یک روش جدید برای افزایش کارایی الگوریتم برنامه‌نویسی ژنتیک با استفاده از آتوماتای‌یادگیر

فاطمه حمیدی[۱]؛ محمدرضا میبدی[۲]

چکیده

الگوریتم برنامه‌نویسی ژنتیک، یکی از انواع الگوریتم‌های تکاملی است که در حل مسائل بهینه‌سازی و تولید خودکار برنامه‌ی کامپیوتری موفق بوده است. در این مقاله، از برنامه‌نویسی ژنتیک برای طراحی دسته‌بندی‌کننده‌ی مسائل C کلاسی استفاده شده است. در این روش، از بازنمایی چنددرختی برای نمایش کروموزم‌ها و همچنین از مفهوم ناشایستگی درخت نیز برای انتخاب ناشایسته‌ترین درخت در داخل کروموزم‌ها استفاده شده است. در این مقاله، روش جدیدی مبتنی برآتوماتای یادگیر برای تشخیص بهترین ژن از کروموزم‌ها جهت انجام عمل بازترکیبی در برنامه‌نویسی ژنتیک پیشنهاد شده است. در روش پیشنهادی، آتوماتای یادگیر با استفاده از بازخورد فضای جستجو، مناسب‌ترین ژن از کروموزم را جهت انجام عمل بازترکیبی انتخاب می‌کند. نتایج آزمایش‌ها بر روی چهار مجموعه‌داده برای مساله‌ی دسته‌بندی و نیز سه تابع استاندارد برای مساله‌ی بهینه‌سازی، نشان داده است که با روش پیشنهادی، کارائی برنامه افزایش و خطا کاهش یافته است.

کلمات کلیدی

برنامه‌نویسی ژنتیک، آتوماتای یادگیر، دسته‌بندی کننده ، نمایش چنددرختی.

A Novel Approach For Improve Genetic Programming Algorithm Using Learning Automata

Fateme Hamidi; Mohammad Reza Meybodi
Sama Technical and Vocational Training College, Islamic Azad University, Zanjan Branch, Zanjan, Iran
Amirkabir University of Technology, Department of Computer Engineering and Information Technology, Tehran, Iran

ABSTRACT

Genetic Programming is an evolutionary algorithm which has been successful in solving optimization problems and automatically producing computer programs. In this paper, we use genetic programming algorithm for designing classifiers for a *C*-class ($C \geq ۲$) problem. A multitree representation of chromosomes is used. We use concept of unfitness of a tree to select trees for genetic operations. In this paper, we propose a new approach based on learning automata for adaptation best gen of chromosome for crossover operation in genetic programming. In our approach, a learning automaton which uses feedbacks from the environment (search space) is used to select appropriate gen of chromosome for crossover operation.

The proposed learning automata based algorithm is tested on four datasets for classification problem and three standard functions for optimization problem and the results are compared with the results obtained from four existing methods. The effectiveness of our approach is demonstrated on several real data sets.

KEYWORDS

Genetic Programming, Learning Automata, Classifier, Multitree Representation.

۱. مقدمه

برنامه‌نویسی ژنتیک(GP)[۱]، یکی از روش‌های جستجو بوده که جزو خانواده‌ی محاسبات تکاملی می‌باشد[۱و۲]. این الگوریتم‌ها امروزه به طور وسیعی در مسائل مختلف موجود در جهان به کارگرفته شده‌اند. در میان کاربردهای موفقیت‌آمیز از محاسبات تکاملی، برنامه‌نویسی ژنتیک با

۱. آموزشکده فنی و حرفه ای سما ، دانشگاه آزاد اسلامی ، واحد زنجان، زنجان، ایران، fhhamidi@yahoo.com
۲. دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیر کبیر، تهران، ایران، mmeybodi@aut.ac.ir

داشتن ویژگی‌های با ارزشی همچون نمایش راه‌حل‌ها با طول متغیر و انعطاف‌پذیری بالا وفقدان انحرافات جمعیتی موقعیت مهمی را در محاسبات تکاملی حفظ نموده است[۳و۴]. برنامه‌نویسی ژنتیک در سال ۱۹۹۰ در آمریکا توسط کزا معرفی شد. در این روش برای نمایش افراد در جمعیت از درخت استفاده می‌شود به طوریکه این درخت‌ها می‌توانند هرعبارت محاسباتی را نمایش دهند[۱و۵]. بنابراین در برنامه‌نویسی ژنتیک، با استفاده از این درخت‌ها، رابطه‌ی پنهانی بین داده‌ها را به صورت روابط ریاضی بیان می‌گردد.

یکی از کاربردهای برنامه‌نویسی ژنتیک در دسته‌بندی داده‌ها می‌باشد به طوریکه آگنلی از برنامه‌نویسی ژنتیک برای کلاس‌بندی تصاویر استفاده کرد. عملگرهای حسابی ساده، توابع نمائی، توابع شرطی، ثابت‌ها و متغیرهای ویژگی برای تولید درخت‌ها مورد استفاده قرار می‌گیرند[۱]. بعد از ساخت درخت‌ها در برنامه‌نویسی ژنتیک، داده را به یک کلاس نسبت می‌دهند اگر پاسخ برای این کلاس مثبت باشد و پاسخ‌ها برای کلاس‌های دیگر منفی باشد[۶]. در این روش یک دسته‌بندی‌کننده‌ی چند درختی شامل C درخت مورد استفاده قرار می‌گیرد که هر کدام از درخت‌ها رکوردهای مربوط به یک کلاس ویژه را دسته‌بندی می‌کند. بعد از طی چندین نسل نهایتاً جمعیت به سمت دسته‌بندی کننده‌ی خاصی گرایش می‌یابد به طوریکه در جمعیت نهایی، دسته‌بندی کننده‌ای که بیشترین کارایی را دارد به عنوان بهترین دسته‌بندی کننده انتخاب می‌گردد. در این مقاله سعی شده تا با ارائه‌ی تکنیک جدیدی به نام GPGENLA، کارایی دسته‌بندی کننده‌ی طراحی شده با برنامه‌نویسی ژنتیکی را با استفاده از آتوماتای یادگیر[۲] افزایش و خطا را کاهش دهیم.

