



۱۳۰۷

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده مهندسی برق و کامپیوتر

پایان نامه دوره کارشناسی ارشد مهندسی کامپیوتر – هوش مصنوعی

حل تحلیلی معادلات دیفرانسیل با مشتقات جزئی بیضوی به کمک برنامه نویسی ژنتیک

کارتزین

توسط:

محمد عبدالمهی

استاد راهنما:

دکتر مهدی علیاری شوره دلی

بهمن ۱۳۹۱

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

## تأییدیه هیات داوران

(برای پایان نامه)

اعضای هیئت داوران، نسخه نهائی پایان نامه خانم / آقای:

را با عنوان:

از نظر فرم و محتوی بررسی نموده و پذیرش آن را برای تکمیل درجه کارشناسی / کارشناسی ارشد تأیید می کند.

امضاء	رتبه علمی	نام و نام خانوادگی	اعضای هیئت داوران
	استادیار	دکتر مهدی علیاری شوره دلی	۱- استاد راهنما
			۲- استاد مشاور
			۳- استاد مشاور
	استاد	دکتر محمد تشنه لب	۴- استاد ممتحن
	دانشیار	دکتر میرمحسن پدram	۵- استاد ممتحن
	استاد	دکتر محمد تشنه لب	۶- نماینده تحصیلات تکمیلی

تقدیم به پدرم

کوهی استوار و حامی من در طول تمام زندگی

تقدیم به مادرم

سنگ صبوری که الفبای زندگی به من آموخت

تقدیم به همسرم

که در سایه همیاری و همدلی او به این منظور نائل شدم .

تقدیم به دلبندم

امید بخش جانم که آسایش او آرامش من است

## تشر و قدر دانی

به مصداق «**من لم یشکر المخلوق لم یشکر الخالق**» بسی شایسته است از استاد

فرهیخته و فرزانه جناب آقای دکتر مهدی علیاری شوره دلی

که با کرامتی چون خورشید ، سرزمین دل را روشنی بخشیدند و گلشن سرای علم و

دانش را با راهنمایی های کار ساز و سازنده بارور ساختند ؛ تقدیر و تشر نمایم.

« **و یزکیهم و یعلمهم الكتاب و الحکمہ.** »

معلمانا مقامت ز عرش برتر باد همیشه توسن اندیشه ات مظفر باد

به نکته های دلاویز و گفته های بلند صحیفه های سخن از تو علم پرور باد

## چکیده

در این نوشتار روشی نوین برای حل معادلات دیفرانسیل با مشتقات جزئی بیضوی به صورت تحلیلی ارائه شده است. روشهای عددی گوناگونی برای حل این دسته از معادلات موجود است، اما ایجاد روشی هوشمند که قابلیت تولید پاسخ های تحلیلی دارد، کمک خواهد کرد که از مزایای تولید راه حل های تحلیلی نیز بهره مند شد. علت اهمیت بالای اینگونه معادلات را می توان در کاربردهای فیزیکی آنها دانست. برای حل این معادلات از برنامه نویسی ژنتیک کارتزین استفاده خواهد شد که در آن عملگر برش نیز افزوده شده تا بتواند پاسخ های تحلیلی مناسب تری را تولید نماید. نوع پیشین این نوع برنامه نویسی، برنامه نویسی ژنتیک است که از ساختار درخت برای تکامل استفاده می نمود. در مقابل ساختار درختی، برنامه نویسی ژنتیک کارتزین از ساختار گرافی بهره می برد. علاوه بر ساختار گرافی، قابلیت منحصربه فرد خنثایی<sup>۱</sup> نیز در این برنامه نویسی وجود دارد که موجب شده تا بتوانیم از مزایای این نوع برنامه نویسی که در مقالات گوناگونی ارائه شده بهره مند شویم. بدین وسیله انتظار می رود که پاسخ های بهتری از لحاظ دقت نسبت به برنامه نویسی ژنتیک معمولی تولید نماید. در نتیجه می توان دامنه بزرگی از مسایل که از معادلات دیفرانسیل با مشتقات جزئی بیضوی پیروی می کنند را مورد خطاب قرار داده و حل تحلیلی برای آنها پیشنهاد کرد.

