پروژه دوم هوش

ژنتیک و بازی

سپهر آزردار 810199357

ژنتیک:

1. جمعیت اولیه ی بسیار کم یا بسیار زیاد چه مشکلاتی را به وجود میآورند؟

اگر جمعیت اولیه خیلی کم باشد. تنوع و گوناگونی رو از دست میدهیم و ممکن است به جواب نرسیم. چون احتمال اینکه از کنار هم قراردادن تکه جواب ها به جواب اصلی برسیم خیلی کم میباشد. اگر هم جمعیت خیلی زیاد باشد. هزینه زمانی و حافظه برای ما دارد. و اینکه باید نسل ها بعد رو جوری به وجود بیاوریم که در mini/max ها محلی نیفتیم. در واقع تنوع و گوناگونی رو حفظ کنیم.

2. اگر تعداد جمعیت در هر دوره افزایش یابد، چه تاثیری روی دقت و سرعت الگوریتم میگذارد؟

با افزایش جمعیت، هزینه زمانی و حافظه برای ما افزایش میابد و در نتیجه سرعت الگوریتم کاهش پیدا میکند. در الگوریتم ژنتیک، همانند طبیعت، ژن های برتر به نسل بعدی منتقل میشود و یا جمعیت کاهش میابد و به یک یا چند کروموزوم میل میکند و یا ثابت میماند. در صورتی که جمعیت افزایش یابد این میل کردن به جواب را ممکن است ما از دست بدهیم. چون طبق یک روند کلی میانگین fitness هر نسل نسبت به نسل قبلی خود بیشتر است.

3. تاثیر هر یک از عملیات های crossover و mutation را بیان و مقایسه کنید. آیا میتوان فقط یکی از آنها را استفاده کرد ؟ چرا ؟

Crossover:

روی دو والد انجام میشود و آنها را باهم ترکیب میکند و دو فرزند تولید میشود . به امید اینکه بچه های آن ها ژن ها خوب والد هایشان را به ارث به ببرن و fitness بالاتری داشته باشن و به جواب نزدیک تر شوند و یا جواب رو تولید کنند.

Mutation:

روی فرزند ها تولید شده اجرا میشود و به طور رندوم یه سری از ژن ها ی کروموزوم رو تغییر میدهد(جهش/تغییر ژنتیکی) به امید اینکه تنوع و گوناگونی حفظ شود و در suboptimal های محلی گیر نکنیم.

به طور کلی احتمال mutation از crossover کمتر است و اولی رو فرزندان نسل و روی یک کروموزوم اجرا میشود ولی دومی رو والد ها نسل قبل و دو یا چند والد رو باهم ترکیب میکند.

به طور کلی باید از هر دو استفاده نمود و برای مثال وقتی جمعیت اولیه کم است و مشکل تنوع و گوناگونی داریم

(در suboptimal های محلی گیر میکنیم. یا اصلا به جواب نمی‌رسیم.) باید نرخ(احتمال) این عملیات رو بالاتر ببریم. و crossover: در صورتی که جمعیت اولیه بالا باشد. استفاده از همین عملیات می تواند کافی باشد. ولی این عملیات رو نمیتوان حذف کرد. زیرا ترکیب کردن کروموزوم ها و رسیدن به جواب به امید کنار هم قرار دادن تکه جواب ها به عهده ی این عملیات است.

4. به نظر شما چه راهکارهایی برای سریعتر به جواب رسیدن در این مسئله ی خاص وجود دارد؟

اگر چند ژن مجاور هم رو در فرایند crossover جا به جا کنیم به جواب سریعتر میرسیم. که علت این امر این است که جواب مسئله از چند بلوک جواب کوچک تر تشکیل شده است و اگر جا این بلوک ها جایشان باهم عوض شوند بازم جواب مسئله برقرار است. پس اگر به صورت بلوکی ما کروموزوم ها رو با همدیگه ترکیب کنیم. میتوانیم سریع تر به جواب برسیم.

5. با وجود استفاده از این روشها، باز هم ممکن است که کروموزومها پس از چند مرحله دیگر تغییر نکنند. دلیل این اتفاق و مشکلاتی که به وجود می آورد را شرح دهید. برای حل آن چه پیشنهادی میدهید؟ (راه حلهای خود را امتحان کنید و بهترین آنها را روی پروژه خود پیاده سازی کنید).

