۱۳۰۷
دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشکده مهندسی برق و کامپیوتر

پایان‌نامه دوره کارشناسی ارشد مهندسی کامپیوتر–هوش مصنوعی

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

توسط:

محمد عبدالهی

استاد راهنما:

دکتر مهدی علیاری شوره دلی

بهمن ۱۳۹۱

تأییدیّه 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 می‌کند.

اعضای هیئت داوران	نام و نام خانوادگی	رتبه علمی	امضاء
۱ – استاد راهنما	دکتر مهدی علیاری شوره دلی	استادیار	
۲ – استاد مشاور			
۳ – استاد مشاور			
٤ – استاد ممتحن	دکتر محمد تشنه لب	استاد	
٥ – استاد ممتحن	دکتر میرمحسن پدرام	دانشیار	
٦ – نماینده تحصیلات تکمیلی	دکتر محمد تشنه لب	استاد	

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

تشکر و قدردانی

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 تشکر نمایم.

« و یزکیهم و یعلمهم الکتاب و الحکمه. »

معلما 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 ارایه شـده اسـت. روشـهای عددی گوناگونی برای حل این دسـته از معادلات موجود اسـت، اما ایجاد روشـی هوشـمند که قابلیت تولید پاسـخ های تحلیلی دارد، کمک خواهد کرد که از مزایای تولید راه حل های تحلیلی نیز بهره‌منـد شـد. علـت اهمیـت بـالای 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 مناسب‌تری را تولید نماید. نوع پیشین این نوع برنامه نویسی، برنامه نویسـی ژنتیک اسـت که از ساختار درخت برای تکامل استفاده می‌نمود. در مقابل ساختار درختی، برنامه نویسـی ژنتیک کارتزین از سـاختار گرافی بهره می‌برد. علاوه‌بر ساختار گرافی، قابلیت منحصربه‌فرد خنثایی[1] نیز در این برنامه نویسی وجود دارد که موجب شده تا بتوانیم از مزایای این نوع برنامه نویسی که در مقالات گوناگونی ارایه شـده بهره‌مند شـویم. بدین‌وسـیله انتظار می‌رود که پاسخ های بهتری از لحاظ دقت نسـبت به برنامه نویسی ژنتیک معمولی تولید نماید. درنتیجه می توان دامنه بزرگی از مسایل که از معادلات دیفرانسیل با مشتقات جزئی بیضوی پیروی می‌کنند را مورد خطاب قرار داده و حل تحلیلی برای آنها پیشنهاد کرد.

کلید واژه: معادلات دیفرانسـیل با مشـتقات جزئی بیضـوی، برنامه نویسـی ژنتیک کارتزین، خاصـیت خنثایی، حل تحلیلی، عملگر برش.

[1] Neutrality

فصل ۱– مقدمه

۱–۱– مروری بر تاریخچه

مدل سازی ریاضی بسیاری از پدیده ها و سیستم‌های علوم مهندسی و فیزیکی، موجب برجسته شدن نقش و اهمیت معادلات دیفرانسیل با مشتقات جزیی گشته اند. شماری از تکنیکهایی که ساختار مناسبی برای حل این معادلات دارند ، توسط دانشمندان ابداع شده‌اند که شامل روش‌های تفاضل محدود[۱][۱]، المان محدود[۲][۲]، احجام محدود[۳][۳] و المان مرزی[۴][۴] می‌باشند. این تکنیک‌ها به حل عددی معادلات دیفرانسیل با مشتقات جزئی پرداخته‌اند. برای سالیان زیادی محققان در جستجوی الگوهای زیستی که الهام گرفته از سیستم‌های بیولوژیکی می‌باشند، به دنبال یافتن روش‌هایی بودند که بتوانند به برنامه نویسی ژنتیک[۵] سرعت ببخشند و کارایی آن را افزایش دهند تا بدین‌وسیله بتوانند مسایل بزرگتر و پیچیده‌تری را به صورت تحلیلی حل کنند. تکنیکی که برنامه‌نویسی‌ژنتیک‌کارتزین[۶][۵] نامیده می‌شود و در آن به جای استفاده از درخت، مسایل به صورت گراف ارایه می شود، این امید را ایجاد کرد که بتوان راه‌حل‌های بهتری را برای حل این معادلات تولید نماید. علاوه‌بر نمایش مسایل به صورت گرافی ویژگی های منحصربفردی که این نوع برنامه نویسی دارد، موجب شده که تلاش خود را معطوف به این موضوع کنیم که مطالعات دقیق‌تری روی آن انجام دهیم تا بتوان مسایل پیچیده‌تری 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 برخودار است. علت اهمیت بالای اینگونه معادلات را می‌توان در کاربردهای فیزیکی آنها دانست، همچون جریانات گردابی اطراف آبشکن‌ها[۷] در مهندسی عمران و تلاش در مهار آب های تندرو که همگی از معادلات دیفرانسیل با مشتقات جزئی بیضوی پیروی