**کلید واژه:** معادلات دیفرانسیل با مشتقات جزئی بیضوی، برنامه نویسی ژنتیک کارتزین، خاصیت خنثایی، حل تحلیلی، عملگر برش.

---

<sup>۱</sup> Neutrality

فصل ۱- مقدمه .....	۱
۱-۱- مروری بر تاریخچه .....	۱
۲-۱- هدف از انجام تحقیق .....	۱
۳-۱- نوآوری تحقیق .....	۲
۴-۱- ساختار گزارش .....	۲
فصل ۲- معرفی برنامه نویسی ژنتیک .....	۳
۱-۲- برنامه نویسی ژنتیک .....	۳
۲-۲- مقایسه برنامه نویسی ژنتیک و الگوریتم ژنتیک .....	۴
۳-۲- ویژگی های برنامه نویسی ژنتیک .....	۵
۴-۲- مراحل آماده سازی الگوریتم برنامه نویسی ژنتیک .....	۵
۱-۴-۲- مجموعه ترمینالها و مجموعه توابع پایه .....	۷
۲-۴-۲- پارامترهای الگوریتم برنامه نویسی ژنتیک .....	۷
۵-۲- باز نمایی .....	۹
۶-۲- عملگرهای برنامه ریزی ژنتیک .....	۱۰
۲-۶-۱- باز تولید .....	۱۰
۲-۶-۲- عملگر ترکیب .....	۱۰
۲-۶-۳- عملگر جهش .....	۱۲
فصل ۳- معرفی برنامه نویسی ژنتیک کارتزین .....	۱۳
۱-۳- برنامه ریزی ژنتیکی مبتنی بر گراف .....	۱۳
۲-۳- برنامه ریزی ژنتیکی کارتزین .....	۱۴
۱-۲-۳- خاستگاه CGP .....	۱۴
۳-۲-۲- فرم کلی CGP .....	۱۴
۳-۲-۳- محدودیت های مقداری ژنی .....	۱۷

- ۱۷.....مثالهایی از کاربردها.....۴-۲-۳
- ۲۲.....کدگشایی فضای ژنی CGP.....۳-۲-۵
- ۲۶.....تکامل فضای ژنی CGP.....۶-۲-۳
- ۲۷.....افزونگی ژنی درون فضای ژنی CGP.....۷-۲-۳
- ۳۰.....تنظیمات پارامتری برای CGP.....۸-۲-۳
- ۳۱.....برنامه نویسی ژنتیک کارترین حلقوی.....۹-۲-۳

#### فصل ۴ - معادلات دیفرانسیل با مشتقات جزئی بیضوی.....۳۳

- ۳۳.....۴-۱- معادلات دیفرانسیل با مشتقات جزئی.....۳۳
- ۳۴.....۲-۴ انواع کاربردها.....۳۴
- ۳۵.....۳-۴ انواع معادلات دیفرانسیل با مشتقات جزئی.....۳۵
- ۳۶.....۴-۳-۱- روش اجزاء محدود یا روش المانهای محدود.....۳۶
- ۳۶.....۴-۳-۲- روش تفاضل محدود.....۳۶
- ۳۷.....۳-۳-۴ روش احجام محدود.....۳۷

#### فصل ۵ - حل تحلیلی معادلات دیفرانسیل با مشتقات جزئی بیضوی.....۳۸

- ۳۸.....۱-۵ تعریف معادلات به منظور تکامل یافتن.....۳۸
- ۳۹.....۲-۵ حل تحلیلی معادلات دیفرانسیل با مشتقات جزئی.....۳۹
- ۴۰.....۱-۲-۵ محاسبه پاسخ مساله با بهره گیری از برنامه نویسی ژنتیک.....۴۰
- ۴۲.....۲-۲-۵ محاسبه پاسخ مساله با بهره گیری از برنامه نویسی ژنتیک کارترین.....۴۲
- ۴۵.....۳-۵ تغییر بازنمایی CGP برای افزودن عملگر برش کارا.....۴۵
- ۴۷.....۵-۳-۱- تلاش برای ایجاد عملگر برش کارا.....۴۷
- ۵۴.....۲-۳-۵ بازنگری در چگونگی ایجاد برش کارا.....۵۴
- ۶۵.....۴-۵ حل تحلیلی معادلات دیفرانسیل با مشتقات جزئی با استفاده از عملگر برش کارا.....۶۵
- ۶۷.....۱-۴-۵ مساله اول معادلات دیفرانسیل با مشتقات جزئی بیضوی با شرایط مرزی.....۶۷
- ۶۸.....۲-۴-۵ مساله دوم معادلات دیفرانسیل با مشتقات جزئی بیضوی با مرز دایره‌ای.....۶۸



