آزمایش نرم افزار و بکارگیری الگوریتم تکاملی هوش مصنوعی

محمد خدایار[۱]، سید محسن هاشمی[۲] و منصور امینی لاری[۳]
[۱] دانشگاه آزاد اسلامی، علوم تحقیقات، واحدفارس، شیراز، ایران،Md.khodayar@Gmail.com
[۲] دانشگاه آزاد اسلامی، واحدمیبد، باشگاه پژوهشگران جوان و نخبگان، میبد، ایران،Sayedmohsenhashemi@Yahoo.com
[۳] دانشگاه آزاد اسلامی، واحد علوم تحقیقات فارس،گروه فناوری اطلاعات و کامپیوتر،شیراز،ایران، Aminilari@Gmail.com

چکیده - امروزه استفاده از روش ها و ابزار هایی که بتوان یک نرم افزار را مورد ارزیابی و کنترل قرار دهد ارزش و اعتبار بالایی دارد.آزمایش فرایندی برای بررسی صحت،کامل بودن و کیفیت نرم افزار است. همانطورکه آزمایش نرم افزار یک بخش مهم و با ارزش از چرخه حیات توسعه نرم افزار است ولی آزمایش فرآیندی وقت گیر، پر زحمت و پرهزینه می باشد لذا آزمایش جامع همیشه عملی نیست و نیاز به استفاده از تکنیک هایی جهت کاهش هزینه ها و کاهش زمان داریم. آزمایش علاوه بر یافتن خطاهای نرم افزار، برای آزمایش عملکرد، ایمنی و تحمل خطا یا موارد امنیتی نیز مورد استفاده قرار می گیرد. روش ها و راه حل هایی تا کنون ارائه گردیده اند. در این مقاله به معرفی الگوریتم ژنتیک از الگوریتم های تکاملی در هوش مصنوعی پرداخته می شود.

کلید واژه- آزمایش، آزمایش نرم افزار، آزمایش نرم افزار بوسیله ژنتیک، الگوریتم ژنتیک

۱- مقدمه

آزمایش یکی از معروف ترین و پر استفاده ترین روشهای ارزیابی کیفیت نرم افزار است. دو فرایند مهم وجود دارد که در آزمایش شی گرا نرم افزار استفاده می شود. اول، نرم افزار باید با مجموعه ای از مقادیر مقداردهی اولیه شود. این مقادیر برای تنظیم کردن تعدادی از متغیرها که مربوط به تست می باشد استفاده می شود [۸و۲] .

مقادیر این متغیرها را در یک حالت واحد از مجموعه حالات که توسط این نرم افزار تعیین شده است تعریف می کند. این مقادیر هم می تواند یک مقدار اولیه مانند مقادیر صحیح یا پیچیده مانند یک شی باشند. با استفاده از تست مقداردهی اولیه نرم افزار [۷] و با گرفتن یک یا چند خصوصیات نرم افزار، خروجی نرم افزار و ورودی معتبر تعیین می شود. از آنجا که تعدادی از اشیاء به عنوان پارامتر هستند، این اشیاء نیاز به مقداردهی اولیه دارند. برای تعیین اینکه آیا فرایند تست کردن موفق بوده است یا خیر، خصوصیات نرم افزار [۶] باید استفاده شود. تعداد حالتهای یک نرم افزار ممکن است نمایی[۱] باشد. در این حالت آزمایش کردن همه آنها غیر ممکن است.

تست نرم افزار [۱۰و۷و۴] یکی از اصلی ترین تکنیک های اولیه برای بدست آوردن کیفیت بالای نرم افزار می باشد. تست نرم افزار برای تشخیص وجود خطاها [۶و۵] که به علت خرابی نرم افزار ها است انجام می شود. با این حال، تست نرم افزار کار وقت گیر و گرانی است.برای پیاده سازی یک نرم افزار می بایست اصولی را رعایت نمود که معمولاً به شرح زیر می باشد:

[1] Exponential

۱. جمع آوری داده
۲. تحلیل داده ها
۳. طراحی
۴. آزمایش
۵. پیاده سازی

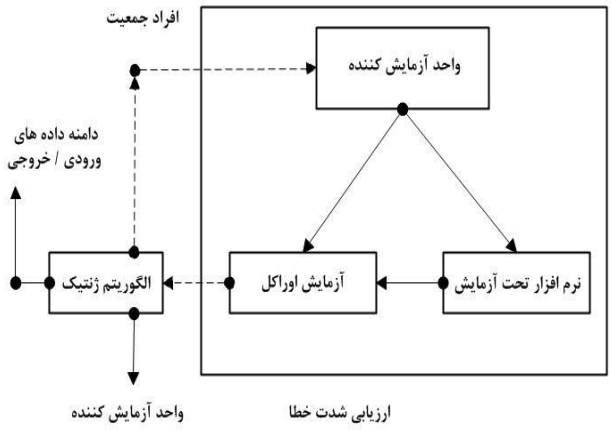
شکل ۱: فرآیند های یک محصول نرم افزاری

۲- الگوریتم ژنتیک

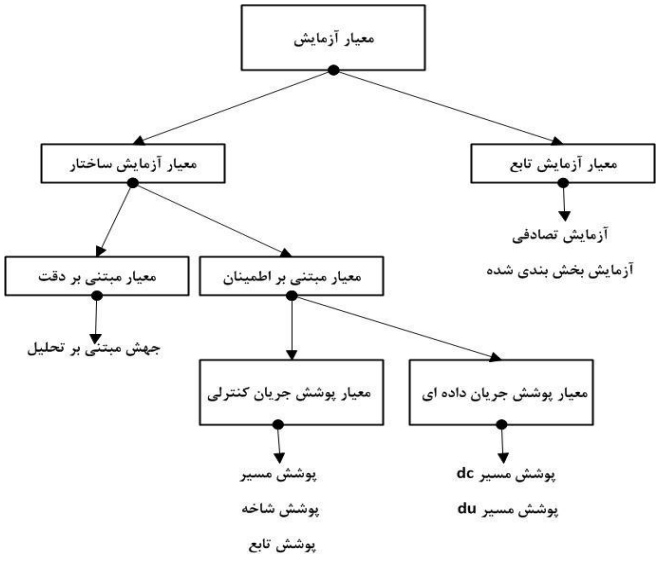
ایده اولیه الگوریتم ژنتیک برای رمزگذاری مقادیر پارامترها یک مسئله بهینه سازی در یک کروموزوم است. که توسط تابع هدف ارزیابی شده و برای طیف گسترده ای از برنامه های کاربردی که شامل بهینه سازی، برنامه ریزی، و مشکلات طراحی هستند با موفقیت پیاده سازی شده است. الگوریتم ژنتیک در مسائل بهینه سازی بسیاری اعمال شده است و همچنین مبتنی بر مدل تولید، آزمایش و استفاده قرار گرفته است. همانطور که در شکل۴ نشان داده شده است. الگوریتم توسط مقدار دهی اولیه و یا به صورت تصادفی تولید مجموعه ای از کروموزوم (جمعیت) شروع می شود.

در پایان هر نسل، هر یک از کروموزوم ها و با توجه به تعدادی از عملیات ها و به منظور تولید جمعیت جدید ارزیابی و اصلاح می شود [۳ و ۱].

این روند تا زمانی تکرار می شود تا یک شماره از پیش تعریف شده از تعداد نسل ها ایجاد شده محاسبه شود. استفاده از الگوریتم ژنتیک برای آزمودن تولید داده ها و برای آزمایش نرم افزار، یک فرایند شناسایی است. مجموعه ای از داده ها ورودی برنامه، با توجه به معیار تست ارضا شده است [۹ و ۸].



