مسئله کوله پشتی با GA

امیرحسین انتظاری ۶۱۰۳۹۹۱۰۳

امیرعلی امینی ۶۱۰۳۹۹۱۰۲

**تعریف مسئله:**

مسئله N-Queen یک مسئله معروف در حوزهٔ ترکیبیات است. در این مسئله، باید N عدد سرباز را در صفحهٔ شطرنجی NxN به گونه‌ای قرار داد که هیچ دو سربازی با هم در تهدید یکدیگر قرار نگیرند. سربازها در صفحهٔ شطرنجی می‌توانند افقی، عمودی و قطری حرکت کنند.

شرایطی که باید برآورده شود عبارتند از:

1. هیچ دو سربازی نباید در یک سطر قرار گیرند.

2. هیچ دو سربازی نباید در یک ستون قرار گیرند.

3. هیچ دو سربازی نباید در یک قطر قرار گیرند.

مسئلهٔ N-Queen برای انواع مختلفی از ابعاد صفحهٔ شطرنجی قابل حل است. هدف این مسئله پیدا کردن یک ترتیب قرارگیری سربازهاست که همهٔ شرایط بالا برآورده شود. یکی از روش‌های حل این مسئله استفاده از الگوریتم‌های زیستی مانند الگوریتم ژنتیک است.

**الگوریتم ممتیک:**

االگوریتم ممتیک یا همان Memetic Algorithm را نمی‌توان یک الگوریتم ثابث فرض کرد. در واقع الگوریتم‌های ممتیک را می‌توان از دسته‌ی الگوریتم‌ها دانست که به نوعی شاخه ای از الگوریتم ژنتیک (Genetic Algorithm) هستند. در این پروژه ما به این الگوریتم‌ می پردازیم.

الگوریتم ژنتیک را به یاد بیاورید. به صورت خلاصه، این الگوریتم با تولید افراد نسل‌های مختلف و ترکیب (Crossover) و جهش (Mutation) در نسل‌ها، می‌توانست به یک حالت بهینه‌ی خوب دست پیدا کند. اما جالب این‌جاست در الگوریتم ژنتیک هر کدام از افراد (ژن‌ها) تلاشی برای بهبود خود نمی‌کنند. در واقع در الگوریتمِ ژنتیک، هر کدام از افراد (ژن‌ها) در نسل‌های مختلف فقط منتظر عملیات Crossover و Mutation هستند تا شاید بهبود حاصل شود، و نکته‌ی اصلی در الگوریتم ممتیک در این‌جا نهفته است. این الگوریتم مانند الگوریتم ژنتیک عمل می‌کند با این تفاوت که هر کدام از افراد (ژن‌ها) بعد از تولد (به وجود آمدن از نسل قبلیِ خود) به بهبود وضعیت خویش به صورت نسبی و محلی می‌پردازند. آن‌ها این کار با الگوریتم تپه‌نوردی (Hill Climbing) انجام می‌دهند.

اگر بخواهیم با مثالی شهودی الگوریتم ممتیک را متوجه شویم، به این صورت است که فرض کنید افراد مختلف (همان ژن‌ها) در جهان گسترده شده‌اند و به دنبال رشد و شکوفایی هستند. آن‌ها با عملیاتی مانند تبادل فرهنگی (Crossover در الگوریتم ژنتیک) قسمتی از فرهنگ خود را می‌دهند و قسمتی از فرهنگ دیگران را دریافت می‌کنند تا با ترکیب آن فرهنگ با فرهنگِ خود، بتوانند شاید به یک حالت بهینه‌تر و رشد و شکوفایی بیشتر دست پیدا کنند. حالا آن‌هایی که بهینه‌تر از بقیه هستند شانس بیشتری برای زنده ماندن و سپس ازدواج با بقیه‌ی بازماندگان دارند! این افرادی که زنده مانده‌اند با افراد دیگر از نقاط جغرافیاییِ دیگر ازدواج کرده و فرزندانی را به وجود می‌آورند و با این‌کار امید دارند که فرزندانشان بهتر از خودشان شوند. تا این‌جای کار الگوریتم ژنتیک بود. الگوریتم ممتیک مانند این است که فرزند متولد شده، در مکانی که قرار دارد، به دنبال بهبود و رشد و شکوفایی خویش (بدون توجه به ترکیب و جهش) باشد. مثلاً خودش به صورت حریصانه هر روز خود را بهبود دهد. سپس دوباره به چرخه‌ی ژنتیک برگردد و با عملیات Crossover و Mutation با کمک هم نسل‌های خود، فرآیند بهبود را در ادامه دهد.