ادامه‌ی مقاله از بخش‌های زیر تشکیل شده است: در بخش ۲، روش طراحی دسته‌بندی کننده‌ی چند درختی مبتنی بر برنامه‌نویسی ژنتیک بیان شده است که شامل نحوه‌ی محاسبه‌ی شایستگی، محاسبه‌ی ناشایستگی درخت‌ها، بیان عملگرهای بازترکیبی و جهش ، بیان شرط خاتمه‌ی GP بوده و در بخش ۳، نیز به شرح مختصری از آتوماتای یادگیر و در بخش ۴، به چند نمونه از روش‌های بهبود برنامه‌نویسی ژنتیک از قبیل افزایش جمعیت،کاهش جمعیت و تغییر تصادفی جمعیت پرداخته می‌شود. در بخش ۵، الگوریتم پیشنهادی ارائه می‌گردد. نتایج آزمایش‌ها نیز در بخش ۶ ارائه شده است.

۲. دسته‌بندی‌کننده‌ی چنددرختی مبتنی بر GP

در این مقاله، برای دسته‌بندی کننده از ساختاری که دارگا[۳] و همکارانش بکار برده‌اند استفاده کرده‌ایم[۶]. در این ساختار، یک دسته‌بندی کننده‌ی D نگاشتی مانند $D: R^p \rightarrow N_{hc}$ است که یک بردار ویژگی در p بعد را به عنوان ورودی می‌گیرد و یک برچسب کلاس به آن نسبت می‌دهد. به طوریکه برای هر بردار $x \in R^p$ ، $D(x)$ یک بردار C بعدی است که فقط یک مولفه‌ی آن ۱ و تمام مولفه‌های دیگر آن ۰ است[۶و۷]. با توجه به اینکه دسته‌بندی‌کننده برای مسائل C کلاسی طراحی شده است بنابراین هر کروموزم برای هر کلاس یک درخت دارد. بنابراین، کروموزم i ام ، C درخت دارد که با $T_k , k = ۱ , ۲, ..., C$ مشخص می‌شود. بنابراین راه‌حل ممکن یا یک فرد در GP به صورت C درخت $(T_۱ , T_۲ , ... , T_C)$ نمایش داده می‌شود. برای هر الگوی x رابطه‌ی (۱) را داریم :

$$if \quad T_i(x) \geq 0 \quad and \quad T_j(x) < 0 \quad forall \quad j \neq i \quad i, j \in \{1,2,...,c\} \quad then \quad x \in class. \ i \qquad (۱)$$

مراحل دسته‌بندی‌کننده‌ی چنددرختی در زیربخش‌های زیر خلاصه می‌شود:

۱،۲. مقداردهی اولیه

هر یک از درخت‌های کروموزم به وسیله‌ی مجموعه‌ی توابع F شامل توابع حسابی و مجموعه‌ی ترمینال‌ها T شامل متغیرهای ویژگی و ثابت‌ها ایجاد می‌شوند. مجموعه‌ی توابع و مجموعه‌ی ترمینال‌های مورد استفاده در این روش به ترتیب $F = \{ +, - , * , / \}$ و $T = \{$متغیرهای ویژگی و $R\}$ است که R شامل ثابت‌های تولید شده به صورت تصادفی در بازه [۰،۰،۱۰،۰] است[۶].

۲،۲. آموزش و معیار شایستگی

مقدار شایستگی هر کروموزم از جمعیت با رابطه‌ی (۲) محاسبه می‌گردد[۶]:

$$f = \frac{number \quad of \quad training \quad samples \quad correctly \quad classified \, (for \, which \quad l_i = c)}{N} \qquad (۲)$$

با توجه به رابطه‌ی بالا، مقدار شایستگی برای هر کروموزم حاصل تقسیم تعداد نمونه‌های آموزشی که توسط آن کروموزم به درستی دسته‌بندی شده بر تعداد کل نمونه‌های آموزشی است. تعریف دسته‌بندی صحیح به وسیله‌ی درخت این است که برای الگوی x که متعلق به کلاس k است، در یک کروموزم اگر $T_k \ (x) \geq 0$ باشد درخت T_k الگوی x را درست دسته‌بندی کرده است و اگر $T_{j \neq k} \ (x) < 0$ باشد درخت T_j الگوی x را

درست دسته‌بندی کرده است. در هر مرحله تکرار بعد از اینکه با استفاده از تابع شایستگی بالا مقدار شایستگی هر کروموزم را به دست آوردیم در ادامه با استفاده از معیار انتخاب تورنمنت دو کروموزم را برای اعمال عملگرهای ژنتیکی انتخاب می‌نمائیم.

۳,۲. ناشایستگی درخت‌ها

بعد از انتخاب دو کروموزم با استفاده از روابط شایستگی ، بوسیله‌ی تابع ناشایستگی درخت ناشایسته در داخل هر کروموزم را به صورت زیر تعیین می‌نمائیم[۶]. برای درخت i ام در کروموزم انتخاب شده مقدار احتمال ناشایستگی آن با رابطه‌ی (۳) تعیین می‌گردد:

$$p_i = \frac{k_i}{\sum_{j=1}^{c} k_j} \qquad (۳)$$

p_i احتمال انتخاب i امین درخت به وسیله‌ی چرخ رولت به عنوان درخت ناشایسته برای انجام عملیات ژنتیک است و k_i ،تعداد نمونه‌های آموزشی که به درستی توسط این درخت دسته‌بندی نشده‌اند.

۴,۲. عملگر بازترکیبی

عملگر بازترکیبی نقش اساسی را درGP برای تکامل بازی می‌کند. برای اعمال این عملگر در ابتدا دو کروموزم را با استفاده از معیار انتخاب تورنمنت تعیین نموده و در ادامه درختی را در داخل این کروموزم‌ها با استفاده از چرخ رولت معین نموده به طوریکه اگر درخت انتخاب شده در کروموزم اول T_l^1 باشد و از آنجائیکه درخت‌های هم کلاس در دو کروموزم با هم ترکیب می‌شوند بنابراین درخت T_l^2 از کروموزم دوم نیز برای انجام عمل بازترکیبی در نظر گرفته می‌شود. در ادامه دو زیر درخت به صورت تصادفی از هر یک از درخت‌های انتخابی با یکدیگر تعویض شده سپس، درخت‌های T_l^1 و T_l^2 را برای همه‌ی $j = l+1,...,C$ با هم تعویض می‌کنیم[۶].