شکل ۲ : رویکرد الگوریتم ژنتیک مبتنی بر آزمایش نرم افزار



شکل ۳ : چارچوب طبقه بندی شده کلی برای محتوای آزمایش نرم افزار

۲-۱- گام های اساسی الگوریتم ژنتیک

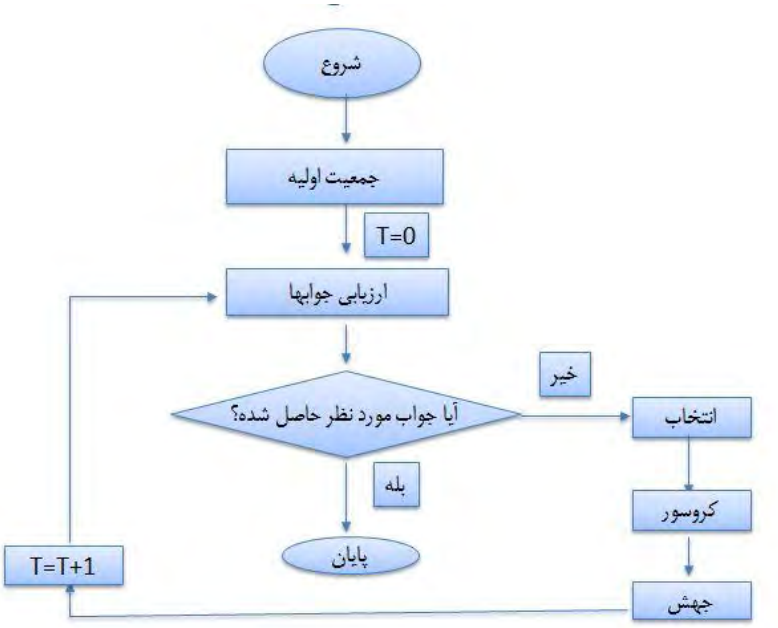
جمعیت: جمعیت اولیه عبارت است از یک نسل از کروموزوم‌های اولیه، که به شکل کاملاً تصادفی تولید شده‌اند.

انتخـاب: تعدادی از کروموزوم های والد بر مبنای مقدار برازندگی‌ای که تابع ارزیابی از آنها بدست آورده است، انتخاب می‌گردند.

ترکیب: ویژگیهای کروموزوم‌های والد با یکدیگر تلفیق شده و یک یا چند فرزند تولید می‌گردد.

جهش: جهش، یک یا چند ژن در کروموزوم فرزند تغییر پیدا می‌کند.

جایگزینی: طی این عملیات فرزندان تولید شده جایگزین والدین خود می‌شوند.



شکل ۴ : نمودار جریان الگوریتم ژنتیک

۲-۲- معمول ترین روش های انتخاب

Elitist Selection : مناسب‌ترین عضو هر اجتماع انتخـاب می‌شود.

Roulette Selection: یک روش انتخاب است که در آن عنصری که عدد برازش (تناسب) بیشتری داشته باشد، انتخاب می‌شود.افراد با توجه به ارزش های شایستگی خود نسبت به مردم انتخاب می شوند. احتمال [15] که از یک فرد برداشته شده است برابر :

$$p1 = \frac{fitness}{\sum_{i=0} len(population) fitness_i} (۱)$$

- **ScalingSelection:** به موازات افزایش متوسط عدد برازش جامعه، سنگینی انتخاب هم بیشتر می‌شود و جزئی‌تر. این روش وقتی کاربرد دارد که مجموعه دارای عناصری باشد که عدد برازش بزرگی دارند و فقط تفاوت‌های کوچکی آن‌ها را از هم تفکیک می‌کند.
- **Tournament Selection:** یک زیر مجموعه از صفات یک جامعه انتخاب می‌شوند و اعضای آن مجموعه با هم رقابت می‌کنند و سرانجام فقط یک صفت از هر زیرگروه برای تولید انتخاب می‌شوند.

۲-۳- انواع ترکیب:

- **بازترکیبی تک نقطه ای**[۲]:

اگر عملیات بازترکیبی را در یک نقطه انجام دهیم به آن بازترکیبی تک نقطه ای می گویند.در این روش یک مکان تصادفی در طول رشته انتخاب می شود و ژنها از این مکان به بعد جابجا می شوند.

- **روش ادغام دو نقطه ای**[۳]: در این روش دو مکان را به صورت تصادفی انتخاب کرده و مقادیر بین این دو نقطه را جابجا می کنیم.

- **ادغام چند نقطه ای**[۴]: می توانیم این عملیات را در چند نقطه انجام دهیم،که به آن بازترکیبی چند نقطه ای می گویند.