مراحل اصلی الگوریتم ممتیک عبارت‌اند از:

1. ایجاد جمعیت اولیه تصادفی از افراد.

2. ارزیابی برازش کورومزوم افراد بر اساس پارامتر های تعریف شده.

3. تولید نسل بعدی با استفاده از عملیات شامل ترکیب ژنتیکی (تهیه‌ی نسل جدید از ترکیب دو یا چند جهش) و ایجاد نسل جدید.

4. تولید تعدادی از همسایه های یک جواب و پیدا کردن بهترین جواب در کل همسایگی.(local search)

5. انتخاب جهش‌هایی برای تولید نسل بعدی.

6. تکرار مراحل 2 تا 5 تا رسیدن به شرایط خاتمه مسئله یا دستیابی به جواب بهینه.

با تکرار مراحل فوق، الگوریتم ممتیک به تدریج به راه‌حل‌های بهینه نزدیک‌تر می‌شود و می‌تواند در مسائل بهینه‌سازی پیچیده و فضاهای جستجوی بزرگ عملکرد خوبی داشته باشد. الگوریتم ممتیک در مسائل مختلفی مانند یافتن کوتاه ترین مسیر، برنامه‌ریزی زمان،بهینه‌سازی تولید و … مورد استفاده قرار می‌گیرد.

**روش حل مسئله:**

برای حل مسئله n-queen با الگوریتم ممتیک، ابتدا باید تعیین کنیم که جواب های مسئله ما چگونه باید در قالب کورموزوم پیاده سازی شوند. یک راه پیشنهادی این است که بیایم یک جواب مسئله که صفحه شطرنج هست را مستقیما به عنوان کورموزوم انتخاب کنیم. کوروموزوم گفته شده به صورت یک ماتریکس خواهد بود. اما این روش احتمال هزینه محاسبات را بسیار بالا میبرد. روش دیگر برای نمایش راه حل در کورموزوم، استفاده از نمایش آرایه ای است، در این روش، اندیس هر آرایه، ستون های شطرنج در نظر گرفته شده، و مقدار درون هر آرایه نیز نشان دهنده سطر است. از مزایای این روش این است که میتوان در ابتدای کار، یک جایگشت از اعداد ۱ تا n تولید کرد، و برای عملیات هایی مانند جهش، صرفا جای دو عضو را عوض کرد.

پس از انتخاب نحوه نمایش کوروموزم، ابتدا یک جمعیت اولیه از کورموزوم ها ایجاد میکنیم. من اینکار را با کوروموزم اولیه با مقادیر رندوم از جایگشت های ۱ تاn انجام دادم. تعداد جمعیت اولیه نیز بسته به بزرگی ورودی مسئله تغییر میکند. سپس، مقدار فیتنس هر کورموزوم را در یک آرایه ذخیره میکنیم که خانه i ام آرایه مربوط به کورموزوم i ام از جمعیت اولیه است.

حال باید یک تابع بازترکیب(crossover) برای ترکیب دو کورموزوم با یکدیگر ارائه دهیم. من از روشی شبیه بازترکیب دو نقطه ای برای تابع بازترکیب استفاده کردم. در این روش ابتدا بچه اول برابر با والد اول و بچه دوم برابر با بچه دوم میشود، سپس دو اندیس رندوم انتخاب میشود و از اندیس اول تااندیس دوم کوروزموم والد دوم به بچه اول و از اندیس اول تا اندیس دوم والد اول به بچه دوم ارث می رسد. اما نکته مهم این است که از تکرار اعداد جلوگیری میشود. یعنی اگر عددی تکراری بود، به ارث نمیرسد و درنهایت این موضوع هندل میشود. در نهایت بچه با برازش بیشتر انتخاب می شود.

در کنار تابع بازترکیب باید تابع جهش را نیز تعریف کنیم، تا بتوانیم به الگوریتممان پویش بدهیم. از بین روش های جهش موجود، من روش جا به جایی را انتخاب کرده ام. در این روش دو اندیس رندوم انتخاب شده، و مقدار موجود در کورومزوم با این اندیس ها با هم جابه جا میشوند.