۵,۲. عملگر جهش

در جهش، یک کروموزم به طور تصادفی برای انجام عملیات جهش انتخاب می‌شود. سپس احتمال p_i ناشایستگی درخت‌های کروموزم محاسبه می‌شود و با توجه به این احتمال یک درخت در داخل کروموزم برای اعمال عملگر جهش انتخاب می‌شود. حال، یک نود تصادفی از درخت انتخاب می‌شود. اگر نود انتخاب شده تابع باشد، با نود تابع به طور تصادفی انتخاب شده جایگزین می‌شود و اگر نود انتخاب شده پایانه باشد ، با نود پایانه به طور تصادفی انتخاب شده ، جایگزین می‌شود بنابراین تغییرات بسیار کوچک است. برای ساختن تغییرات قابل توجه، این فرآیند چند بار تکرار می‌شود. با توجه به اینکه بعد از عمل جهش انتظار داریم که شایستگی درخت افزایش یابد بنابراین اگر شایستگی درخت جهش یافته بیشتر یا مساوی شایستگی درخت اصلی باشد درخت جهش یافته حفظ می‌شود در غیر این صورت، این درخت نادیده گرفته می‌شود[۶].

۶,۲. خاتمه‌ی GP

GP به دو صورت خاتمه می‌یابد؛ اگر تمامی N نمونه‌ی آموزشی به درستی به وسیله دسته‌بندی کننده، دسته‌بندی شوند و یا به تعداد نسل از پیش تعریف شده‌ی M برسیم. اگر تمامی N نمونه‌ی آموزشی به درستی دسته‌بندی شوند، بهترین فرد جمعیت[۴] همان دسته‌بندی‌کننده‌ی بهینه است در غیر این صورت، اگر GP با رسیدن به تعداد نسل از پیش تعیین شده خاتمه یابد باید ، بهترین فرد جمعیت با تکنیک OR-ing به دسته-بندی کننده‌ی بهینه تبدیل شود.

۳. آتوماتای یادگیر

آتوماتای یادگیر، ماشینی است که می‌تواند تعدادی متناهی عمل را انجام دهد. هر عمل انتخاب شده توسط یک محیط احتمالی ارزیابی می-شود و نتیجه‌ی ارزیابی در قالب سیگنالی مثبت یا منفی به آتوماتا داده می‌شود و آتوماتا از این پاسخ در انتخاب عمل بعدی تأثیر می‌گیرد. هدف نهایی این است که آتوماتا یاد بگیرد تا از بین اعمال خود، بهترین عمل را انتخاب کند. بهترین عمل، عملی است که احتمال دریافت پاداش از محیط را به حداکثر برساند[۸و۹].

محیط را می‌توان توسط سه‌تایی $E \equiv \{\alpha, \beta, c\}$ نشان داد که در آن $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه‌ی ورودی‌ها، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_m\}$ مجموعه‌ی خروجی‌ها و $c \equiv \{c_1, c_2, ..., c_r\}$ مجموعه‌ی احتمال‌های جریمه است. هرگاه β مجموعه‌ای دو عضوی باشد، محیط از نوع P است. 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 فاصله‌ی [۱و۰] را اختیار کند و در محیط از نوع S، $\beta(n)$ متغیر

تصادفی در فاصله‌ی [۰و۱] است. c_i احتمال اینکه عمل α_i نتیجه‌ی نامطلوب داشته باشد است. در محیط ایستا، مقادیر c_i بدون تغییر می‌مانند، حال آن‌که در محیط غیرایستا این مقادیر در طی زمان تغییر می‌کنند.

آتوماتاهای یادگیر با ساختار متغیر که در این مقاله از آنها استفاده شده است را می‌توان توسط چهارتایی $\{\alpha, \beta, p, T\}$ نشان داد که $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه‌ی عمل‌های آتوماتا، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_r\}$ مجموعه‌ی ورودی‌های آتوماتا، $p = \{p_1, .., p_r\}$ بردار احتمال انتخاب هریک از عمل‌ها و $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ الگوریتم یادگیری است. فرض می‌کنیم عمل α_i در مرحله‌ی n ام انتخاب شود.

- به هنگام پاسخ مطلوب از محیط، رابطه‌ی (۴) را داریم:

$$\begin{aligned} p_i(n+1) &= p_i(n) + a[1 - p_i(n)] \\ p_j(n+1) &= (1-a)\, p_j(n) \qquad \forall j \quad j \neq i \end{aligned} \qquad (۴)$$

- به هنگام پاسخ نامطلوب از محیط، رابطه‌ی (۵) را داریم:

$$\begin{aligned} p_i(n+1) &= (1-b)\, p_i(n) \\ p_j(n+1) &= (b/r-1) + (1-b)\, p_j(n) \quad \forall j \quad j \neq i \end{aligned} \qquad (۵)$$

در روابط (۴) و (۵)، a پارامتر پاداش و b پارامتر جریمه می‌باشند

۴. روش‌های بهبود برنامه‌نویسی ژنتیک

در این قسمت، سه روش بهبود برنامه‌نویسی ژنتیکی که به الگوریتم‌های PV موسوم هستند، به طور خلاصه توضیح داده می‌شود.

۱،۴. روش کاهش خطی اندازه‌ی جمعیت[۵]

این الگوریتم که PVR نامیده می‌شود، با یک جمعیت اولیه‌ی بسیار بیشتر از جمعیت اولیه‌ی برنامه‌نویسی ژنتیکی استاندارد (الگوریتم با اندازه‌ی جمعیت ثابت) شروع می‌شود و به همین دلیل تنوع جمعیت در ابتدای اجرای الگوریتم بیشتر است. در روش کاهش خطی اندازه‌ی جمعیت، اندازه‌ی جمعیت در ابتدای هر تکرار از الگوریتم، با استفاده از رابطه‌ی (۶) تعیین می‌شود[۱۰].

$$P(g) = m \times g + P_{PV}(0) \qquad (۶)$$

که m ضریب کاهش خطی، $P_{PV}(0)$ اندازه‌ی جمعیت اولیه در الگوریتم تغییر جمعیت و $P(g)$ اندازه‌ی جمعیت در شروع تکرار g است. یعنی جمعیت جدید را برای هر تکرار، $P(g)$ نفر که دارای بالاترین شایستگی هستند، تشکیل می‌دهند.

۲،۴. روش افزایش خطی اندازه‌ی جمعیت[۶]

این روش که PVI نامیده می‌شود، با جمعیتی کمتر از روش استاندارد شروع می‌شود و در حین اجرا اندازه‌ی جمعیت افزایش پیدا می‌کند. در روش افزایش اندازه‌ی جمعیت در ابتدای هر تکرار، تعداد مشخصی از بهترین افراد جمعیت بر اساس شایستگی انتخاب می‌شوند، عملگر جهش بر روی آنها اعمال می‌شود و سپس به جمعیت جدید اضافه می‌شوند. در این روش نیز، اندازه‌ی جمعیت در ابتدای هر تکرار از الگوریتم، با استفاده از رابطه‌ی (۶) تعیین می‌شود، با این تفاوت که در آن m ضریب افزایش خطی است[۱۰].