فصل ۶- نتیجه گیری و پیشنهادات.....۷۱

۶-۱- نتیجه گیری .....۷۱

۶-۲- پیشنهادات .....۷۱

ضمیمه : الگوریتم تفاضل تکاملی DE.....۷۲

واژه نامه فارسی به انگلیسی.....۷۸

واژه نامه انگلیسی به فارسی.....۷۹

- جدول ۱-۲: مقادیر پیشنهادی برای  $X$  در مکانیزم *Over – Selection* ..... ۹
- جدول ۱-۳: توابع در پردازش تصویر [۱۴] ..... ۲۱
- جدول ۱-۵: تنظیمات مربوط به *GP* ..... ۴۱
- جدول ۲-۵: تنظیمات مربوط به *CGP* ..... ۴۳
- جدول ۳-۵: کدگذاری توابع در *CGP* ..... ۴۵
- جدول ۴-۵: کدگذاری ورودی ها در *CGP* ..... ۴۵
- جدول ۵-۵: میانگین تعداد نسلها برای همگرایی و تلاشهای محاسباتی در معادله  $x^6 - 2x^4 + x^2$  ..... ۶۴
- جدول ۶-۵: میانگین تعداد نسلها برای همگرایی و تلاشهای محاسباتی در معادله  $x^5 - 2x^3 + x$  ..... ۶۵
- جدول ۷-۵: تنظیمات *CGP* استفاده شده در روند تکامل ..... ۶۶
- جدول ۸-۵: مقایسه عملکرد *GP* و *CGP* ..... ۷۰

- شکل ۲-۱: مثالی از یک درخت تجزیه [۹] ..... ۴
- شکل ۲-۲: ورودی های برنامه نویسی ژنتیک [۹] ..... ۵
- شکل ۲-۳: ساختار کلی برنامه نویسی ژنتیک [۹] ..... ۶
- شکل ۲-۴: درخت عبارت منطقی [۹] ..... ۹
- شکل ۲-۵: نمایشگر عملگر ترکیب [۳۱] ..... ۱۱
- شکل ۲-۶: نمایشگر عملگر جهش [۳۱] ..... ۱۲
- شکل ۳-۱: فرم کلی  $CGP$  [۱۴] ..... ۱۵
- شکل ۳-۲: فضای ژنی  $CGP$  و فضای راهحل مرتبط با یک مدار ضرب کننده دو بیتی [۱۴] ..... ۱۸
- شکل ۳-۳: نمایه فضای راهحل مرتبط با آن برای مجموعه‌های از چهار معادله ریاضی [۱۴] ..... ۱۹
- شکل ۳-۴: نمایه فضای راهحل برای برنامه ای که یک تصویر را تعریف میکند [۱۴] ..... ۲۰
- شکل ۳-۵: تصویر تولیدی فضای ژنی تکامل یافته کدگذاری و اجرا میشود [۱۴] ..... ۲۲
- شکل ۳-۶: روند کدگشایی برای فضای ژنی  $CGP$  برای مساله ضرب کننده دوبیتی ..... ۲۴
- شکل ۳-۷: نمونه ای از عملگر جهش نقطه ای قبل و بعد از اعمال آن روی فضای ژنی  $CGP$  [۱۴] ..... ۲۹
- شکل ۳-۸: خنثایی اجازه داده شده و در دیگری بدون خنثایی اجراها صورت گرفته است [۱۴] ..... ۳۱
- شکل ۵-۱: پاسخ دقیق برای  $ue_{x,y} = e - x(x + y^3)$  ..... ۴۰
- شکل ۵-۲: تولید جمعیت اولیه ..... ۴۶
- شکل ۵-۳: ایجاد پاسخ دقیق ..... ۴۷
- شکل ۵-۴: انتخاب نقطه برش تصادفی ..... ۴۸
- شکل ۵-۵: نمودار انتخاب نقطه برش تصادفی ..... ۴۸
- شکل ۵-۶: نقطه برش تصادفی بین گره ای ..... ۴۹
- شکل ۵-۷: نمودار نقطه برش تصادفی بین گره ای ..... ۵۰
- شکل ۵-۸: انتخاب یک گره تصادفی و جابه جایی آن ..... ۵۱

