پروژه دوم – الگوریتم ژنتیک

نازنین یوسفیان 810197610

هدف

الگوریتم ژنتیک در فضایی که تعداد حالات ما زیاد هستند برتری بیشتری نسبت به الگوریتم های سرچ کلاسیک دارند به این علت که در زمان و حافظه صرفه جویی می کنند. در این پروژه سعی داریم که با انتخاب معیار های درست، به جوابی برسیم که نسل به نسل بهبود می یابد و تنها روی چند فرد از نسل تاثیر نگذارد.

مقدمه

در این پروژه یک فایل csv. که حاوی جدول درستی یک مدار است به ما داده می شود. با استفاده از الگوریتم ژنتیک سعی داریم که مدار متناسب با آن را پیدا کنیم. گیت هایی که در این مدار استفاده می کنیم گیت های AND، NAND، OR، NOR، XOR و XNOR هستند.

توضیح پروژه

مرحله اول

در ابتدا مفاهیم ژن و کروموزوم را در این پروژه تعریف می کنیم. به هر کدام از گیت هایی که می توانیم استفاده کنیم یک عدد نسبت می دهیم. این گونه به طور مثال گیت AND عدد 0 می گیرد. ژن ها در واقع گیت های ما هستند و یک کروموزوم که مجموعه ای از ژن هاست توالی گیت ها برای رسیدن به جواب را به ما نشان می دهد. تعداد ژن های کروموزم بسته به تعداد گیت هایی است که در آن مدار باید استفاده شود. به طور مثال در مداری با 10 ورودی از 9 گیت استفاده می کنیم پس کروموزوم های ما دارای 9 ژن خواهند بود. (هر ژن می تواند 6 مقدار از 0 تا 5 بگیرد.)

مرحله دوم

برای شروع جمعیت اولیه ای را به صورت رندوم انتخاب می کنیم. باید توجه شود که در مراحل پیش رو باید تنوع جمعیت حفظ شود. پس اگر تعداد جمعیت اولیه کم باشد، پس از چند مرحله کروموزوم هایی که تولید می شوند تقریبا یکسان شده و این باعث می شود که نتوانیم جلوتر برویم و به جواب مسئله برسیم. هم چنین جمعیت زیاد باعث کاهش سرعت الگوریتم می شود و هزینه زمانی بیشتر می شود. جمعیت 300-200 نفر برای این مسئله مناسب است.

مرحله سوم

در مرحله بعد باید معیاری داشته باشیم که بتوانیم بر اساس آن از بین این جمعیت افرادی را برای زاد و ولد انتخاب کنیم. این انتظار را داریم که اگر والدان فرزندی معیار خوبی داشته باشند، از ترکیب آن ها فرزند بهتری تولید می شود. پس افرادی را انتخاب می کنیم که در جمعیت کنونی ما معیار بهتری دارند. معیار را در این مسئله اینگونه تعریف می کنیم که هر فرد به چند حالت جدول ورودی پاسخ درست می دهد و مجموع آن را برای هر فرد محاسبه می کنیم. برای انتخاب از بین جمعیت فعلی برای زاد و ولد می توانیم از دو روش roulette wheel و rank-based استفاده کنیم. در روش roulette wheel هر چقدر fitness فرد بیشتر باشد، احتمال انتخاب آن بیشتر است و درواقع وزن بیشتری دارد. در روش rank-based درواقع افراد را نسبت به هم می سنجیم و به هرکدام یک rank اختصاص می دهیم. این روش هزینه زمانی sort را نیز دارد که در مقایسه با عملیات محاسبه fitness قابل چشم پوشی است. هم چنین بر اساس این انتخاب، زودتر به جواب می رسیم و نسل های کمتری تولید می کنیم.

مرحله چهارم

پس از انتخاب افرادی که قرار است زاد و ولد انجام دهند، لازم است که فرزاندان جدید را تولید کنیم. ابتدا از crossover استفاده می کنیم. در این عملیات ابتدا پدران را به صورت رندوم می چینیم. سپس در هر مرحله دوتا از آن ها را انتخاب می کنیم و با احتمال p_c عملیات crossover را روی آن ها انجام می دهیم. این احتمال را عددی بالا و تقریبا نزدیک به 1 (در اینجا 0.8) در نظر می گیریم. زیرا این انتظار را داریم که ترکیب دو پدر خوب فرزند بهتری تولید کند. در اینجا از 1-point crossover استفاده کردیم زیرا عوض کردن هر گیت تغییر زیادی در جواب مسئله می دهد و همین تک نقطه برای رسیدن به جواب کافی است.