- **ادغام جامع**[۵]: اگر تمام نقاط کروموزوم را بعنوان نقاط بازترکیبی انتخاب کنیم به آن بازترکیبی جامع می گوئیم.

۲-۴- سه عملیات اساسی جهش

۱-**پرش جهش کننده**[۶]، تنها یک ژن از کروموزوم را به یک مقدار تصادفی با توجه به محدوده مشخص شده توسط آلل (آلل یک فرم جایگزین یک ژن است که در آن در یک موقعیت خاص بر روی یک کروموزوم خاص واقع شده است) تغییر می دهد[۱۳].

۲-**مبادله گر نامتجانس**[۷]: به صورت تصادفی تعدادی از ژن های کروموزوم را تعویض می کند.

۳-**جهش کننده گاوسی**[۸]: یک مقدار جدید را از اطراف مقدار جاری با استفاده از یک توزیع Guassian انتخاب می کند [۲].

[2] Single Point Crossover
[3] Two-point Cross Over
[4] Multipoint Crossover
[5] Uniform Crossover
[6] Flip mutator
[7] Swap mutator

عمل جهش با توجه به ساختار کروموزوم تعریف می شود. هنگامی که کروموزوم در یک درخت ذخیره می شود جهش ممکن است آن را تبدیل به یک زیر درخت کند که در شکل ۵ نشان داده شده است.کرموزوم ها فراهم کننده های اساسی هستند که تصمیم می گیرند که آیا باید جهش انجام گیرد یا خیر.

۲-۵- ویژگی های اصلی الگوریتم ژنتیک

- تمرکز بر روی کروموزوم ها با شایستگی بالاتر از حد متوسط
- سوء استفاده از اطلاعات در مورد تعداد زیادی از مقادیر در حالی که پردازش یک جمعیت کوچک است.
- مانع از جستجوی راکد در بهینه محلی است.
- با استفاده از دانش های قدیمی در یک جمعیت گرفته شده راه حل های برای تولید راه حل های جدید با بهبود عملکرد استفاده می کند.

۲-۶- بهره وری

بهره وری از این الگوریتم می تواند با ترکیب الگوریتم ژنتیکی و تست خودکار در یک سیستم واحد بهبود پیدا کند. در هر حال هر بار تست خودکار توسط الگوریتم ژنتیک فراخوانی می شود که باید تست تحت کلاس بارگذاری و تجزیه[۹] شود.

۲-۷- سنجش کد[۱۰]

در حال حاضر اطلاعات در مورد پیچیدگی روش های که در حال آزمایش هستند در نظر گرفته نشده است. در هنگام مقداردهی اولیه استراتژی های آزمایش سنجش کد [۱۶] برای مقدار دهی اولیه دقیق می تواند مورد استفاده قرار گیرد.

[8] Gaussian mutator
[9] parse
[10] Code Metrics

۳- مدل سازی :

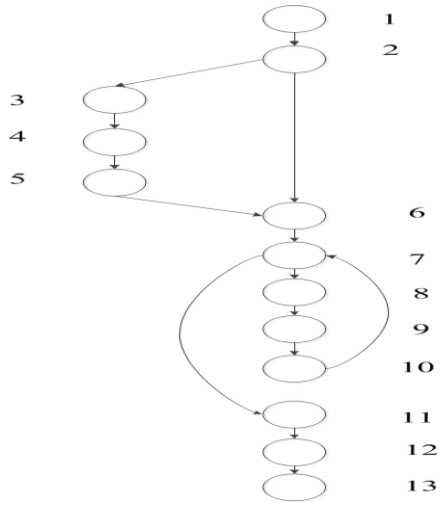
در این روند ما بایستی راه حل مورد نیاز مساله را به گونه ای تعریف کنیم که قابل نمایش بوسیله یک کروموزوم باشد. چند نمونه از بازنمائی هایی را که معمولاً استفاده می شوند به شرح زیر است :

۱. اعداد صحیح
۲. رشته های بیتی
۳. اعداد حقیقی در فرم نقطه شناور
۴. اعداد حقیقی به فرم رشته های بیتی
۵. یک مجموعه از اعداد حقیقی یا صحیح
۶. ماشینهای حالت محدود
۷. هر فرم دیگری که بتوانیم عملگرهای ژنتیک را بر روی آنها تعریف کنیم.