- شکل ۵-۹: نمودار انتخاب یک گره تصادفی و جابه‌جایی آن ..... ۵۲
- شکل ۵-۱۰: گرفتن مقدار صحیح هر فرزند به صورت تصادفی از والد ها ..... ۵۳
- شکل ۵-۱۱: نمودار گرفتن مقدار صحیح هر فرزند به صورت تصادفی از والد ها ..... ۵۴
- شکل ۵-۱۲: نمودار کارایی برش در  $GA$  ..... ۵۵
- شکل ۵-۱۳: تعریف برش کارا در  $CGP$  با تغییر بازنمایی آن ..... ۵۶
- شکل ۵-۱۴: تخصیص اعداد اعشاری به توابع ..... ۵۷
- شکل ۵-۱۵: تخصیص اعداد اعشاری به ورودی ها در  $CGP$  ..... ۵۸
- شکل ۵-۱۶: نحوه عملکرد برش کارا در  $CGP$  ..... ۵۹
- شکل ۵-۱۷: فرآیند کدگشایی بین فضاها ی ژنی اعداد اعشاری و اعداد صحیح [۱۴] ..... ۶۰
- شکل ۵-۱۸: مقایسه  $CGP$  اعداد صحیح و  $CGP$  اعداد اعشاری بدون عملگر برش ..... ۶۰
- شکل ۵-۱۹: همگرایی میانگین برای  $x^2 - 2x^4 + x^6$  ..... ۶۲
- شکل ۵-۲۰: همگرایی میانگین برای  $x - 2x^3 + x^5$  ..... ۶۲
- شکل ۵-۲۱: همگرایی میانگین برای  $x^2 - 2x^4 + x^6$  ..... ۶۳
- شکل ۵-۲۲: همگرایی میانگین برای  $x - 2x^3 + x^5$  ..... ۶۴
- شکل ۵-۲۳: راه حل دقیق  $ue(x, y) = e - x(x + y^3)$  ..... ۶۷
- شکل ۵-۲۴: راه حل دقیق  $ue(x, y) = (x^2 + y^2 - 1)sin(x)$  ..... ۶۹

- روند ۱-۳: پردازش گره ها در  $CGP$  [۱۴] ..... ۲۵
- روند ۲-۳: کدگشایی  $CGP$  [۱۴] ..... ۲۵
- روند ۳-۳: محاسبه تطابق  $CGP$  [۱۴] ..... ۲۶
- روند ۴-۳: استراتژی تکاملی (۱+۴) [۱۴] ..... ۲۸

## فصل ۱- مقدمه

### ۱-۱- مروری بر تاریخچه

مدل سازی ریاضی بسیاری از پدیده ها و سیستم های علوم مهندسی و فیزیکی، موجب برجسته شدن نقش و اهمیت معادلات دیفرانسیل با مشتقات جزئی گشته اند. شماری از تکنیک هایی که ساختار مناسبی برای حل این معادلات دارند، توسط دانشمندان ابداع شده اند که شامل روش های تفاضل محدود<sup>۱</sup> [۱]، المان محدود<sup>۲</sup> [۲]، احجام محدود<sup>۳</sup> [۳] و المان مرزی<sup>۴</sup> [۴] می باشند. این تکنیک ها به حل عددی معادلات دیفرانسیل با مشتقات جزئی پرداخته اند. برای سالیان زیادی محققان در جستجوی الگوهای زیستی که الهام گرفته از سیستم های بیولوژیکی می باشند، به دنبال یافتن روش هایی بودند که بتوانند به برنامه نویسی ژنتیک<sup>۵</sup> سرعت ببخشند و کارایی آن را افزایش دهند تا بدین وسیله بتوانند مسایل بزرگتر و پیچیده تری را به صورت تحلیلی حل کنند. تکنیکی که برنامه نویسی ژنتیک کارتزین<sup>۶</sup> [۵] نامیده می شود و در آن به جای استفاده از درخت، مسایل به صورت گراف ارایه می شود، این امید را ایجاد کرد که بتوان راه حل های بهتری را برای حل این معادلات تولید نماید. علاوه بر نمایش مسایل به صورت گرافی و ویژگی های منحصر بفردی که این نوع برنامه نویسی دارد، موجب شده که تلاش خود را معطوف به این موضوع کنیم که مطالعات دقیق تری روی آن انجام دهیم تا بتوان مسایل پیچیده تری را حل نماییم.

### ۱-۲- هدف از انجام تحقیق

به دلیل اهمیت بالای معادلات دیفرانسیل در رشته های مهندسی مختلف و همچنین مطابق بودن با برخی از فرآیندهای طبیعت، حل آنها از اهمیت بالایی برخوردار است. علت اهمیت بالای اینگونه معادلات را می توان در کاربردهای فیزیکی آنها دانست، همچون جریانات گردابی اطراف آبشکن ها<sup>۷</sup> در مهندسی عمران و تلاش در مهار آب های تندرو که همگی از معادلات دیفرانسیل با مشتقات جزئی بیضوی پیروی