پس از اینکه فرزندان ساخته شدند، عملیات mutation را روی آن ها انجام می دهیم به این صورت که هرکدام با احتمال p_m، mutate می شوند و یک ژن آن ها به صوت رندوم تغییر کرده و مقدار جدیدی می گیرد. با توجه به اینکه ژن های ما در این مسیله گیت ها هستند و تغییر هرکدام باعث تفاوت زیادی در مدار می شود، فقط یک ژن را عوض می کنیم. احتمال p_m را در ابتدا بیشتر در نظر می گیریم و سپس به مراحل جلوتر که می رسیم آن را کمتر می کنیم. در ابتدا ما یک سری افراد را به صورت رندوم انتخاب کرده ایم و خیلی به جواب مسئله نزدیک نیستند پس این احتمال را برایشان بیشتر در نظر می گیریم تا جواب های جدید تر را نیز ببینیم که شاید بهتر باشند ولی در مراحل جلوتر تقریبا به جواب های نزدیکی به جواب موردنظر رسیده ایم و خیلی نمی خواهیم تغییر کنند. برای اینکه جمعیت ما یکنواخت نشود و حالت های جدیدتری نیز تولید شوند همچنان mutation را انجام می دهیم ولی با احتمال پایین تر. پس احتمال اولیه را برابر 1/chromosome_length در نظر می گیریم و پس از هر 10 نسلی که تولید کردیم احتمال را 0.9 برابر می کنیم تا به مقدار 1/population_size برسد.

جواب مسئله

تولید نسل ها را آنقدر ادامه می دهیم تا در نسلی به فردی برسیم که جواب مورد نظر ما را دارد. در اینجا جواب پیدا کردن فردی است که مقدار fitness برای او 1024 باشد و تمام حالات مسئله را پاسخ بدهد.

بهبود راه حل

در حین اجرای الگوریتم ممکن است به حالاتی برسیم که در نقطه local maximum گیر کنیم و دیگر نتوانیم حالتی را پیدا کنیم که به جواب بهتری برسیم. این حالت معمولا وقتی اتفاق می افتد که از حالاتی که داریم نتوانیم حالت بهتری بسازیم. براس رفع این مشکل باید بتوانیم تنوع جمعیت را زیاد کنیم. یکی از روش ها می تواند این باشد که احتمال mutation و تعداد نقاط آن را بیشتر کنیم. پس می توانیم پس از چند جمعیت که تولید کردیم اگر عدد تغییر نکرد این احتمال را بیشتر کنیم. اینگونه جواب های ما بیشتر تغییر می کنند و می توانیم حالت های جدیدتری داشته باشیم. روش دیگر این است که پس از اینکه از بین جمعیت افرادی را برای زاد و ولد انتخاب کردیم، چند فرد با ژن های رندوم به آن ها اضافه کنیم. پس از پایان عملیات crossover به تعداد افرادی که اضافه کردیم از فرزندان آن هایی که fitness کمتری دارند را حذف کنیم. روش دوم بیشتر به تنوع جمعیت کمک می کند و سریع تر می توانیم از حالتی که در آن گیر کرده ایم رها شویم.

نتیجه گیری

در فضاهایی که تعداد حالات زیاد است و جمعیت زیادی داریم، اینکه تمامی حالات را بررسی کنیم برای ما مقدور نیست و هزینه زیادی هم از نظر زمان و هم حافظه برای ما دارد. در اینجا از الگوریتم ژنتیک استفاده می کنیم و با مجموعه ای از جواب ها شروع می کنیم و در هر مرحله سعی می کنیم این جواب ها را بهبود داده و به جواب اصلی مسئله نزدیک شویم. پارامتر هایی که در این الگوریتم استفاده می کنیم تاثیر چشمگیری در رفتار الگوریتم ما دارند و در انتخاب آن ها باید بسیار دقت شود. مهم ترین بخش پیدا کردن معیار سازگاری است که افراد را با آن بسنجیم. این الگوریتم ممکن است همیشه ما را به جواب بهینه نرساند ولی می توانیم جواب های نزدیک به جواب بهینه را در زمان کمتری پیدا کنیم.

منابع

1. *Russell, Stuart J. (2018). Artificial intelligence a modern approach. Norvig, Peter (4th ed.)*

2. https://towardsdatascience.com/genetic-algorithm-implementation-in-python-5ab67bb124a6