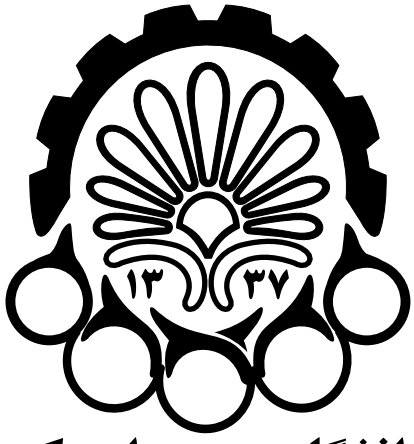


دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

موضوع : حل مسئله N-Queen با روش تکامل تدریجی

نویسنده: شروین غفاری

استاد: دکتر یوسفی مهر

درس: هوش مصنوعی

چکیده :

الگوریتم تکامل تدریجی از طبیعت الهام گرفته شده و برای حل مسائلی به کار میرود که در زمان Polynomial قابل حل نیستند و میتوانند جواب بهینه ای در فضای stochastic برای مسئله پیدا کنند.

همچنین پاسخی که ارائه میدهند برای اکثر مسائل در دنیای واقعی قابل اطمینان و کارآمد است.

الگوریتم ژنتیک (Genetic) نیز نوعی تکامل تدریجی محسوب میشود که رفتار هایی مشابه طبیعت ارائه میدهد به طوری که ابتدا یک جمعیت اولیه را تولید و طی نسل های بعدی تنها فرزندانی از نسل ها که تغییراتی و جهش هایی همسو با طبیعت دارد انتخاب میشوند و نسل های قبلی خود را تکمیل تر میکنند.

هر کدام از individual ها شامل تعدادی chromosome هستند و با ایجاد فرزند جدید بخشی از ژن آنها به نسل بعد منتقل می شود.

یکی از دلایل موفقیت این الگوریتم استفاده از داده های تصادفی است به طوری که در مسیر رسیدن به جواب بهینه در هر نسل بهترین ها را انتخاب میکند همچنین هر چقدر تعداد نسل ها و جمعیت اولیه بیشتر باشد احتمال رسیدن به بهینه ترین جواب ممکن بیشتر میشود.

مقدمه :

مسئله n وزیر از آن دسته از مسائلی است که در n های بزرگ در مدت های exponential حل میشود که بهینه نیست.
مسئله یک صفحه شطرنجی n*n است که شامل n وزیر است و باید به شکلی در صفحه قرار بگیرند که هیچ دو وزیری یکدیگر را تحدید نکنند.

هر وزیر در هر خانه که قرار بگیرد به شکل سطری و ستونی و قطری صفحه را پوشش میدهد به شکلی که نباید وزیر دیگری در این مکان ها قرار داشته باشد.

برای حل این مسئله از الگوریتم ژنتیک استفاده میشود و در تکرار هر نسل سعی میشود تا بهترین چینش برای وزیر ها که از طریق crossover و mutation,selection ایجاد شده انتخاب شود.

در نهایت individual که بهترین تغییرات را در هر نسل داشته به عنوان جواب بهینه معرفی میشود.
همچنین برای هر n تعداد جوابها محدود است.

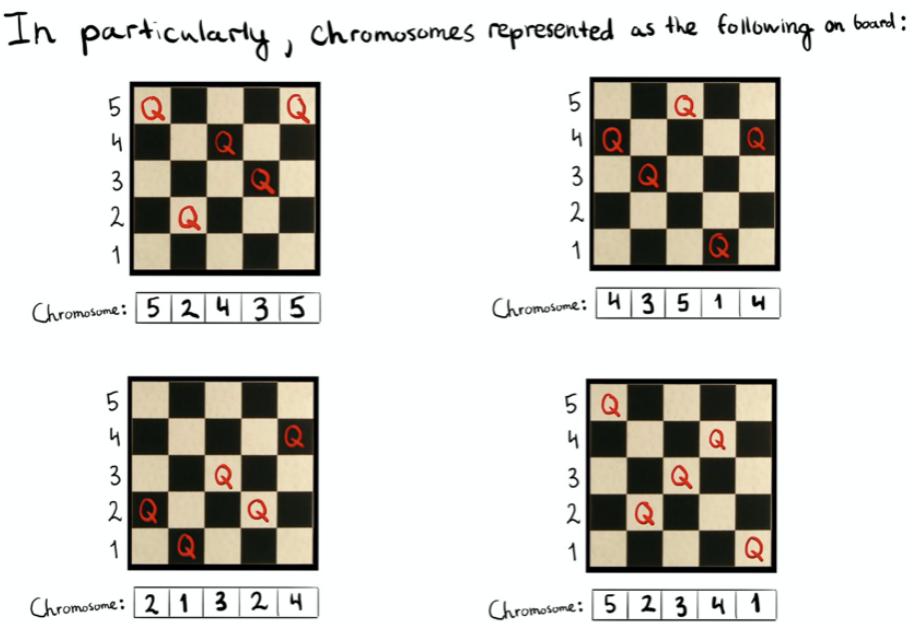
مدل کردن مسئله :

برای مدل کردن این مســـئله ابتدا هر individual را به صـــورت یک آرایه تک بعدی که index های آن نشـــان دهنده ســـتون و مقدار element هر index نشان دهنده سطر در صفحه n*n است .