۳،۴. روش تغییر تصادفی اندازه‌ی جمعیت[۷]

در این روش که PVRAN نامیده می‌شود، اندازه‌ی جمعیت به صورت تصادفی در یک محدوده‌ی از قبل تعیین شده و با استفاده از روش‌های ذکر شده در الگوریتم‌های PVR و PVI، کاهش یا افزایش پیدا می‌کند[۱۰].

۵. روش پیشنهادی

در الگوریتم پیشنهادی، جهت افزایش کارائی GP از آتوماتای یادگیر استفاده شده است. آتوماتای یادگیر، با استفاده از بازخورد فضای جستجو ژن مناسب از کروموزم را جهت انجام عمل بازترکیبی انتخاب می‌کند. آتوماتای یادگیر استفاده شده در این مقاله از نوع p و دارای سه عمل "انتخاب چرخ رولت" ، "انتخاب تورنمنت" و " انتخاب تصادفی" است.

مراحل الگوریتم پیشنهادی به صورت زیر است:

۱- جمعیت اولیه‌ای با سایز P به صورت تصادفی و به روش نیمه‌کامل تولید می‌شود.

۲- مقدار شایستگی هر کروموزم از جمعیت محاسبه می‌شود.

۳- یک عملگر ژنتیکی با توجه به مقداراحتمال انتخاب می‌شود :

الف- اگر عملگر تکثیر باشد، یک فرد متناسب با شایستگی از جمعیت کنونی انتخاب می‌شود و در جمعیت جدید کپی می‌شود. این عملگر بر اصل انتخاب طبیعی و بقا شایسته ترین‌ها تاکید دارد.

ب - اگر عملگر بازترکیبی باشد:

a) دو کروموزم (C۱ و C۲) از جمعیت با روش تورنومنت انتخاب می‌شوند.

b) یکی ازعمل‌های آتوماتای یادگیر براساس بردار احتمال انتخاب عمل‌های آتوماتا انتخاب می‌شوند.

c) احتمال ناشایستگی $P'i$ ، i = ۱, ۲, ... ,C ، برای C درخت کروموزم C۱ محاسبه می‌شوند.

d) درخت (T'_L) را از کروموزم C۱ با استفاده از روش عمل انتخاب شده در مرحله‌ی b و بر اساس احتمال ناشایستگی $P'i$ انتخاب می‌شود.

e) یک نود تصادفی از هرکدام از درخت‌های T'_L و T''_L انتخاب می‌شود. احتمال انتخاب یک نود از نوع تابع q_f و احتمال انتخاب یک نود از نوع پایانه q_t است که $q_f + q_t$ = ۱ خواهد بود.

f) زیردرخت‌های ریشه‌دار نودهای انتخاب شده‌ی دو درخت با هم تعویض می‌شوند.

g) درخت‌های T'_j و T''_j ، برای همه j = L + ۱,...,C باهم تعویض می‌شوند.

ج - اگرعملگر جهش باشد:

a) یک کروموزم به طور تصادفی برای جهش انتخاب می‌شود.

b) احتمال Pi ناشایستگی درخت‌های کروموزم محاسبه می‌شود.

c) با توجه به Pi و با استفاده از روش انتخاب چرخ رولت، یک درخت ناشایسته برای جهش انتخاب می‌شود.

d) یک نود به صورت تصادفی در درخت انتخاب می‌شود.

e) اگرنود انتخاب شده تابع باشد ، با نود تابع به طور تصادفی انتخاب شده، جایگزین می‌شود. اگر نود انتخاب شده پایانه باشد ، با نود پایانه به طور تصادفی انتخاب شده، جایگزین می‌شود.

f) مرحله‌های d و e برای m ٪ از کل نودهای درخت تکرار می‌شود.

g) شایستگی درخت جهش یافته با شایستگی درخت اصلی مقایسه می‌شود. این کار روی۵۰٪ از نمونه‌های آموزشی کلاس i ام انجام می‌شود.

h) اگر دو شایستگی مساوی بودند، شایستگی دو درخت روی باقیمانده‌ی نمونه‌های آموزشی کلاس i ام محاسبه می‌شود.

i) اگر شایستگی درخت جهش یافته بیشتر بود، درخت جهش یافته مورد قبول است ، در غیر این صورت درخت جهش یافته با احتمال ۰,۵ حفظ می‌شود.

j) مراحل c تا i ، C بار (تعداد کلاس‌ها) تکرار می‌شود.

۴- مرحله‌ی ۳ تا زمانی که سایز جمعیت جدید P شود، تکرار می‌شود.

۵- با فرض اینکه A برابر با میانگین شایستگی افراد نسل جاری و B برابر با میانگین شایستگی افراد نسل قبلی باشد، سیگنال β (پاسخ محیط به عمل انتخاب شده توسط آتوماتای یادگیر) را مطابق رابطه‌ی (۷) محاسبه می‌کنیم:

$$\beta = \begin{cases} 1 & A - B \le treshold \\ 0 & A - B > treshold \end{cases} \qquad (۷)$$

۶- احتمال عمل‌های آتوماتای یادگیر را بر اساس سیگنال β و روابط (۴) و (۵) بروزرسانی می‌کنیم.

۷- برخلاف الگوریتم ژنتیک[۸]، GP همگرا نمی‌شود و مراحل ۲ تا ۶ تازمانی ادامه می یابد که راه‌حل خواسته شده بدست آید، در غیر این صورت، عملیات GP بر اساس تعداد نسل‌های از پیش تعیین شده خاتمه می‌یابد.

۶. مجموعه‌داده‌ها و نتایج آزمایش‌ها

۱,۶. مجموعه‌داده‌ها

در دسته‌بندی داده‌ها، ما از چهار مجموعه‌داده‌ی بیوپا[۹] و ایریس[۱۰] و گلس[۱۱] و یاست[۱۲] (این مجموعه‌داده‌ها از مخزن داده‌ای UCI Data Repository استخراج شده‌اند). جدول (۱) مشخصات چهار مجموعه‌داده را نشان می‌دهد.