---

<sup>۱</sup> Finite Difference Methods

<sup>۲</sup> Finite Element Methods

<sup>۳</sup> Finite Volume Methods

<sup>۴</sup> Boundary Element Methods

<sup>۵</sup> Genetic Programming

<sup>۶</sup> Cartesian Genetic Programming

<sup>۷</sup> Vortex Flows Around Spur-Dike

می‌کنند [۶]. معادلات دیفرانسیل توصیف‌کننده حرکت سیارات، که از قانون دوم نیوتن بدست می‌آیند، هم شامل شتاب و هم شامل سرعت می‌شوند. در مورد حرکت موشک‌ها در نزدیکی سطح زمین و در فضا، معادلات دیفرانسیل پیچیده‌ترند. مسایل فیزیکی زیادی بعد از فرمول‌بندی آنها به زبان ریاضی به معادلات دیفرانسیل منجر می‌شوند. در رشته مهندسی شیمی، معادلات دیفرانسیل نقش منحصر به فردی به عهده دارند. همین‌طور در مواردی چون سود مرکب، واپاشی رادیواکتیو - قانون سرمایه‌گذاری نیوتن و رشد جمعیت کاربرد فراوانی دارد. همه موارد ذکر شده بسیاری از محققان را مجاب ساخته تا به دنبال راه‌حل مناسبی برای اینگونه مسایل باشند. غالب راه‌حل‌های ارائه شده، عددی می‌باشد که همین امر اهمیت تولید روشی به منظور حل تحلیلی معادلات دیفرانسیل با مشتقات جزئی را افزایش می‌دهد. از جمله مزایای راه‌حل‌های تحلیلی می‌توان به قابلیت مشتق‌گیری، تفسیر پذیری و استنباط رفتار سیستم‌های تبعیت‌کننده از معادلات دیفرانسیل با مشتقات جزئی بیضوی اشاره نمود.

### ۱-۳- نوآوری تحقیق

در این تحقیق از برنامه نویسی ژنتیک کارت‌زین که به آن روش جدیدی برای اعمال عملگر برش کار [۷] ارائه شده به حل معادلات دیفرانسیل با مشتقات جزئی بیضوی می‌پردازد. این نوع معادلات در غالب راه‌حل‌ها به صورت عددی حل می‌شدند، اما در تلاش انجام شده در این نوشتار به صورت تحلیلی و با دقت بالایی نتایج بسیار مطلوبی بدست آمده‌اند.

### ۱-۴- ساختار گزارش

در نوشتار ارائه شده که در شش فصل تنظیم شده است، مطالب بدین‌گونه بیان خواهند شد. فصل اول مقدمه، فصل دوم معرفی برنامه نویسی ژنتیک و نحوه عملکرد آن، فصل سوم معرفی برنامه نویسی ژنتیک کارت‌زین، فصل چهارم معادلات دیفرانسیل با مشتقات جزئی و انواع راه‌حل‌های موجود برای حل آنها، فصل پنجم حل تحلیلی معادلات دیفرانسیل با مشتقات جزئی توسط برنامه نویسی ژنتیک و برنامه نویسی ژنتیک کارت‌زین و در نهایت فصل ششم که به بررسی نتایج بدست آمده و همچنین پیشنهاداتی برای کارهای آتی که در این زمینه مناسب می‌باشد، خواهد پرداخت.

## فصل ۲- معرفی برنامه نویسی ژنتیک

### ۲-۱- برنامه نویسی ژنتیک

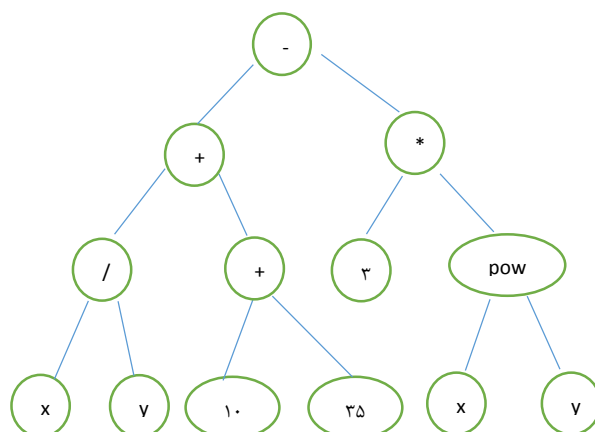
برنامه نویسی ژنتیکی یکی از جوان‌ترین الگوریتم‌های تکاملی است که در دهه ۱۹۹۰ در آمریکا معرفی شد. در برنامه نویسی ژنتیک سعی می‌کنیم که با استفاده از الگوریتم‌های ژنتیک، و مفاهیم درختهای تجزیه برای کاربردهای خاص، به جای اینکه کد برنامه لازم را بنویسیم، به کامپیوتر این امکان را بدهیم که تنها با دانستن مفهوم کلی از کار، برنامه مورد نظر را آماده کند. در واقع یک دستور سطح بالا به کامپیوتر بدهیم و خود کامپیوتر برنامه لازم برای اجرای برنامه مورد نظر را آماده کند، سپس برنامه را اجرا و خروجی مطلوب را ارائه دهد. [۷]

