با توجه به نتایج تولید شده توسط مورچه‌ها، اقدام به تطبیق پارامترهای هر مورچه می‌نماید. اعمال اتوماتای یادگیر برای m/2مورچه‌ای که بهترین راه‌حل ها را یافته‌اند پاداش می-گیرند و اتوماتاهای یادگیرm/2 مورچه‌ای که بدترین راه‌حل ها را به یافته‌اند. جریمه می‌شوند.

۶ آزمایشات

آزمایشات بر روی ۴ تابع استاندارد با ویژگی های متفاوت انجام گرفته اند که معمولاً به عنوان معیار سنجش الگوریتم های بهینه سازی در فضای پیوسته و ایستا مورد استفاده قرار می‌گیرند. توابع بکار رفته به همراه دامنه آن‌ها در**جدول ۱** لیست شده اند. لازم به ذکر است که مقدار بهینه برای کلیه توابع صفر می باشد. الگوریتم در محیط MATLAB 7.6.0 پیاده‌سازی شده‌است. آزمایشات ۳۰ بار تکرار شده‌اند و متوسط نتایج و انحراف معیار آن‌ها گزارش شده است

آزمایش۱. بررسی تاثیر پارامترهای ACO_R

در این آزمایش الگوریتم ACO_R را به ازای مقادیر مختلف پارامترها اجرا می‌نمایم. شکل ۶. کارآیی الگوریتم **ACOR** برای تنظیمات مختلف پارامترهاکارآیی الگوریتم را با تنظیمات متفاوت پارامتر روی توابع تست ۱۰ بعدی نشان می‌دهند. کارآیی الگوریتم را با ۴ ترکیب متفاوت از پارامترهایq و k در الگوریتم ACO_R، $q \in (0.0001, 0.$ و $k \in (50, 1$ با مقادیر ثابت m=2 و ξ=0.85، به دست آورده‌ایم. تفاوت کارآیی با تنظیمات متفاوت در شکل بیانگر این است که یک متد تنظیم خودکار پارامتر به بهبود کارآیی الگوریتم کمک خواهد کرد.

آزمایش ۲. مقایسه کارآیی الگوریتم پیشنهادی با الگوریتم های دیگر

نتایج حاصل از الگوریتم پیشنهادی با الگوریتم ACO_R، و دو نسخه دیگر از الگوریتم کلونی مورچه پیوستهCIAC [۸] و DACO[۹]مقایسه شده است. برای الگوریتم ACO_R، q=10e-4، ξ=0.8، k=50 و m=2 است. پارامترهای الگوریتم CIAC و DACO طبق [۸, ۹] تنظیم شده‌اند. الگوریتم Adaptive-ACO_R را با پارامترهای Growth=4، MaxArchiveSize=100، InitArchiveSize=2 و اتواتای نوع L با عمق ۱۰ اجرا می‌کنیم. نتایج حاصل از اجرای الگوریتم‌ها در جدول ۳ نشان داده شده‌است نتایج موجود در جدول بیانگر میانگین و واریانس مقادیر بهینه‌ی تابع، حاصل از ۳۰ بار می‌باشند. نتایج نشان می‌دهد الگوریتم Adaptive-ACO_R در مقایسه با الگوریتم اصلی و الگوریتم‌های دیگر مورچه کارآیی بالاتری دارد. زیرا درAdaptive-ACO_R با انتخاب مقادیر مناسب برای پارامترها در تعداد تکرارهای مشابه به دقت بیشتری می‌رسیم.

آزمایش ۳. بررسی تاثیر و نوع اتوماتا در کارآیی الگوریتم Adaptive-ACO_R

در این آزمایش تاثیر پارامتر عمق در اتوماتاهای یادگیر و نوع الگوریتم یادگیری را بررسی می‌کنیم. برای این منظور یک بار Adaptive-ACO_R را با اتوماتاهای یادگیر ثابت که هر یک از اعمال آن‌ها دارای عمق ۳ می‌باشد و بار دیگر با اتوماتاهای ثابت که هریک از اعمال آن‌ها دارای عمق ۱۰ می‌باشد، بر روی دو تابع Sphere و Rosenbrock تست می‌کنیم. نتایج حاصل درجدول ۴ و جدول ۵ آمده است. این نتایج نشان می‌دهد که افزایش عمق اتوماتا تاثیری مثبت بر نتایج به دست آمده داشته است با این وجود تفاوت کم بین نتایج بیان‌گر این موضوع است که الگوریتم به این پارامتر و نوع الگوریتم یادگیری اتوماتا حساس نیست.