اگر بخشی از جمعیت به طور مستقیم به مرحله بعدی بروند احتمال وقوع همچین چیزی بیشتر است. و دلیل دیگر آن این است که ما با یه احتمالی هر دو والد رو crossover میکنیم و بنابراین بعضی از والد ها بدون هیچ تغییری به مرحله بعدی میروند. میتوان در crossover pool والد ها دو به دو جفت کرد و فرایند رو انجام تا به این شکل مطمئن شویم همه والد ها در این عملیات شرکت میکنند. و باید شافل کنیم و احتمال crossover و mutation رو در گذر زمان تغییر دهیم.

مشکلات: هدف این الگوریتم این است که کروموزوم ها در هر مرحله با همدیگر ترکیب شوند تا نسل بعد fitness بهتری را داشته باشد. ولی با این اتفاق، این فرایند دچار مشکل میشود.

6 .چه راه حلی برای تمام شدن برنامه در صورتی که مسئله جواب نداشته باشد پیشنهاد میدهید؟

میتوان از یه حد بالا استفاده کرد و همانند IDS تا یک جایی عملیات رو ادامه بدهیم مثلا پس از 100 نسل بهترین جواب(fitness) رو به عنوان جواب چاپ کنیم.

بازی:

تعداد	زمان	شانس پیروزی	نودها	عمق
100	0.0025	0.98	54.7 و 52	1
100	0.24 و 0.04	0.98	5100 و 900	3
100	18.3 و 0.63	0.98	394339 و 14500	5
100	11.50	0.95	143231	7

سوال ۱ : یک heuristic خوب چه ویژگیهایی دارد؟ علت انتخاب heuristic شما و دلیل برتری آن نسبت به تعدادی از روشهای دیگر را بیان کنید.

سرعت محاسبه اش بالا است. و در مدت زمان کم برای نا تخمین میزند و بدین ترتیب سرعت الگوریتم نیز بالا میرود.

نکته دیگر این است که هرچقدر هیوریستیک در عمق پایین تری کاشته شود، دقتش بیشتر است ولی نود هایی که باید بررسی کنیم بیشتر میشود. هیوریستیکی خوب است که در حالیکه در عمق کم کاشته میشود، همچنان دقتش بالا باشد. تا سرعت الگوریتم حفظ شود.

هیوریستیک انتخابی: score_red – score_blue

Score یک رنگ متناظر با تعداد یال هایی است که میتوان در گراف فعلی انتخاب کرد و رنگ کرد. بدون اینکه مثلثی تشکیل شود.

با فرض اینکه به طور معمول وقتی که تعداد حالت های ممکن برای قرمز بیشتر از آبی باشد، احتمال برنده شدن بیشتر است، این

هیوریستیک انتخاب شده است. بدین ترتیب وقتی قرمز 5 حالت بیشتر از آبی دارد امتیاز 5 و اگر این اختلاف 3 بود امتیاز 3 میگیرد. حالت اول به دومی برتر دارد. با فرض مساوی بودن شرایط به طور کلی برای دو رنک انتظار داریم رنگی که تعداد حالت های بیشتری دارد، شانس بیشتری برای برد داشته باشد.

سوال ۲ :آیا میان عمق الگوریتم و پارامترهای حساب شده روابطی میبینید؟ به طور کامل بررسی کنید که عمق الگوریتم چه تاثیراتی بر روی شانس پیروزی، زمان و گرههای دیده شده میگذارد.

بله. هرچه عمق بیشتر میشود، تعداد گره های دیده شده بیشتر میشود و همین طور زمان. برای سطر اول ما 14/15 انتخاب داریم و برای دفعه بعدی، دویال در گراف رنگ میشود. و تعداد حالات دوتا کاهش میابد. با فرض اینکه آبی اول شروع کند : انتخاب قرمز برای یال اول به ترتیب عمق ها(1و3و5) به : 14و 14*13*12 و 10*11*12*13*14، نود برای بررسی نیاز دارد.

پس زمان عمق دومی تقریبا 80 برابر اولی و زمان عمق سومی 75 برابر دومی میباشد. علت اینکه نسبت اولی(80) برابر با 12*13 = 156 نشد. این است که در انتخاب دوم قرمز دوتا یال دیگر رنگ میشود و قرمز 12 حالت بیشتر ندارد به همین دلیل نسبت کمتر است. و همین طور درخت کامل نیست. و همین طور وابسته به انتخاب شانسی آبی در هر مرحله.

سوال ۳ :وقتی از روش هرس کردن استفاده میکنید، برای هر گره درخت، فرزندانش به چه ترتیبی اضافه میشوند؟ آیا این ترتیب اهمیت دارد؟ چرا این ترتیب را انتخاب کردید؟

ترتیبی انتخاب نکردم.