در واقع یکی از نیازمندی‌های اساسی در تولید برنامه‌ها آن است که بایستی از لحاظ دستوری، درستی برنامه‌های نوشته شده تأیید گردند. به همین دلیل کامپایلرها سعی می‌کنند که برنامه نوشته شده را به یک فرم یکنوا و ساده‌تر که درختهای تجزیه نام دارند، تبدیل کنند.

در جریان کامپایل یک برنامه سطح بالا (مثل  $C++/C$ )، کد نوشته شده، به یک درخت تجزیه تبدیل می‌شود، تا کامپایلر راحت‌تر بتواند بر روی آن کار کند. از طرف دیگر در صورتی که در برنامه نویسی ژنتیک از این ساختار استفاده کنیم، هم تعریف عملگرهای ژنتیک بر روی آن ساده‌تر است و هم دردسر خطایابی کدها و اصلاح خطاهای دستوری را نخواهیم داشت. زیرا در صورتی که کامپایلر بتواند از روی برنامه نوشته شده، با موفقیت درخت تجزیه را ایجاد کند، برنامه نوشته شده از لحاظ دستوری خطایی ندارد.

به عنوان مثال برای عبارت  $(x/y + (10 + 3^2)) - (2^{pow(x,y)})$  درخت تجزیه به صورت شکل ۲-۱ است.  $GP$  می‌تواند به منظور پیدا کردن برنامه‌ای که قابلیت حل مساله مورد نظر را داشته باشد و یا تا حد امکان بهتر از بقیه به حل مساله بپردازد، فضای برنامه را جستجو کند. برنامه‌ای که توسط  $GP$  بدست می‌آید در رقابتی براساس شایستگی انتخاب می‌شود. برای این منظور صدها یا هزاران برنامه کامپیوتری با اندازه و شکل‌های مختلف که به صورت تصادفی خلق شدند با هم به رقابت پرداخته و این برنامه‌ها بر اساس فرضیه بهترین‌ها (داروین) و انجام اعمال ژنتیکی بهبود و توسعه می‌یابند.





شکل ۲-۱: مثالی از یک درخت تجزیه [۹]

## ۲-۲- مقایسه برنامه نویسی ژنتیک و الگوریتم ژنتیک<sup>۱</sup>

برنامه ریزی ژنتیک از خانواده الگوریتمهای ژنتیک است که از لحاظ ساختاری شباهت زیادی به الگوریتمهای ژنتیک دارد. با این وجود بین این دو روش تفاوت‌هایی نیز موجود می باشد از جمله:

- هدف *GA* جستجو و بهینه‌سازی می باشد، درحالی که هدف اصلی *GP* برنامه نویسی خودکار است.
- ساختار کروموزوم‌ها در *GA* خطی است ولی در حالیکه ساختار کروموزوم‌های *GP*، غیر خطی و درختی است.
- ژنها در *GA* می توانند انواع مختلفی داشته باشند در حالی که در *GP* دارای دو نوه خاص می باشند.
- ژنها در *GA* می توانند انواع مختلفی داشته باشند در حالی که در *GP* دارای دو نوع خاص می باشند.
- طول کروموزوم‌ها در *GA* می تواند ثابت یا متغیر باشد درحالی که طور کروموزوم در *GP* استاندارد اغلب متغیر است.
- نحوه عملکرد عملگرها در *GA* و *GP* بسته به ساختار کروموزوم‌ها متفاوت است.

<sup>۱</sup> Genetic Algorithm

- جهت ایجاد جمعیت جدید در  $GA$  غالباً از روش خاصی استفاده نمی‌کنیم در حالی که برای ایجاد جمعیت اولیه در  $GP$  از روشهای خاصی استفاده می‌نماییم.