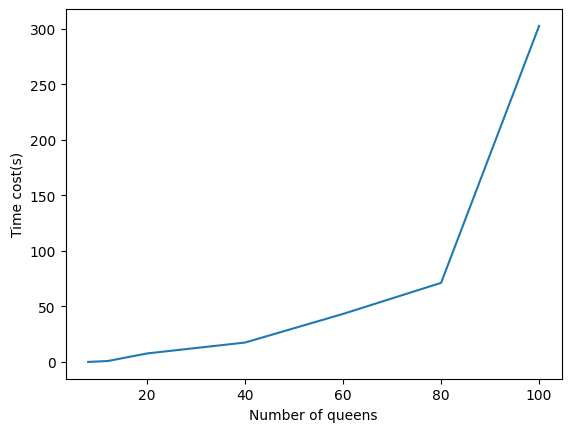
پس از بازترکیب و تولید یک بچه جدید، باید پوینت اصلی الگوریتم ممتیک، یعنی جستجوی محلی را نیز انجام دهیم. برای اینکار، ابتدا یک تابع همسایه تعریف میکنیم که یک کورموزوم را گرفته و همسایه ای رندوم از آن را تولید میکند. قبل از تعریف خود تابع، باید تعریف کنیم که همسایگی در مسئله با چگونه تعریف می شود؟ یک همسایگی از جواب موجود را میتوان اینگونه در نظر گرفت که جای دو وزیر با یکدیگر عوض شود. از انجا که در هر سطر و ستون، فقط یک وزیر وجود دارد، این همسایگی منطقی است. در نتیجه دو اندیس رندوم انتخاب میکنیم، سپس مقدار مربوط در کروموزوم با اندیس ها را با یکدیگر عوض میکنیم.

پس از تعریف تابع همسایه، باید تابع جستجوی محلی را تعریف کنیم. این تابع را اینگونه تعریف میکنیم که یک پارامتر n میگیرد و به تعداد n تا، همسایه رندوم تولید میکند که شامل خود کورمورزوم هم می شود. سپس از بین این همسایه ها، همسایه ای با بهترین برازش را خروجی می دهد. هر چند میتوانیم برای جلوگیری از پویش بیش از حد، با احتمال یک پارامتر(مثلا ۰.۲) به جای بهترین همسایه، یک همسایه رندوم خروجی دهیم.

حال که تابع بازترکیب و جهش و جستجوی محلی را تعریف کردیم، میتوانیم به تعریف تابع انتخاب بپردازیم. از بین روش های موجود برای تابع انتخاب، من روش تورنومنت و رنک را پیاده سازی کرده ام. هر چند که روش رنک بسیار انتفاع را زیاد میکند و سریع در نقطه بهینه محلی گیر میوفتیم و در نتیجه استفاده ای از آن نکردم. در روش تورنومنت، ابتدا n عضو برای والد اول، سپس m عضو برای والد دوم انتخاب کردم، و بهترین والد ها را از هر دوی این دسته برای بازترکیب انتخاب میکنم. پس از تولید هر بچه، کوروموزوم را به تابع جهش داده و با احتمال ۰.۲ جهش میدهیم. تفاوت اصلی این تابع انتخاب با تابع انتخاب در الگوریتم ژنتیک، این است که باید در این تابع، پس از تولید هر فرزند، نقطه بهینه محلی فرزند را نیز پیدا کنیم.

نکته ای که باید در هنگام بازترکیب و جهش همواره توجه کنیم، قابل قبول بودن(feasible) جواب هاست. اما این موضوع با توجه به پیاده سازی مان در کوروموزم و توابع، به طور خودکار برطرف میشود.

در نهایت عملیات انتخاب را تا زمان برقراری شرط مورد نظر که من برای این الگوریتم، رسیدن به جواب پیشنهادی و محدودیت نسل را انتخاب کرده ام، ادامه میدهیم.

**تحلیل نمودار:**

همانطور که در نمودار ملاحظه میکنیم، با افزایش تعداد وزیر ها ، میزان زمان صرف شده برای رسیدن به جواب، به صورت نمایی رشد میکند. این به علت آن است که فضای سرچ بسیار بزرگ میشود، اما فضای جواب های مورد نظر با سرعت نمایی رشد نمیکند، درنتیجه در زمان کم به جواب های خوب میرسیم، اما به جواب های بهترین(۰ برخورد) یا نمیرسیم، یا در زمان زیاد و پویش و انتفاع زیاد میرسیم. هر چند که در این الگوریتم، با اینکه جستجوی محلی تاثیر داده شده است، اما تغییر چندانی به علت دلیل گفته شده ملاحظه نمیشود.