جدول (۱) مشخصات ۴ مجموعه‌داده‌ی مورد استفاده

مجموعه‌داده	تعداد کلاس‌ها	تعداد ویژگی‌ها	اندازه‌ی مجموعه‌داده و توزیع کلاس‌ها
بیوپا	۲	۶	(۱۴۵ +۲۰۰) = ۳۴۵
ایریس	۳	۴	(۵۰ + ۵۰+۵۰) =۱۵۰
گلس	۷	۹	(۷۰+۷۶+۱۷+۱+۱۲+۹+۲۹) =۲۱۴
یاست	۱۰	۸	(۴۶۳+۵+۳۵+۴۴+۵۱+۱۶۳+۲۴۴+۴۲۹+۲۰+۳۰) =۱۴۸۴

۲,۶. توابع بهینه‌سازی استاندارد

در بهینه‌سازی توابع، ما از سه تابع استاندارد اسفیر[۱۳] و رزنبرگ[۱۴] و رستریجین[۱۵] برای انجام آزمایش‌ها استفاده کرده‌ایم. فرمول توابع نامبرده در روابط (۸) و (۹)و (۱۰) و مشخصات آنها در جدول (۲) آمده است.

۱ - تابع اسفیر: (۸)

$$f_1(x) = \sum_{i=1}^{D} x_i^2$$

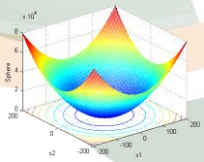
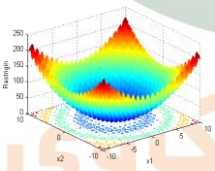
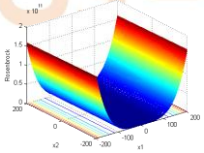
۲ - تابع رزنبرگ: (۹)

$$f_2(x) = \sum_{i=1}^{D} \left(100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2 \right)$$

۳ - تابع رستریجین: (10)

$$f_3(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10 \right)$$

جدول (۲) توابع مورد استفاده در بهینه‌سازی توابع

نام تابع	محدودهٔ فضای جستجو	شکل دو بعدی
اسفیر	$[-۵,۱۲،۵,۱۲]^D$	
رسترجین	$[-۵,۱۲،۵,۱۲]^D$	
رزنبرگ	$[-۵،۱۰]^D$	

۳,۶. نتایج آزمایش‌ها

تمامی پیاده‌سازی‌ها در محیط Visual Studio۲۰۰۸ و به زبان C# انجام گرفته است و نتایج آزمایش‌ها در ادامه آورده شده است. از آنجا که مجموعه‌داده‌های استفاده شده در این مقاله، دارای مجموعه‌ی تست جداگانه نمی‌باشند بنابراین ما در هر مجموعه‌داده به صورت تصادفی، ۱/۵ از داده‌ها را برای تست الگوریتم جدا کرده و مابقی را برای آموزش استفاده کرده و سپس الگوریتم را ۵۰ بار اجرا کرده و میانگین خطای دسته‌بندی غلط نمونه‌های تست در ۵۰ بار اجرا را یافته‌ایم.

جدول(۳)، مقادیر پارامترهای عمومی مورد استفاده در پیاده‌سازی را نشان می‌دهد. جدول‌های (۴) و (۵)، بهترین خطای نمونه‌های تست و میانگین بهترین خطای نمونه‌های تست در ۵۰ بار اجرا برای مجموعه‌داده‌های بیوپا و ایریس و گلس و یاست را نشان می‌دهد. مقادیر بهتر به صورت پر رنگ نشان داده شده است. شکل‌های (۱) تا (۴)، نمودارهای میانگین بهترین خطای نمونه‌های تست الگوریتم پیشنهادی GPGENLA در مقایسه با سایر الگوریتم‌ها را در مجموعه‌داده‌های بیوپا و ایریس و گلس و یاست نشان می دهند.

جدول (۶)، بهترین خطای نمونه‌های تست در۵۰ بار اجرا برای توابع اسفیر و رستریجین و زرنبرگ را نشان می‌دهد. مقادیر بهتر به صورت پر رنگ نشان داده شده است. شکل‌های (۵) تا (۷)، نمودارهای میانگین بهترین خطای نمونه‌های تست الگوریتم پیشنهادی GPGENLA در مقایسه با سایر الگوریتم‌ها را در بهینه‌سازی تابع اسفیر ۵و ۱۰و ۱۵بعدی نشان می دهند.

جدول (۳) مقادیر پارامترهای عمومی

پارامتر	مقدار	پارامتر	مقدار	پارامتر	مقدار
اندازه‌ی جمعیت(μ)	۱۰۰	انتخاب نا شایسته‌ترین درخت کروموزم	توسط آتوماتا	تعداد نودها درجهش	٪۲
تعداد فرزندان(λ)	$\mu = \lambda$	روش کنترل اندازه‌ی درخت	آستانه‌ی قطعی	اندازه‌ی تورنمنت	۱۰/ اندازه‌ی جمعیت
نرخ جهش	۰,۱۵	عمق اولیه‌ی درخت‌ها	۳	عمق نهایی درخت‌ها	۶
نرخ بازترکیبی	۰,۷۵	تابع شایستگی	خطای دسته-بندی	ماکسیمم تعداد نسل‌ها	۵۰
نرخ بازتولید	۰,۱			معیار خاتمه	رسیدن به شایستگی ۱۰۰٪
انتخاب والدین	تورنمنت	احتمال انتخاب نود تابعی در حین عملیات بازترکیبی	۰,۸		
انتخاب بازماندگان	جایگزینی	احتمال انتخاب نود پایانه در حین عملیات بازترکیبی	۰,۲	تعداد اجرای آزمایش	۵۰
نوع انتخاب بازماندگان	(μ, μ)	احتمال انتخاب نود تابعی در حین عملیات جهش	۰,۹	پارامتر پاداش آتوماتا	a =۰,۱
روش تولید جمعیت اولیه	نیمه‌کامل	احتمال انتخاب نود پایانه در حین عملیات جهش	۰,۱	پارامترجریمه‌ی آتوماتا	b =۰,۱

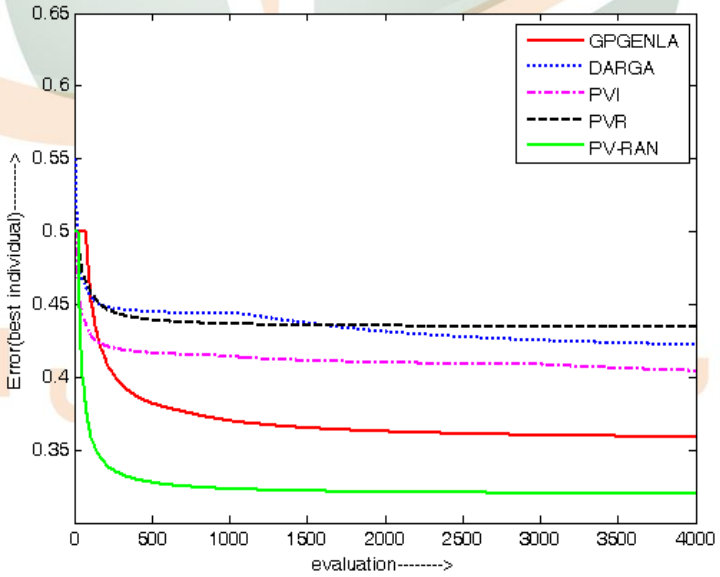
جدول (۴) مقایسه‌ی الگوریتم پیشنهادی GPGENLA با سایر الگوریتم‌ها در مجموعه‌داده‌های بیوپا و ایریس