---

[۱] Finite Difference Methods

[۲] Finite Element Methods

[۳] Finite Volume Methods

[۴] Boundary Element Methods

[۵] Gentic Programming

[۶] Cartesian Genetic Programming

[۷] 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 منحصربه‌فردی به عهده دارند. همین‌طور در مواردی چون سود مرکب، واپاشی رادیواکتیو - قانون سرمایش 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 ارایه شده، عددی می‌باشد که همین امر اهمیت تولید روشی به منظور حل تحلیلی معادلات دیفرانسیل با مشتقات جزئی را افزایش می‌دهد. از جمله مزایای راه‌حل های تحلیلی می‌توان به قابلیت مشتق گیری، تفسیر پذیری و استنباط رفتار سیستم‌های تبعیت کننده 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 کارا[۷] ارایه شده به حل معادلات دیفرانسیل با مشتقات جزئی بیضوی می پردازد. این نوع معادلات در غالب راه حل ها 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 ارایه 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 راه‌حل های 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 می باشد، خواهد پرداخت.

فصل ۲–معرفی برنامه نویسی ژنتیک

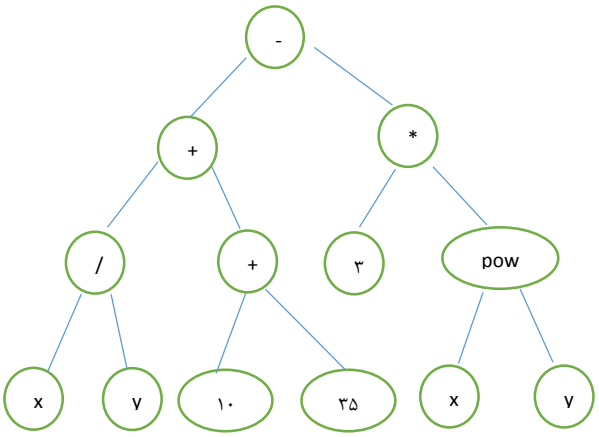
۲–۱– برنامه نویسی ژنتیک

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 الگوریتم های‌ژنتیک، 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 ارایه دهد.[۷]

در واقع یکی از نیازمندی‌های اساسی در تولید برنامه‌ها آن است که بایستی از لحاظ دستوری، درستی برنامه های نوشته شده تائید گردند. به همین دلیل کامپایلرها سعی می کنند 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 می شود، 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 $(x/y+(۱۰+۳۲))-(۳(pow(x,y)))$ 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. برنامه ای که توسط GP بدست می آید در رقابتی براساس شایستگی انتخاب می شود. برای این منظور صدها یا هزاران برنامه کامپیوتری با اندازه و شکل های 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[۱]

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 داشـــته باشـــند در حالی‌که در *GP* دارای دو نوه خاص می‌باشند.
- ژنها در *GA* می‌توانند انواع مختلفی داشـــته باشـــند در حالی‌که در *GP* دارای دو نوع خاص می‌باشند.
- طول کروموزوم‌هـا در *GA* می‌توانـد ثـابت یا متغیر باشـــد درحالی‌که طور کروموزوم در *GP* استاندارد اغلب متغیر است.
- نحوه عملکرد عملگرها در *GA* و *GP* بسته به ساختار کروموزوم‌ها متفاوت است.

[۱] Genetic Algorithm

- جهت ایجاد جمعیت جدید در *GA* غالبا از روش خاصی استفاده نمی‌کنیم در حالی‌که برای ایجاد جمعیت اولیه در *GP* از روشهای خاصی استفاده می نماییم.

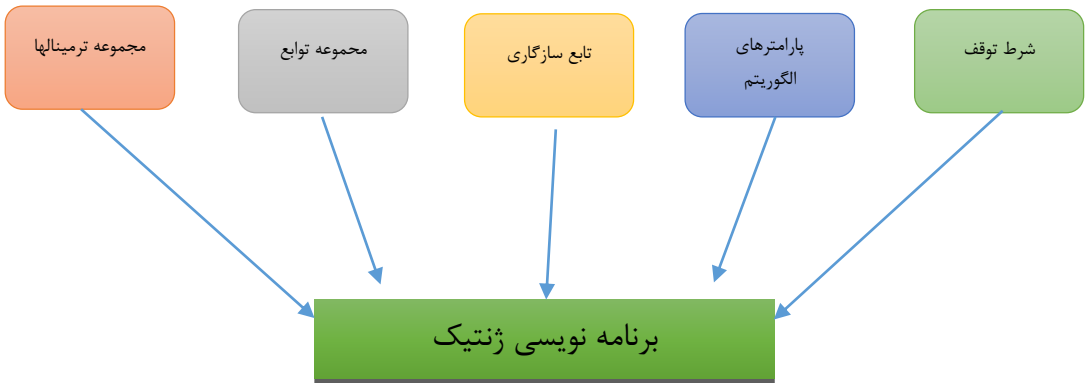
۲-۳- ویژگی های برنامه نویسی ژنتیک

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 می شود و تکامل قادر است از مینیمم محلی فرار کند، در دام مینیمم محلی نمی افتد[۸].