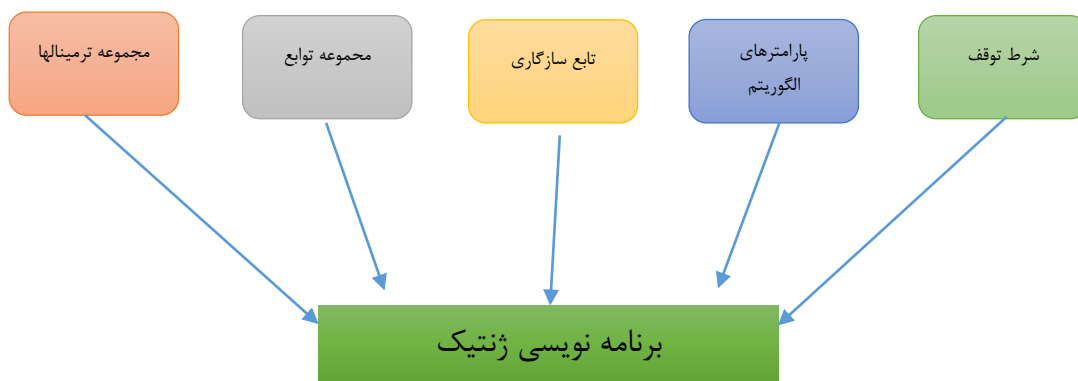
## ۲-۳- ویژگی‌های برنامه نویسی ژنتیک

$GP$  همچون شبکه‌های عصبی قادر به تولید مدل‌هایی برای حل مساله می‌باشد و از این منظر قابل رقابت با شبکه‌های عصبی است، ولی مزیتی که نسبت به شبکه‌های عصبی دارد این است که شبکه‌های عصبی ممکن است در مینیمم محلی گیرکنند ولی در  $GP$  چون از تکامل استفاده می‌شود و تکامل قادر است از مینیمم محلی فرار کند، در دام مینیمم محلی نمی‌افتد [۸].

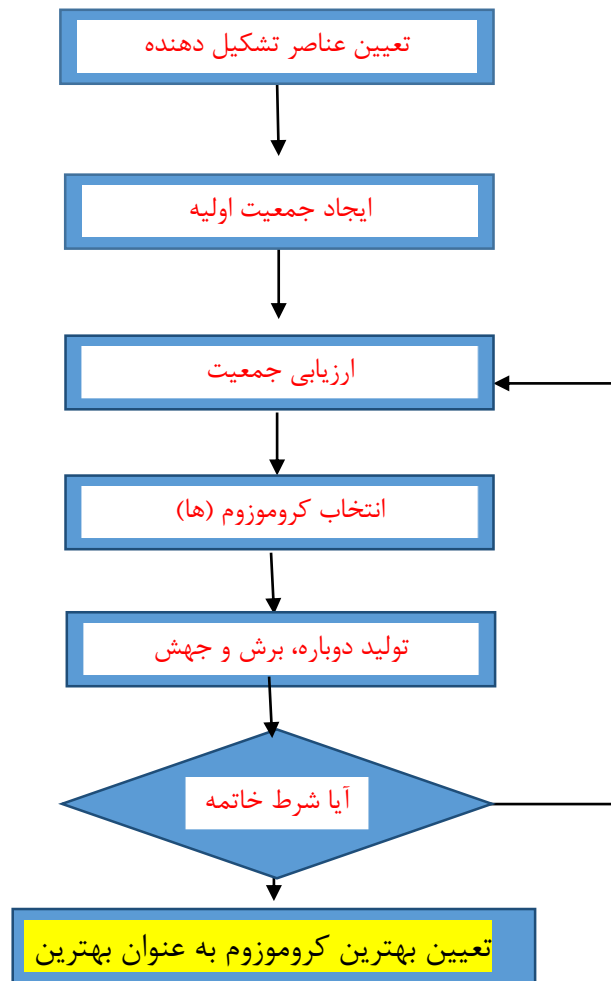
یکی از معایب  $GP$  آن است که نیاز به جمعیت‌های با اندازه خیلی بزرگ دارد (هزاران یا حتی میلیون). این الگوریتم‌ها به طور کلی خیلی کند هستند به دلیل اینکه باید با درخت‌ها کار کنند و همان‌طور که قبلاً گفته شد اندازه جمعیت هم خیلی بزرگ است. اما این کند بودن خیلی چالش‌انگیز نیست چون فقط یک بار الگوریتم  $GP$  انجام می‌شود و پس از آنکه در پایان اجرا، مدلی برای حل مساله حاصل شد از این مدل می‌توان به صورت بهنگام استفاده کرد؛ در واقع هزینه استفاده از این مدل خیلی کم است اما هزینه بدست آوردن مدل باری اولین بار ممکن است بسیار بالا باشد و حتی ماه‌ها یا سال‌ها ممکن است اجرای یک الگوریتم  $GP$  روی کامپیوترهای قوی طول بکشد.

