به نام خدا

هوش مصنوعی

تمرین کامپیوتری شماره 2 ، بخش : Genetic

امیرحسین کهربائیان – 810199478

شرح پروژه :

در این قسمت باید مسئله Equation Problem را به کمک الگوریتم ژنتیک حل کنیم. در واقع با داشتن لیستی از عملوند ها و عملگر ها باید بتوانیم عبارت ریاضی را تولید کنیم که برابر با یک عدد مشخص شود. مثلا اگر عملوند های ما 2 ، 5 ، 7 و همچنین عملگر ها + ، * باشد و عدد هدف 17 باشد، باید با کمک الگوریتم ژنتیک عبارتی تولید کنیم که حاصلش برابر 17 است.

به عنوان مثال می توان عبارت (2+3*5) یا (3*5+2) را در نظر گرفت.

روش کلی حل مسئله :

همانطور که گفته شد این مسئله را باید به کمک الگوریتم ژنتیک حل کنیم. به طور خلاصه روش کلی به شرح زیر می باشد :

1- ابتدا جمعیت اولیه را می سازیم. جمعیت شامل تعدادی chromosome است که هر کروموزوم از تعدادی ژن تشکیل شده است. در این مسئله هر ژن یکی از عملگر ها یا عملوند ها می باشد و هر کروموزوم را که می تواند یک پاسخ برای مسئله باشد، یک عبارت ریاضی فرض می کنیم. در ساخت جمعیت اولیه ژن ها را به طور رندوم انتخاب کرده و با آن ها یک کروموزوم می سازیم و به جمعیت اضافه می کنیم.

2- به هر کروموزوم عددی به نام fitness را نسبت می دهیم که میزان نزدیک بودن آن کروموزوم به جواب اصلی است. در واقع اگر fitness یک کروموزوم برابر با 1 باشد، آن کروموزم جواب اصلی است و هر چه به 0 نزدیک تر باشد به معنای دوری از جواب اصلی می باشد. بنابراین fitness معیار خوبی برای ارزیابی کروموزوم ها است . تابع $E(i)$ را به این صورت تعریف می کنیم :

$$E(i) = |eval(i) - goalNumber|$$

i : کروموزوم i ام از جمعیت

$eval(i)$: مقدار محاسبه شده عبارت ریاضی مربوط به کروموزم i

$goalNimber$: عددی که مقدار ریاضی کروموزم باید برابر با آن شود

به عنوان مثال اگر $i : 2 + 3 * 7$ و عدد هدف برابر 20 باشد : $E(2 + 3 * 7) = |23 - 20| = 3$

حال می توانیم تابع $F(i)$ را به شکل زیر تعریف کنیم :

$$fitness(i) = F(i) = \frac{1}{E(i) + 1}$$

واضح است که مقدار F برای کروموزوم جواب برابر یک است. دقت شود 1 در مخرج برای صفر نشدن مخرج است. برای مثال اگر $i : 5 * 3 + 2$ و $goalNumber = 17$ آنگاه : $(eval(i) = 15 + 2 = 17)$

$$E(i) = |17 - 17| = 0 \Rightarrow F(i) = \frac{1}{0 + 1} \Rightarrow F(i) = 1 \Rightarrow chromosome\ i\ is\ result$$

3- پس از ساخت جمعیت اولیه و محاسبه fitness برای اعضای آن، ابتدا تعدادی از کروموزوم هایی که دارای fitness بالایی هستند(نسبت به بقیه کروموزوم ها به جواب نزدیک تر هستند) را انتخاب کرده تا مستقیما به جمعیت بعدی انتقال دهیم. درصد تعداد کروموزوم های انتخابی برای انتقال مستقیم به جمعیت بعد را از طریق متغیر P_carry کنترل می کنیم. حال عملیات crossover و mutation را به این شکل انجام می دهیم:

CROSSOVER : برای انجام این کار ابتدا لیستی از کروموزوم ها را به عنوان پدر درست کرده و سپس از طریق کروموزوم های پدر، فرزندان را با استفاده از crossover تک نقطه ای می سازیم. نرخ ایجاد کروموزوم های جدید را نیز با کمک متغیر P_crossover کنترل می کنیم. در این بین برخی کروموزوم ها نیز مستقیم به لیست کروموزوم های جدید اضافه میشوند و در نهایت از لیست آن هایی به نسل بعد منتقل می شوند که بیشترین fitness را دارند.

MUTATION : در مرحله قبل لیستی از کروموزوم های کاندید برای انتقال به نسل بعد توسط عملیات crossover ساختیم، حال قبل از انتقال آنان به نسل بعد روی کروموزوم ها عملیات mutation نیز انجام می دهیم. این عملیات که نرخش با متغیر P_mutation کنترل می شود، برخی از ژن های کروموزوم را تغییر می دهد تا هم یک نواختی بین نسل ها مدیریت شود و هم این که به جواب نزدیک تر شویم. البته P_mutation مقدار زیادی ندارد و برخی از کروموزوم ها بعد از این عملیات بدون تغییر می مانند.

4- ایجاد نسل های جدید از جمعیت آنقدر تکرار می شود تا بالاخره به نسلی برسیم که در آن کروموزوم جواب وجود داشته باشد.