داده یا کرموزم بوسیله الگوریتم ژنتیک بررسی می گردد که به دو دسته ساختاری و تابعی تقسیم بندی میشود. دسته بندی تابعی بصورت تصادفی و با دقت کمتری از دسته بندی ساختاری انجام میشود. در مقابل دسته بندی تابعی ، دسته بندی ساختاری با دقت و اطمینان بیشتر را داریم که یکی از عملگر های الگوریتم ژنتیک به نام جهش این وظیفه رو بعهده دارد. روش دیگری نیز وجود دارد که هم بصورت داده ای یعنی کرموزومی و یا جریانی کنترل میشود که باعث بالارفتن درصد اطمینان از آزمایش نرم افزار می شود. نمونه برنامه زیر را در نظر بگیرید.

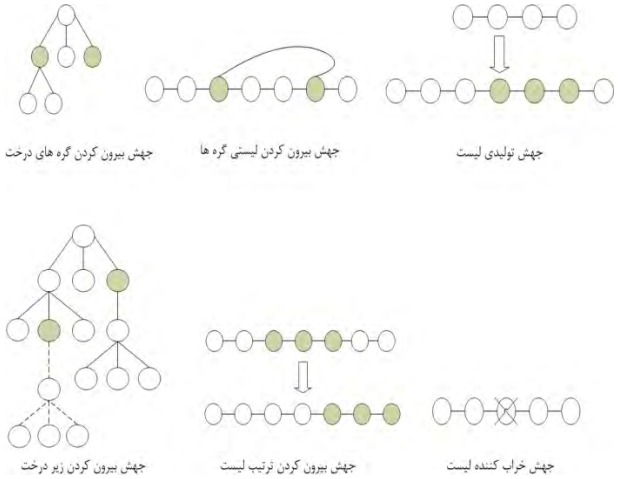
```
1.  MKH (Int m, int n)
 {
 int r;
2.  If (n>m) {
3.  r=m;
4.  m=n;
5.  n=r;
{
6.  While (r !=0 )
{
7.  M=n;
8.  N=r;
9.  r=m % n ;
10.}
11. Return n;
12.
13.}
```

شکل ۵ : نمونه برنامه جهت ایجاد ساختار درختی



شکل ۶ : ساختار درختی نمونه برنامه مثال قبل

یکی از مهمترین عملگرها در الگوریتم ژنتیک عملگر جهش می باشد و همیشه این عملگر با احتمال کمی اجرا می شود. عملگر جهش در فرآیند آزمایش نرم افزار بسیار کاربرد دارد و باعث می شود توسط تابع برارزندگی قسمتهای از نرم افزار آزمایش شوند که الگوریتم ژنتیک مشخص کرده است. در شکل ۷ بعضی از ساختارهای درخت را مشاهده می کنید که عملگر جهش باعث شده آنها را تبدیل به حالت های متفاوتی کند . در واقع با این مثال کاربرد الگوریتم ژنتیک در آزمایش قسمتهای مختلف نرم افزار به خوبی مشخص می شود. به شکل ۷ توجه کنید .



شکل ۷ : مثال های از عملکرد عملگر جهش

۴- مشکلات:

با تکامل استراتژی آزمایش برای زمان طولانی ممکن است تعداد خطاهای بیشتری پیدا شود، به خصوص برای کلاس های با تعداد خطای کم [۷ و ۴] .

مشکلات دیگر به عنوان پرچم و متغیرهای شمارشی وجود دارد همراه با کنترل جریان بدون ساختار است. تحقیقات بیشتری برای غلبه بر این مشکلات مورد نیاز است.

یکی از مشکلات عمده [۱۱، ۸] در تست نرم افزار تولید داده ها به صورت خودکار است که معیار برآورده شدن رضایت می باشد. برای حل این مشکل، بسیاری از این نسخه ها کار تحقیق، که در گذشته انجام شده است وجود دارد. شاید یکی از شایع ترین این مشکلات مواجه شدن با تولید داده های تصادفی تست ، تولید مجموعه داده های نمادین (یا مسیر گرا) بر اساس الگوریتم ژنتیک [۱۲، ۱۰] باشد.