مجموعه‌داده	روش	بهترین خطای نمونه‌های تست	میانگین بهترین خطای نمونه‌های تست
مجموعه داده بیوپا	روش دارگا	۰,۴۱۳۰۴۳۴۷۸	۰,۴۲۲۰۸۶۳۳۵
	PVI	۰,۳۹۱۳۰۴۳۴۸	۰,۴۰۴۲۴۲۴۵۶
	PVR	۰,۴۳۳۸۲۳۵۲۹	۰,۴۳۴۴۹۸۳۵۲
	PVRAN	۰,۳۱۸۸۴۰۵۸	۰,۳۱۹۹۲۰۱۹۵
	GPGENLA	**۰,۲۳۴۳۱۷**	**۰,۲۶۷۶۴۴**
مجموعه داده ایریس	روش دارگا	۰,۱۲۵	۰,۱۴۰۶۴۳۶۲۳
	PVI	۰,۲۷۵۸۶۲۰۶۹	۰,۲۸۴۶۳۰۰۳۳
	PVR	۰,۲	۰,۲۱۵۱۲۳۸۲۵
	PVRAN	۰,۲۴۴۴۴۴۴۴	۰,۲۵۲۷۳۴۷۳۷

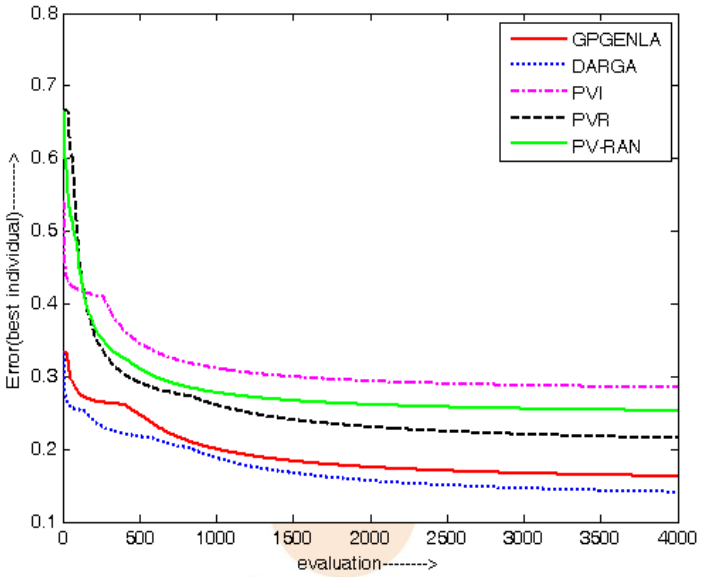
	GPGENLA	۰,۱۱۱۱۲۳	۰,۱۰۲۴۷۵۱۶۹

جدول (۵) مقایسه‌ی الگوریتم پیشنهادی GPGENLA با سایر الگوریتم‌ها در مجموعه‌داده‌های گلس و یاست

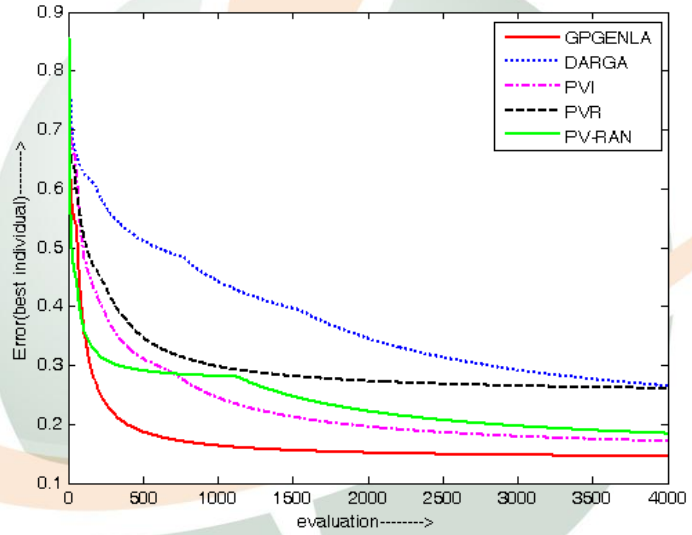
مجموعه‌داده	روش	بهترین خطای نمونه‌های تست	میانگین بهترین خطای نمونه‌های تست
مجموعه داده گلس	روش دارگا	۰,۱۸۳۶۷۳۴۶۹	۰,۲۶۴۷۵۵۳۴۶
	PVI	۰,۱۴۶۲۵۸۵۰۳	۰,۱۷۰۹۰۳۱۶۵
	PVR	۰,۲۴۸۲۹۹۳۲	۰,۲۶۰۷۰۴۷۵۹
	PVRAN	۰,۱۴۶۲۵۸۵۰۳	۰,۱۸۴۰۵۴۵۵۱
	GPGENLA	**۰,۱۳۹۴۵۵۷۸۲**	**۰,۱۴۵۳۹۰۹۳۳**
مجموعه داده یاست	روش دارگا	۰,۲۲۸۹۵۶۲۲۹	۰,۳۵۴۲۰۷۹۲۲
	PVI	۰,۱۵۹۹۳۲۶۶	۰,۱۸۳۹۵۱۱۵۲
	PVR	۰,۱۵۱۸۵۱۸۵۲	۰,۱۸۸۹۴۵۸۷۵
	PVRAN	۰,۱۴۲۰۸۷۵۴۲	۰,۱۸۵۱۱۲۹۴۹
	GPGENLA	**۰,۱۰۲۰۲۰۲۰۲**	**۰,۱۲۳۸۶۸۴۰۲**