سوالات (بخش شش) :

1– جمعیت اولیه بسیار زیاد باعث طولانی شدن محاسبات لازم برای تولید نسل بعد می شود. اگر جمعیت بسیار زیاد باشد، برای ساختن نسل بعد زمان بسیار زیادی صرف می شود و در یک نسل زمان بیش از اندازه ای را صرف می کنیم و نمی توانیم در زمان معقول با تولید نسل های مختلف به جواب نزدیک تر شویم.

اگر جمعیت بسیار کم باشد نیز اعضای کمی برای تولید نسل بعدی داریم و کروموزوم های متنوعی نداریم که بتوانیم اعضای جدید و متنوعی بسازیم که به جواب نزدیک شویم. بنابراین اگر جمعیت اولیه بسیار کم باشد تنوع کمی از کروموزوم ها داریم و در نسل های بعد نیز تنوع لازم میان کروموزوم ها وجود ندارد و ممکن است به جواب نرسیم یا بعد از زمان بسیار زیادی طی تولید نسل های مختلف به پاسخ برسیم.

2- با صعودی شدن تعداد جمعیت در هر دوره، سرعت الگوریتم برای محاسبات لازم مانند fitness و همچنین تولید نسل بعد بسیار کاهش می یابد و در هر نسل زمان خیلی زیادی متوقف می شویم. همچنین با تولید کروموزوم های دور از جواب عملکرد نسل ممکن است کاهش یابد و در نسل بعدی به جواب به میزان لازم نزدیک نشویم.

3- در عملیات crossover با ترکیب دو کروموزوم خوب سعی بر تولید کروموزوم بهتر و نزدیک تر به جواب داریم. با کمک این الگوریتم کروموزوم های نزدیک تر به جواب را می سازیم و نسل بعد را بهبود می دهیم. در عملیات mutation نیز با جهش کروموزوم مواجه می شویم و یکی از تاثیرات موثر این عملیات، کنترل یکنواختی میان نسل ها است و از یکنواخت شدن نسل ها و شبیه بودن نسل های به هم تا حدی جلوگیری می کند و می تواند کروموزوم هایی نزدیک تر به جواب تولید کند. با توجه به خصوصیات ذکر شده برای هر عملیات، برای تولید نسل های بهبود یافته و نزدیک تر به جواب باید از هر دو عملیات استفاده کنیم .

4- انتخاب مناسب تعداد جمعیت ، نرخ عملیات های crossover و mutation و P_carry می تواند تاثیر قابل توجهی در زمان حل مسئله داشته باشد. همچنین انتخاب مناسب کروموزوم های پدر و بهبود هر نسل نسبت به نسل قبلی نیز اهمیت دارد. واضح است که طول کروموزوم و نحوه تعریف آن نیز موثر است.

5- اگر تعداد جمعیت اولیه کم باشد یا به خوبی عملیات های crossover و mutation انجام نشود یا هر نسل نسبت به نسل قبل بهبود نیابد(fitness کلی نسل نسبت به نسل قبل بالاتر نرود)، به مرور در نسل ها ممکن است یکنواختی ایجاد شود و کروموزوم ها تغییر نکنند که در این صورت ممکن است پاسخ یافت نشود یا پس از مدت زمان بسیار طولانی به پاسخ برسیم و الگوریتم به درستی کار نکند.

برای عدم ایجاد یکنواختی و کروموزوم های مشابه، باید موارد زیر را رعایت کرد :

- ایجاد جمعیت اولیه رندوم با تعداد قابل قبول(تعداد جمعیت اولیه نباید خیلی کم باشد).
- مقدار دهی مناسب متغیر های P_crossover، P_carry و P_mutation ، به عنوان مثال اگر نرخ عملیات mutation بیش از اندازه کم باشد، به دلیل عدم جهش کروموزوم ها ممکن است یک نواختی میان نسل ها دیده شود.
- نحوه انجام عملیات crossover اهمیت دارد. ضمن انتخاب درست والدان، باید توجه کنیم که ممکن است انجام این عملیات با دو نقطه برشی و یا بیشتر بهتر باشد نسبت به یک نقطه.

6- می توانیم یک حداقل مقداری برای fitness مشخص کنیم و پس از ایجاد تعداد مشخصی از نسل ها، در صورت عدم یافتن جواب این مقدار را کاهش دهیم. یعنی در ابتدا مقدار fitness یک کروموزوم جواب باید حتما 1 باشد اما در صورتی که پس از گذشت زمان و تولید نسل های مختلف به کروموزوم جواب نرسیدیم می توان مقدار 1 را به 0.8 کاهش دهیم و اگر fitness کروموزومی برابر با 0.8 بود الگوریتم پایان یابد. در غیر این صورت این مقدار باز هم پس از مدتی کاهش یابد تا بدین صورت الگوریتم در یک مرحله به جواب برسد.

توجه شود که میزان fitness هر کروموزوم با توجه به تعاریفی که در بالا ذکر شده حتما مقداری بین صفر تا یک است، بنابراین اگر روش گفته شده را اجرا کنیم الگوریتم حتما پایان پذیر خواهد بود.