## ۲-۴- مراحل آماده سازی الگوریتم برنامه نویسی ژنتیک

همانطور که در شکل ۲-۲ دیده می‌شود، برنامه ریزی ژنتیک دارای پنج ورودی می‌باشد. به منظور انجام روند تکامل نیز در شکل ۲-۳ فلوچارت این برنامه نویسی آورده شده است.



شکل ۲-۲: ورودی‌های برنامه نویسی ژنتیک [۹]



شکل ۲-۳: ساختار کلی برنامه نویسی ژنتیک [۹]

برای اینکه بتوانیم با استفاده از برنامه نویسی ژنتیک یک برنامه برای یک مساله سطح بالا تهیه کنیم، بایستی با انجام یک سری اعمال، برنامه سطح بالا را به فرم مناسب الگوریتم ژنتیک تبدیل کنیم. پنج قدم اساسی زیر را باید در ابتدا به وسیله یک عامل انسانی، انجام دهیم:

- ۱) یک مجموعه از عناصر پایانی تعیین کنیم:
- عناصر پایانی، متغیرهای مستقل مساله، توابع بدون پارامتر ورودی یا ثابت هایی هستند که به صورت تصادفی ایجاد شده‌اند.
- ۲) مجموعه توابع اولیه که بایستی برنامه بر پایه آنها تولید شود.
- ۳) معیار شایستگی (برای این که به وسیله آن بتوانیم برنامه‌های تولید شده به وسیله *GA* را ارزیابی و برنامه‌های بهتر را شناسایی کنیم)

- ۴) پارامترهایی که برای کنترل اجرای *GA* مورد نیاز هستند، به عنوان مثال نوع انتخاب والدین و انتخاب فرزندان، نرخ جهش، نرخ بازترکیبی و ...
- ۵) شرط خاتمه و روشی برای اینکه نتیجه اجرا را بتوانیم تعیین کنیم.

## ۲-۴-۱- مجموعه ترمینالها و مجموعه توابع پایه

همان طور که در قسمتهای قبلی بیان شد، ساختار سازگار با برنامه نویسی ژنتیک، ساختار مرتبه ای می باشد که اندازه آن در طول اجرای برنامه متفاوت و قابل تغییر است. مجموعه ساختارهای موجود که با سیستم برنامه-نویسی ژنتیک سازگار باشند، شامل دو مجموعه است که اعضای جمعیت موجود با آنها ساخته می شوند:

۱. مجموعه توابع پایه<sup>۱</sup> که می توانند به صورت بازگشتی با یکدیگر ترکیب شوند.

۲. مجموعه ترمینالها<sup>۲</sup>

ترمینالها در درخت، برگها را تشکیل می دهند، در حالی که توابع گره های داخلی درخت هستند و به تعداد آرگومانهای خود فرزند می گیرند. هر یک از توابع موجود در مجموعه توابع می توانند تعداد مشخصی آرگومان را به عنوان ورودی دریافت کنند. بسته به نوع مساله، توابع می توانند عملگرهای استاندارد ریاضی مانند جمع، تفریق، ... و یا توابع استاندارد ریاضی مانند *sin*, *cos*, *exp* و... و یا عملگرهای منطقی مانند *If-Then-Else* باشند. مجموعه ترمینالها نیز می توانند متغیرها یا مقادیر ثابت باشند.

به طور کلی کلیه زبانهای برنامه نویسی قادر به پیاده سازی برنامه ریزی ژنتیک می باشند. اما به طور معمول برای پیاده سازی این روش از زبان *LISP* استفاده می شود.

## ۲-۴-۲- پارامترهای الگوریتم برنامه نویسی ژنتیک

### ۲-۴-۲-۱- ایجاد جمعیت اولیه

اولین مرحله از اجرای برنامه ریزی ژنتیک، ایجاد یک جمعیت اولیه می باشد. در واقع این جمعیت اولیه شامل اعضای است که به طور تصادفی تولید شده اند. در ابتدا اعضای جمعیت اولیه به صورت تصادفی و با اندازه های متفاوت ایجاد می شوند. مساله ای که در اینجا باید به آن توجه داشت این است که با توجه به الگوریتم بالا، ممکن است اندازه درخت ایجاد شده بسیار بزرگ شود. بنابراین می توانیم یک شرط را نیز برای این الگوریتم قرار دهیم که اندازه درخت بدست آمده از یک حد معین تجاوز نکند. به عنوان مثال

---

<sup>۱</sup> . Function Set

<sup>۲</sup> . Terminal Set