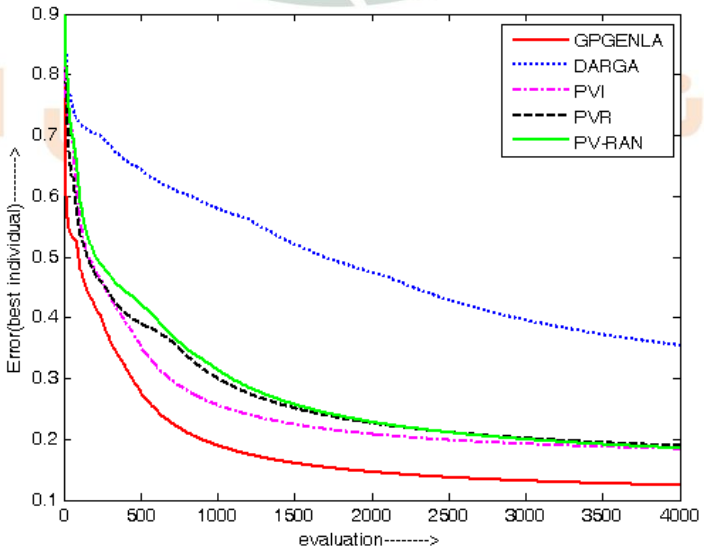
شکل (۱) مقایسه‌ی میانگین بهترین خطای نمونه‌های تست الگوریتم پیشنهادی GPGENLA با سایر الگوریتم‌ها در مجموعه‌داده‌ی بیوپا



شکل (۲) مقایسه‌ی میانگین بهترین خطای نمونه‌های تست الگوریتم پیشنهادی GPGENLA با سایر الگوریتم‌ها در مجموعه‌داده‌ی ایریس



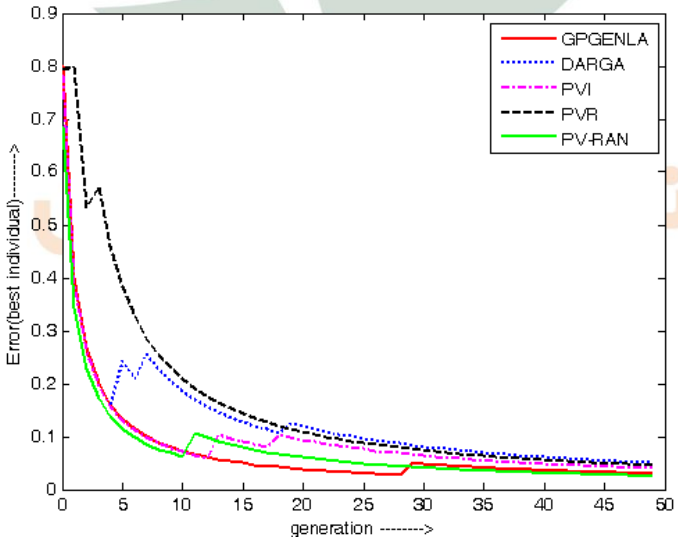
شکل (۳) مقایسه‌ی میانگین بهترین خطای نمونه‌های تست الگوریتم پیشنهادی GPGENLA با سایر الگوریتم‌ها در مجموعه‌داده‌ی گلس



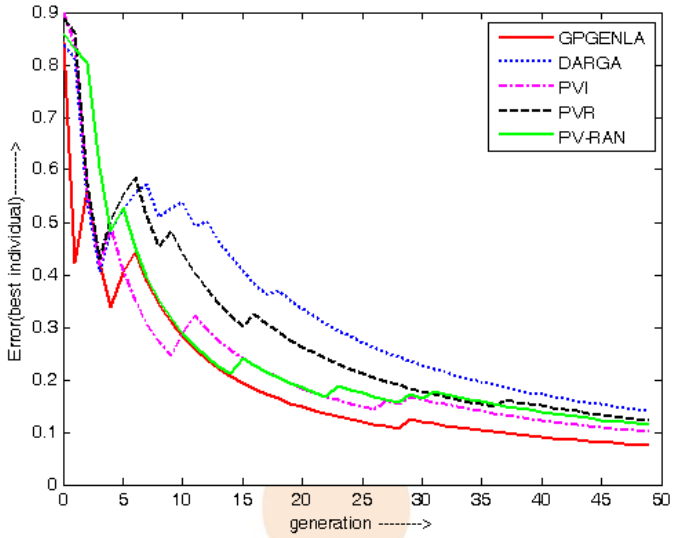
شکل (۴) مقایسه‌ی میانگین بهترین خطای نمونه‌های تست الگوریتم پیشنهادی GPGENLA با سایر الگوریتم‌ها در مجموعه‌داده‌ی یاست

جدول (۵) مقایسه‌ی الگوریتم پیشنهادی GPGENLA با سایر الگوریتم‌ها در بهینه‌سازی توابع اسفیر و رستریجین و رزنبرگ

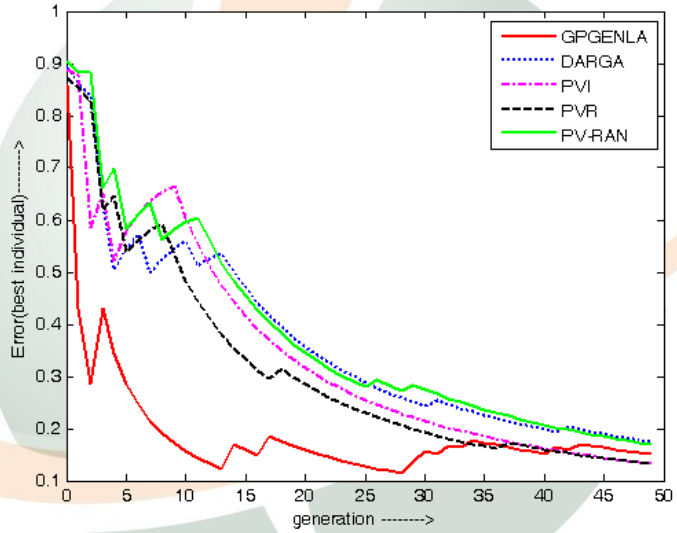
مجموعه‌داده	روش	بهترین خطای نمونه‌های تست		
		تعداد ابعاد = ۵	تعداد ابعاد = ۱۰	تعداد ابعاد = ۱۵
تابع اسفیر	روش دارگا	۰,۰۵۰۳۹۱۷۵۳	۰,۱۴۰۹۲	۰,۱۷۴۵۸۶۶۶۷
	PVI	۰,۰۳۹۷۲	۰,۱۰۰۹۲۹۲۹۳	۰,۱۳۲۶۵۳۳۳۳
	PVR	۰,۰۴۵۸	۰,۱۲۳۰۴	۰,۱۳۱۶۲۶۶۶
	PVRAN	۰,۰۲۵۵۲	۰,۱۱۳۷۵۵۱۰۲	۰,۱۷۰۱۹۰۴۷۶
	GPGENLA	**۰,۰۲۳۹۶**	**۰,۰۷۴۶۰۶۰۶**	**۰,۱۱۰۱۳۳۳۳**
تابع رستریجین	روش دارگا	۰,۰۵۹۹۶	۰,۰۷۹۹۸	۰,۰۷۹۹۱۸۳۶۷
	PVI	۰,۰۴	۰,۰۷۹۹۲	۰,۱
	PVR	۰,۰۳۹۹۲	۰,۰۶	۰,۰۵۹۹۶
	PVRAN	۰,۰۳۹۹۵۵۶	۰,۰۵۹۹۶	۰,۰۷۹۹۳۳۳۳۳
	GPGENLA	**۰,۰۳۴۵۹۲**	**۰,۰۳۹۹۴**	**۰,۰۴۹۹۸۶۵۳۲**
تابع رزنبرگ	روش دارگا	۰,۰۴	۰,۰۶	۰,۰۲۷۳۴
	PVI	۰,۰۲۴۳۷	۰,۰۴۵	۰,۰۵۳۴۵۳
	PVR	۰,۰۲۲۲۲۲	۰,۰۳۴۳۱۳	۰,۰۴۲۲۱۳
	PVRAN	۰,۰۳۳۲۴	۰,۰۳	۰,۰۳
	GPGENLA	**۰,۰۲۱۲۵**	**۰,۰۱۸۴۴۴**	**۰,۰۱۵۳۴**



