بنام خدا

گزارش پروژه Genetic Programming

برای توابعی که دارای دو عملوند و یا یک عملوند هستند، تعدادی توابع ثابت مانند add, pow, sin, cos در نظر گرفته شده است . همچنین برای اعداد ثابت عدد ،اعداد بین 1 تا 20 را در نظر گرفتیم.(از در نظر گرفتن اعداد بزرگتر به این دلیل که با اعداد قبلی ساخته می شوند صرف نظر شده است.)

برای اینکه Terminal ها شامل عدد pi و متغیرx باشد با یک احتمال این انتخاب توسط یک عدد رندوم انتخاب میشود.

هر کروموزوم به صورت یک درخت از Node هایی که دارای حالت Terminal یا function تشکیل شده است که با پیمایش in-order این درخت حاصل نهایی عبارت ریاضی مذکور بدست میاید.

شایستگی هر کروموزم به صورت مینیمم بودن MSE آن معین میشود. هر چه میزان MSE یک کروموزم کمتر باشد، شایسته تر است.

روند طی شده توسط الگوریتم کلی شامل مراحل زیر است :

1- ایجاد جمعیت اولیه :
الگوریتم یک میزان برای تعداد کروموزم ها در جمعیت اولیه قبول میکند که دارای 3 حالت , SMALL , MEDIUM LARGE میباشد که با توجه به پیچیدگی مسئله میتوانند انتخاب شوند. کروموزوم های اولیه دارای یک عمق مشخص هستند که در ورودی خود مقدارش را میگیرند (پیش فرض 3) .
درخت ایجاد شده در نهایت به احتمال 50 درصد متوازن است و همچنین ممکن است قبل از رسیدن به ماکسیمم عمق متوقف شود. امکان این رویداد کاملا رندوم است.
پس از ایجاد جمعیت اولیه مقدار Fitness تمامی انها حساب میشود و اگر در این حین exception رخ دهد آن فرد از جمعیت حذف میشود. برای مثال ممکن است از یک جمعیت 100 نفره حدودا 40 نفر حذف شوند چون تابع تولید شده توسط انها قابل اجرا نبوده است.
2- نحوه انتخاب والدین :
الگوریتم انتخاب والدین roulette wheel میباشد که بر حسب fitness هر فرد یک احتمال انتخاب به ان میدهد . در هر بار اجرای این الگوریتم یک والد انتخاب میشود. (این عمل n/2 جمعیت اولیه انجام میشود تا سایز جمعیت تغییر نکند.)
3- تولید نسل بعد :
پس از انتخاب 2 والد، عملیات cross over انجام میشود و اگر بدون ارور پایان یابد، دو فرزند تولید شده با یک احتمال دارای جهش میشوند؛ به این صورت که هرچه در نسل ها جلوتر میرویم احتمال جهش کمتر میشود و به صورت کسری از generation_number/iteration_count به عنوان ورودی داده میشود، اگر عدد رندوم تولید شده از این مقدار بیشتر باشد، فرزندها دارای جهش میشوند در غیر این صورت خیر. پس از این دو مرحله مقدار fitness دو فرزند محاسبه میشود و از بین 4 نفر خانواده (شامل 2 والد و 2 فزرند) ، دو فرد شایسته تر انتخاب میشوند. یعنی ممکن است فرزندان به نسل بعد نروند.
4- برای خاتمه الگوریتم تعداد iteration به ان داده میشود اما چک میشود که اگر از نسل 6 به بعد دارای یک fitness مورد قبول باشد دیگر اجرا نشود و خاتمه یابد . (برای مثال کمتر از 0.0001)

یکی از چالش هایی که تایم خوبی از من گرفت، مقابله با مقدار None بود که به سبب ارور های محاسباتی در درخت ایجاد میشد و همچنین علت اصلی آن randomness بودن که ویژگی اصلی برنامه است، محسوب میشود . این مقدار در مراحل مختلف catch شده است تا در اجرای برنامه خلالی ایجاد نشود .

تعداد 4 تابع نسبتا ساده آزمایش شده است و اطلاعات مربوطه به کاربر نشان داده میشود .

به علت محدودیت زمانی موفق به پیاده سازی توابع با ورودی 2 بعدی نشدم :(

از طرفی چون رندوم بودن جزئی از ذات اصلی برنامه است ، کد اجرایی مربوط به تست به گونه ای زده شده که ارور ها را بگیرد و تا زمانی که به نتیجه مطلوب نرسیده است به اجرای برنامه ادامه دهد تا به fitness خواسته شده برسد 😊