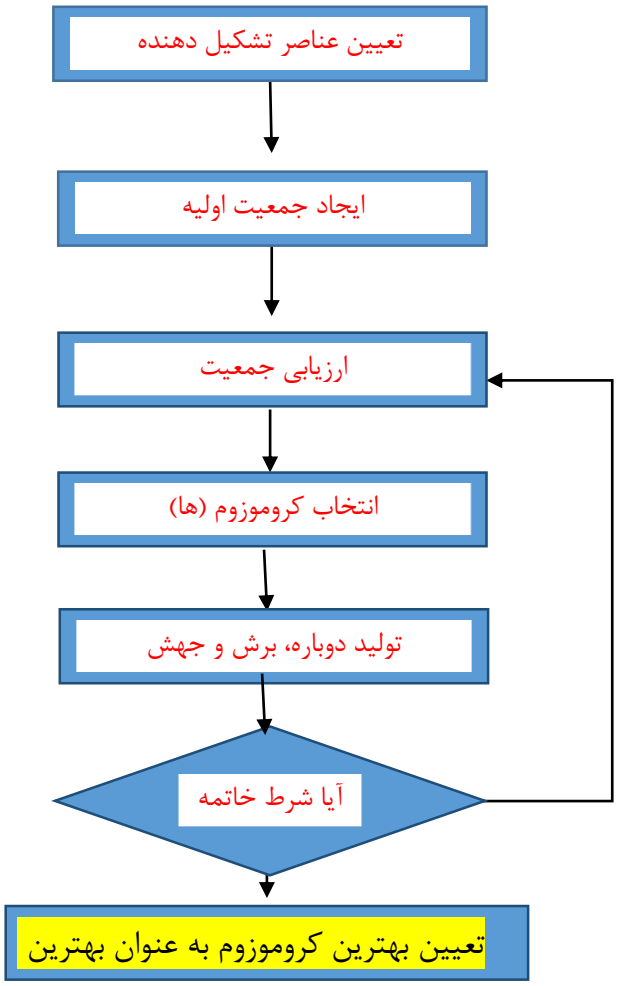
یکی از معایب *GP* آن است که نیاز به جمعیت‌های با اندازه خیلی بزرگ دارد (هزاران یا حتی میلیون). این الگوریتم‌ها به طور کلی خیلی کند هستند به‌دلیل اینکه باید با درخت‌ها کار کنند و همان‌طور که قبلا گفته شد اندازه جمعیت هم خیلی بزرگ است. اما این کند بودن خیلی چالش انگیز نیست چون فقط یک بار الگوریتم *GP* انجام می‌شود و پس‌از آنکه درپایان 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 ماه ها یا سال ها ممکن است اجرای یک الگوریتم *GP* روی کامپیوتر های 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 می باشد. به منظور انجام روند تکامل نیز در شکل ۲-۳ فلوچارت این برنامه نویسی آورده شده است.



شکل ۲-۲: ورودی های 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 هســـتند، 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 می‌باشد 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[۱] که می توانند به صورت بازگشتی با یکدیگر ترکیب شوند.
۲. مجموعه ترمینالها[۲]

ترمینالها در درخت، برگ‌ها را تشـــکیل می‌دهند، در حالی‌که توابع گره‌های داخلی درخت هســـتند و به تعداد آرگومانهای خود فرزند می‌گیرند. هر یک از توابع موجود در مجموعه توابع می‌توانند تعداد مشخصی آرگومان را به عنوان ورودی دریافت کنند. بسته به نوع مساله، توابع می‌توانند 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 می‌توانند متغیرها یا مقادیر ثابت باشند.

به طورکلی کلیه زبانهای برنامه‌نویسی قادر به پیاده‌سازی برنامه ریزی ژنتیک می‌باشند. اما به طور معمول برای پیاده‌سازی این روش از زبان *LISP* استفاده می شود.

۲-۴-۲- پارامترهای الگوریتم برنامه نویسی ژنتیک

۲-۴-۲-۱- ایجاد جمعیت اولیه

اولین مرحلـه از اجرای برنامه‌ریزی ژنتیک، ایجاد یک جمعیت اولیه می‌باشـــد. در واقع این جمعیت اولیه شامل اعضایی است که به طور تصادفی تولید شده‌اند. در ابتدا اعضای جمعیت اولیه به صورت تصادفی و با اندازه های متفاوت ایجاد می‌شـوند. مسـاله‌ای که در اینجا باید به آن توجه داشت این‌است که با توجه به الگوریتم بالا، ممکن اسـت اندازه درخت ایجاد شده بسیار بزرگ شود. بنابراین می توانیم یک شرط را نیز برای این الگوریتم قرار دهیم که اندازه درخت بدســـت آمده از یک حد معین تجاوز نکند. به عنوان مثال

---

۱. Function Set

۲. Terminal Set