شکل (۵) مقایسه‌ی میانگین بهترین خطای نمونه‌های تست الگوریتم پیشنهادی GPGENLA با سایر الگوریتم‌ها در بهینه‌سازی تابع اسفیر ۵ بعدی



شکل (۶) مقایسه‌ی میانگین بهترین خطای نمونه‌های تست الگوریتم پیشنهادی GPGENLA با سایر الگوریتم‌ها در بهینه‌سازی تابع اسفیر ۱۰ بعدی



شکل (۷) مقایسه‌ی میانگین بهترین خطای نمونه‌های تست الگوریتم پیشنهادی GPGENLA با سایر الگوریتم‌ها در بهینه‌سازی تابع اسفیر ۱۵ بعدی

۷. نتیجه

در این مقاله، دسته‌بندی کننده‌ای استفاده کردیم که فقط به یکبار اجرای GP نیاز دارد تا دسته‌بندی کننده‌ی بهینه را برای مسئله‌ی چند کلاسی به دست آورد. برای دسته‌بندی کننده‌ی C کلاسی ، دسته‌بندی‌کننده‌ی چنددرختی شامل C درخت است که هر درخت یک دسته‌بندی کننده را برای یک کلاس ویژه نمایش می‌دهد. هر درخت الگوهای یک کلاس خاص را تشخیص می‌دهد و الگوهای کلاس‌های دیگر را رد می‌کند. درخت‌ها مستقل از هم هستند و هرکدام مسئولیت مساوی در دسته‌بندی‌کننده دارند و همه‌ی درخت‌ها در ارزیابی شایستگی کروموزم تاثیر دارند. ما تکنیک جدید GPGENLA را برای افزایش کارایی مطرح نمودیم. نتایج حاصل از آزمایش‌ها، حاکی از توانایی بالای روش پیشنهادی در دسته-بندی داده‌ها و بهینه‌سازی توابع دارد.

۸. مراجع

[۱] J.R. Koza; "*Genetic Programming: on the Programming of Computers by Means of Natural Selection*", MIT Press, Cambridge, MA, ۱۹۹۲

[۲] N.L. Cramer; "*A representation for the adaptive generation of simple sequential programs*", in:

J.J. Grefenstette (Ed.), Proceedings of an International Conference on Genetic Algrorithms and the Applications, Carnegie Mellon University, Pittsburgh, PA, USA, pp. ۱۸۳–۱۸۷.

[۳] P.A. Whigham; "*Grammatically based genetic programming*", in: J.P. Rosca (Ed.), Proceedings of the Workshop on Genetic Programming: From Theory to Real World Applications, Tahoe City, CA, USA, pp. ۳۳–۴۱.

[۴] Y. Shan; R.I. McKay; R. Baxter; H. Abbass; D. Essam; H.X. Nguyen; "*Grammar model-based program evolution*", in: Proceedings of the Congress on Evolutionary Computation,Portland, US, pp. ۴۷۸–۴۸۵, ۲۰۰۴.

[۵] J.K. Kishore, L. M. Patnaik, V.Mani, and V.K. Agrawal, "*Application of genetic programming for multicategory pattern classification*," IEEE Trans. Evol. Comput, VOL. ٤, PP. ۲٤۲–۲٥۸, SEPT. ۲۰۰۰.

[۶] Durga Prasad Muni, Nikhil R. Pal, and Jyotirmoy Das, "*A Novel Approach to Design Classifiers Using Genetic Programming*", IEEE Trans. On Evolutionary Computation, VOL. ۸, NO. ۲, APRIL ۲۰۰٤.

[۷] C. Fonlupt; "*Solving the ocean color problem using a genetic programming approach*," *Appl. Soft Comput.*, vol. 1, pp. 63–72, June,2001.

[۸] K. S. Narendra and M. A. Thathachar, "Learning Automata: An Introduction"; Prentice Hall, ۱۹۸۹.

[۹] N. R. Harvey; S. P. Brumby; S. Perkins; J. Theiler; J. J. Szymanski; J. J. Bloch; R. B. Porter; M. Galassi; A. C. Young; "*Image feature extraction: GENIE vs conventional supervised classification techniques*," IEEE Trans. Geosci. Remote Sensing, vol. ۴, pp. ۳۹۳–۴۰۴, ۲۰۰۲.

[۱۰] P. Kouchakpour; "*Population Variation in Genetic Programming*", Information Sciences, vol. *۱۷۷*, pp. ۳۴۳۸-۳۴٥۲, ۱ September, *۲۰۰۷.*

---

۱ Genetic Programming
۲ Learning Automata
۳ Durga
٤ BCF
٥ Population Variation Reduce
٦ Population Variation Increase
۷ Population Variation Random
۸ Genetic Algorithm
۹ BUPA
۱۰ IRIS
۱۱ GLASS
۱۲ YEAST
۱۳ Sphere
۱٤ Rosenbrock
۱٥ Rastrigin