۷ نتیجه گیری

در این مقاله یک الگوریتم کلونی مورچه‌ی تطبیقی برای بهینه سازی توابع پیوسته ارائه دادیم. نتایج حاصل از اجرای الگوریتم بر روی مجموعه‌ای از توابع تست استاندارد نشان داد که این الگوریتم به خوبی در چنین مسائلی عمل می‌کند و می‌تواند نسبت به الگوریتم‌های کلونی مورچه دیگر کارآتر باشد. نقطه قوت این الگوریتم این است که در آن هر مورچه می‌تواند یک راه حل متفاوت از نواحی امید به جواب موجود در آرشیو انتخاب کند، بنابراین تنوع در طول اجرای الگوریتم حفظ میشود. همچنین در توابع چند قله‌ای مورچه‌ها با پارامترهای متفاوت می‌توانند قله‌های متفاوت را کاوش کنند.

نام تابع	تابع	محدودهٔ فضای جستجو
Sphere	$f_1(x)=\sum_{i=1}^{D} x_i^2$	$[-100,100]^D$
Rastrigin	$f_2(x)=\sum_{i=1}^{D}\left(x_i^2-10\cos(2\pi x_i)+10\right)$	$[-5.12,5.12]^D$
Rosenbrock	$f_3(x)=\sum_{i=1}^{D}\left(100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right)$	$[-50,50]^D$
Ackly	$f_6(x)=20+e-20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}}-e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$	$[-32,32]^D$

جدول ۱. توابع استاندارد بکار رفته به همراه محدوده فضای جستجو

	q=10e-4,k=100	q=10e-4,k=50	q=10e-2,k=100	q=10e-2,k=50
Sphere	0.8147 ±(0.26)	**0.15932 ±(0.038)**	0.258±(0.052)	3.224 ±(2.76)
Rosenbrock	6.08E-05±(5.80E-05)	**4.53E-05±(3.94E-05)**	8.37E-05±(8.44E-05)	5.22E-05±(5.49E-05)
Ackly	1.351376±(0.2)	1.777±(3.26)	1.5711±(0.2618)	**1.34±(0.22)**
Rastergin	41.814±(7.37)	**40.93±(8.65)**	54.56 ±(7.56)	50.097±(7.19)

جدول ۲. کارآیی الگوریتم ACO_R برای تنظیمات مختلف پارامترها

	Sphere	Rosenbrock	Ackly	Rastergin
Adaptive-ACO_R	**0.0662 ± 0.0694**	0.00966±0.01638	**1.289±8.5569**	**36.823±29.066**
ACO_R	0.1593±0.0388	**4.53E-05±3.94E-05**	1.778±3.262	40.939±8.651
DASO	0.0876±0.01	5.2650±1.9225	18.722± 0.857	49.449±21.502
CIAC	15.764±2.697	21.485±12.933	9.155±3.114	107.166±12.187

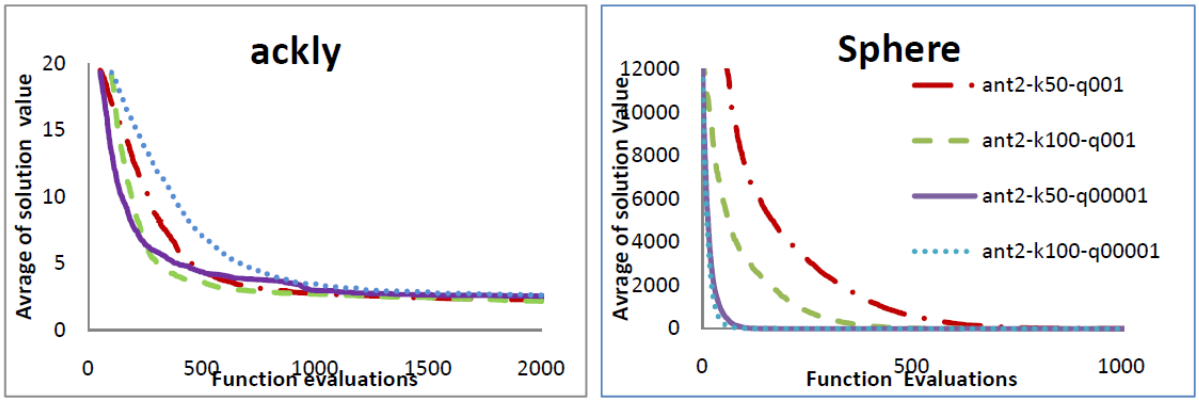
جدول ۳. مقایسه کارآیی الگوریتم پیشنهادی با الگوریتم‌های دیگر

sphere	Depth=3	Depth=10
Automata Type	Mean	Mean
L	0.101±0.1279	0.0662 ±0.069
G	0.115±0.1527	0.0636±0.079
Krinskry	0.115±0.1412	0.0985±0.112
Krylov	0.095±0.1046	0.1137±0.1863