ایجاد جمعیت اولیه :

ابتدا یک جمعیت اولیه را که یک آرایه دو بعدی از individual ها بوده و هر individual یک آرایه به طول n اســـت و همچنین به شـــکل تصـــادفی مقدار دهی شده است ایجاد میکنیم .

برای همگرایی ســـریعتر به جواب بهینه مقدار دهی اولیه برای هر individual شامل عناصر تکراری نیست یعنی هیچ دو وزیری در یک سطر قرار ندارد.
در ادامه با متد های گفته شده سعی بر ایجاد جوابی است که هیچ دو وزیری به شکل قطری یکدیگر را تحدید نکنند.



تابع Fitness :

این تابع برای ارزیابی هر جواب اســـتفاده میشـــود به طوری که ورودی آن یک individual که یک آرایه به طول n اسـت بوده و خروجی آن یک عدد و شـامل مجموع تعداد حالت هایی است که وزیر ها دو به دو یکدیگر را تحدید نمیکنند.

بنابراین هر عنصـر با عناصـر بعدی خود مقایسـه شـده و در صـورتی که به شـکل سطری و ستونی یا قطری تحدیدی ایجاد نکنند مقدار evaluation را یک واحد افزایش میدهند.

تابع selection :

در هر مرحله برای ایجاد جواب بهتر با fitness بیشتر باید از بین population دو عضو به عنوان parent انتخاب شوند و این انتخاب به شکل تصادفی انجام میشود.

به طوریکه یک آرایه دو بعدی به عنوان ورودی داده شـــده و در هر مرحله k=5 نمونه به شکل تصادفی انتخاب شده و بر اساس تابع fitness مرتب میشوند و در نهایت 2 عضو به عنوان parent انتخاب میشود .

تابع crossover :

در این مرحله با استفاده از parent که در تابع selection انتخاب شدند یک offspring یا فرزند که شـــامل chromosome های parent اســـت برای تولید نسل بعد انتخاب میشود .
نحوه ایجاد فرزند به این شـکل اسـت که چون parent شـامل عناصـر تکراری نیســـت در انتخاب chromosome های فرزند نیز نباید عنصـــر تکراری وجود داشته باشد .

در ابتدا parent با fitness بیشـــتر انتخاب می شـــود ســـپس یک طول به صورت تصادفی از آن انتخاب می شود و بخش ابتدایی offspring را تشکیل میدهد سپس chromosome های parent دوم در با شرط اینکه دو وزیر در

یک ســـطر قرار نگیرند انتخاب میشـــود و درنهایت یک offspring که آرایه به طول n است به عنوان خروجی باز گرداننده میشود.

تابع Mutation :

این تابع یک جهش در offspring ایجاد میکند و به شـــکل تصـــادفی دو column انتخاب میکنند و ســـطر های آن را جا به جا میکند در این صـــورت ممکن است مقدار fitness بهتر شود.
اما جهش در هر مرحله با mutation_rate = 0.5 اتفاق میافتد.

تابع Evolutionary :

ابتدا یک جمعیت اولیه ایجاد و سپس به اندازه تعداد نسل های مشخص شده الگوریتم اجرا می شـود و در هر مرحله parent ها و offspring ایجاد شـده از آنها وارد نسل بعد شده و الگوریتم تکرار میشود .
در نهایت در آخرین generation آرایه ای با بیشترین fitness انتخاب میشود و به عنوان جواب بهینه بازگرداننده میشود .
جواب مســـئله بدون هیچ خطا حالتی اســـت که fitness جواب نهایی برابر n*(n-1)/2 شود .

نتیجه گیری :

حل کردن چنین مسئله هایی به روش بازگشتی به زمان و پردازش زیادی احتیاج دارد که عملا بهینه و کارآمد نیست بنابراین استفاده از الگوریتم تکامل تدریجی به واسطه کمک اعداد تصادفی و انتخاب بهترین جواب ها در هر مرحله میتواند در نسل های بعدی جواب های بهتری نسبت به قبل ایجاد کند و ما را به جواب اصلی مسئله نزدیکتر کند .

همچنین به دلیل ماهیت اعداد تصادفی هر چقدر جمعیت اولیه و تعداد تکرار الگوریتم بیشتر باشد احتمال رسیدن به جواب اصلی مسئله بیشتر میشود .
هرچند در مواردی ممکن است به جواب جواب اصلی مسئله همگرا نشویم اما یک جواب suboptimal بدست می آید که بسیار به جواب اصلی مسئله نزدیک است.

منابع :

- https://www.sciencedirect.com/topics/engineering/genetic-algorithm
- https://towardsai.net/p/computer-science/solving-the-5-queens-problem-using-genetic-algorithm
- https://towardsdatascience.com/introduction-to-evolutionary-algorithms-a8594b484ac