۵- نتیجه گیری

در این مقاله یک رویکرد الگوریتم ژنتیک معرفی شده است که برای تست استفاده نرم افزار است و برای کشف فضایی از داده های ورودی و شناسایی و تمرکز بر روی مناطقی که منجر به شکست است مورد استفاده قرار گیرد. تجزیه و تحلیل مثال ها در این مقاله نشان می دهد که الگوریتم ژنتیک را می توان به عنوان یک ابزار برای کمک به نرم افزار جستجو تستر، مکان یابی، و جداسازی خرابی در یک سیستم نرم افزار تست شده استفاده کرد. استفاده از الگوریتم های ژنتیکی تست خودکار را پشتیبانی می کند و به شناسایی شکست ها که اغلب شدید و به احتمال زیاد برای کاربر اتفاق می افتد کمک می کند.

مراجع :

[1] Goldberg, D.E.,“Genetic Algorithms in Search of Optimisation”, 1989.

[2] Alan C. Schultz, John J. Grefenstette, aid Kenneth A. De Jong,“Test And Evaluation by Genetic Algorithms”, IEEE,1993.

[3] Jefferson Offutt, W. Michael Craft,“Using Compiler Optimisation Techniques to Detect Equivalent Mutants”, The Journal of Software Testing, Verification and Reliability, 4(3), pp. 131–154, 1994.

[4] Painton, L., Campbell, J.,“Genetic algorithms in Optimisation of system reliability”, IEEE Trans. Reliab. 44 (2), pp. 172– 178, 1995.

[5] Michalewicz, Z.,“Genetic Algorithms, Data Structures Evolution Programs”, Verlag, Heidelberg, Berlin, Third Revised and Extended Edition, 1999.

[6] Roy P Pargas, Mary Jean Harrold, Robert R Peck,“Test Data Generation Using Genetic Algorithms”, Journal of Software Testing, Verification and Reliability,1999.

[7] W. Caarls.,“Genetic algorithm visualisation. Master's thesis, Faculty of Science”,University of Amsterdam, The Netherlands, September 2002.

[8] S. Yang.,“Statistics-based adaptive non-uniform crossover for crossover for genetic algorithms”, In J. Bullinaria, editor, Proceedings of the 2002 UK Workshop on Computatiorial lntelligence (UKCI-02), pp. 201-208, 2002.

[9] P. McMinn,“Search-Based Software Test Data Generation: A Survey”, Software Testing, Verification and Reliability, vol. 14, No. 2, pp. 105-156, 2004.

[10] Phil. McMinn,“Search-Based Software Test Data Generation: A Survey”, Software Testing, Verification and Reliability, Vol. 14, No. 3, pp. 212-223, 2004.

[11] M. R. Girgis,“Automatic Test Data Generation for Data Flow Testing Using a Genetic Algorithm”, Journal of Universal Computer Science, vol. 11, No. 6, pp. 898-915, 2005.

[12] B. Antonia,"Software Testing Research: Achievements, Challenges, Dreams", in 2007 Future of Software Engineering: IEEE Computer Society, 2007.

[13] H. Shahriar,"Mutation-based testing of buffer overflows, SQL injections, and format string bugs", Ph.D. dissertation, University of Queen, Aug. 2008.

[14] H. Shahriar, M. Zulkernine,"MUSIC: mutation-based SQL injection vulnerability checking", in Proceedings of the 8th International Conference on Quality Software (QSIC '08), Aug. 2008, pp. 77–86.

[15] Yong Chen, Yong Zhong, Tingting Shi, Jingyong Liu, "Comparison of Two Fitness Functions for GA-based Path- Oriented Test Data Generation", 2009 Fifth International Conference on Natural Computation, IEEE, 2009.

[16] M. Singh, S. Mishra,"Mutant generation for aspect oriented programs," Indian Journal of Computer Science and Engineering, Vol. 1, No. 4, pp. 409–415, 2010.