جدول ۴.بررسی تاثیر پارامترهای اتوماتا روی تابع sphere

rosenbrock	Depth=3	Depth=10
Automata Type	Mean	Mean
L	0.0068±0.0126	0.0096±0.0163
G	0.0101±0.0192	0.0109±0.0408
Krinskry	0.0125±0.0475	0.0160±0.0466
Krylov	0.0288±0.0574	0.0324±0.1534

جدول ۵. بررسی تاثیر پارامترهای اتوماتا روی تابع rosenbrock



شکل ۶. کارآیی الگوریتم ACOR برای تنظیمات مختلف پارامترها

مراجع

1. Marco Dorigo , V.M., Alberto Colorni Ant system: Optimization by a colony of cooperating agents. IEEE transactions on systems, man, and cybernetics-part b 1996: p. 29-41.
2. Marco Dorigo , L.M.G., Ant Colonies for the Traveling Salesman Problem. BioSystems, 1997. 43: p. 73-81.
3. Marco Dorigo , L.M.G., Ant Colony System: A cooperative learning approach to the traveling salesman problem. IEEE transactions on evolutionary computation 1997. 1: p. 53-66.
4. Bernd Bullnheimer , R.F.H., Christine Strauss Applying the Ant System to the Vehicle Routing Problem, in 2nd International Conference on Metaheuristics. 1997. p. 1-12.
5. Bernd Bullnheimer , R.F.H., Christine Strauss An Improved Ant System Algorithm for the Vehicle Routing Problem Annals of Operations Research 1999. 89: p. 319-328.
6. Silvia Mazzeo, I.L., An Ant Colony Algorithm for the Capacitated Vehicle Routing, in Electronic Notes in Discrete Mathematics. 2004. p. 181-186.
7. Jun Zhang , X.H., X. Tan , J. H. Zhong , Q. Huang Implementation of an Ant Colony Optimization Technique for Job Shop Scheduling Problem. Transactions of the Institute of Measurement and Control 2006. 28: p. 93-108.
8. Dréo, J. and P. Siarry, Continuous interacting ant colony algorithm based on dense heterarchy. Future Generation Computer Systems, 2004. 20(5): p. 841-856.
9. Jiao, L., et al., Application of ACO in Continuous Domain, in Advances in Natural Computation. 2006, Springer Berlin / Heidelberg. p. 126-135.
10. Socha, K. and M. Dorigo, Ant colony optimization for continuous domains. European Journal of Operational Research, 2008. 185(3): p. 1155-1173.
11. Liao, T., Improved ant colony optimization algorithms for continuous and mixed discrete-continuous optimization problems. 2011, Université Libre de Bruxelles: Belgium.
12. Fardin Ebdali Mohammadi, M.R.M., Adaptation of Ants Colony Parameters Using Learning Automata, in Proceedings of 10th Annual CSI Computer Conference Iran, Telecommunication Research Center. 2005: Tehran. p. 972-980
13. Dorigo, M., Learning and Natural Algorithms, in Electrical Engineering. 1992, Politecnico di Milano.
14. Kalyanmoy Deb , A.P., Sameer Agarwal , T. Meyarivan A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 2000. 6: p. 182--197.
15. Holland, J.H., Adaptation in Natural and Artificial Systems. 1975: University of Michigan Press.
16. Narendra, K.S.a.T., M. A. L. , Learning Automata: An Introduction. 1989: Prentice Hall.
17. Thathachar, M.A.L. and P.S. Sastry, Varieties of learning automata: an overview. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2002. 32(6): p. 711-722.
18. de Oca, M.A.M., et al., Incremental Social Learning in Particle Swarms. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2011. 41(2): p. 368-384.
19. Marco A. Montes de Oca, T.S., Towards incremental social learning in optimization and multiagent systems, in Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation. 2008, ACM: Atlanta, GA, USA. p. 1939-1944.