مسئله کوله پشتی با SA

امیرحسین انتظاری ۶۱۰۳۹۹۱۰۳

امیرعلی امینی ۶۱۰۳۹۹۱۰۲

**تعریف مسئله:**

مسئله کوله‌پشتی ۰-۱ یک مسئله بهینه‌سازی در علم کامپیوتر است که در آن باید مجموعه‌ای از اشیاء با ارزش و وزن داده شده را درون یک کوله‌پشتی با ظرفیت مشخص قرار داد. هر شیء دارای وزن و ارزشی است و باید تعیین کنیم کدام شیء‌ها را درون کوله‌پشتی قرار دهیم تا مجموعه‌ای با بیشترین ارزش را داشته باشیم، در حالی که وزن کل شیء‌ها ظرفیت کوله‌پشتی را نباید بیشتر کند.

در این مسئله، هر شیء را می‌توان به دو صورت گرفته یا نگرفته در نظر گرفت. به عبارت دیگر، مجموعه شیء‌های قابل انتخاب برای قرار دادن در کوله‌پشتی یا ۰ (شیء را نگرفتن) یا ۱ (شیء را گرفتن) است.

هدف در این مسئله انتخاب مجموعه‌ای است که مجموع ارزش شیء‌های داخل آن بیشینه شود، در حالی که وزن کلی آن مجموعه نباید بیشتر از ظرفیت کوله‌پشتی باشد. این مسئله به عنوان یک مسئله بهینه‌سازی معروف است و الگوریتم‌های مختلفی برای حل آن وجود دارد.

ما در این پروژه سعی داریم با الگوریتم شبیه سازی تبرید، جواب بهینه ای برای آن پیدا کنیم.

**الگوریتم شبیه سازی تبرید:**

الگوریتم Simulated Annealing یک روش بهینه‌سازی است که بر اساس روشی مشابه فرایند تبرید در فیزیک ایجاد شده است. این الگوریتم برای حل مسائل بهینه‌سازی تکراری که ممکن است در فضای جستجوی بسیار بزرگ دچار مینیمم محلی شوند، استفاده می‌شود.

الگوریتم Simulated Annealing از یک مفهوم سردکردن تدریجی (annealing) برای اکتشاف فضای جستجو استفاده می‌کند. این مفهوم برگرفته از روشی است که در فرآیند تبرید فلزات استفاده می‌شود تا به ساختاری پایدارتر برسند. در این روش، در ابتدا، دما بالا قرار می‌گیرد و سپس تدریجاً به مقدار کمتری کاهش می‌یابد. همچنین، این الگوریتم از مفهوم "احتمال برگشت" استفاده می‌کند تا به جای مینیمم محلی، بتواند به جستجوی مینیمم سراسری بپردازد.

مراحل اصلی الگوریتم تبرید شبیه سازی شده عبارت‌اند از:

۱. یک حالت اولیه تصادفی انتخاب می‌شود.

۲. یک حالت جدید از حالت فعلی تولید می‌شود (معمولاً با یک تغییر کوچک در حالت فعلی که آن را همسایه می نامیم).

۳. اگر حالت جدید بهتر از حالت فعلی باشد، آن را به عنوان حالت فعلی قبول می‌کنیم.

۴. اگر حالت جدید بدتر از حالت فعلی باشد، با احتمالی مشخص، آن را با استفاده از تابعی قبول می‌کنیم.

۵. دما کاهش می‌یابد و فرایند تا زمانی که دما به حداقلی مشخصی برسد، ادامه می‌یابد.

این الگوریتم به این صورت کار می‌کند که در ابتدا به صورت تصادفی در فضای جستجو حرکت می‌کند و به طور تدریجی به احتمال بیشتر به سمت جستجوی مینیمم سراسری حرکت می‌کند. به دلیل مفهوم سردکردن تدریجی، الگوریتم از محلی شدن در مینیمم محلی جلوگیری می‌کند و به جستجوی جواب بهینه در فضای جستجوی بزرگتر ادامه می‌دهد.

Simulated Annealing می‌تواند برای حل مسائل بهینه‌سازی پیچیده مانند در مسائل مختلفی مانند یافتن کوتاه ترین مسیر، برنامه‌ریزی زمان،بهینه‌سازی تولید و … مورد استفاده قرار گیرد. این الگوریتم به دلیل قابلیت کار با فضاهای جستجوی بزرگ و جستجوی مینیمم سراسری، محبوبیت زیادی در بین محققان در بهینه‌سازی دارد.

**روش حل مسئله:**

برای حل مسئله کوله پشتی ۰-۱ 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 شامل تعدادی شی هست را مستقیما به عنوان کورموزوم انتخاب کنیم. کوروموزوم گفته شده به صورت زیر خواهد بود:

Chromosome = [80, 82, 85, 70, 72, 70, 66, 50, 55, 25, 50, 55, 40]

که اعداد درون آرایه، وزن اشیا میباشند. روش دیگر برای نمایش راه حل در کورموزوم، استفاده از نمایش باینری است، که به نظر من روش بهتری است چرا که اگر چند شی با وزن تکراری داشته باشیم، پیاده سازی چک کردن موجود بودن آن در کوله پشتی راحت تر است. کوروموزوم گفته شده به صورت زیر خواهد بود:

Chromosome = [0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0,]

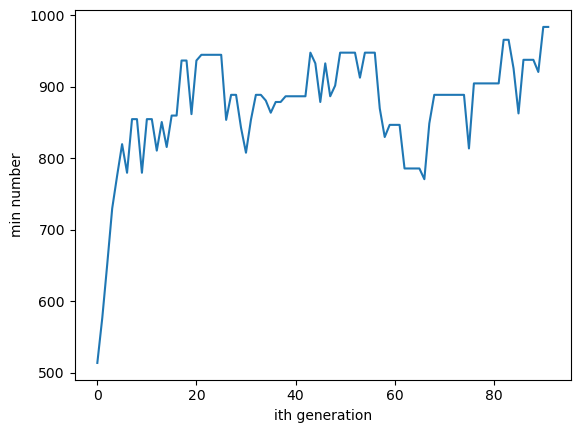
حال یک تابع همسایه تعریف میکنیم که یک کورموزوم را گرفته و همسایه ای رندوم از آن را تولید میکند. قبل از تعریف خود تابع، باید تعریف کنیم که همسایگی در مسئله با چگونه تعریف می شود؟ یک همسایگی از جواب موجود را میتوان اینگونه در نظر گرفت که یک شی را که درون کوله پشتی قرار داده ایم، خارج کنیم، یا شی ای که آن را درون کوله پشتی قرار نداده ایم، درون کوله پشتی بگذاریم. بدین ترتیب ما دو دسته همسایگی داریم. حال به تعریف تابع همسایه می پردازیم. برای تولید یک همسایه از جواب، میتوانیم ابتدا یک اندیس رندوم از اشیائی که برداشته نشده اند و یک اندیس رندوم از اشیائی که برداشته شده اند تولید کنیم. سپس هرکدام از این اشیا انتخاب شده را، معکوس میکنیم(اگر داخل کوله پشتی است، خارج میکنیم، اگر خارج کوله پشتی است، داخل کوله پشتی میگذاریم). حال دو همسایه از جواب را تولید کرده ایم. کدام را انتخاب کنیم؟ من برای این انتخاب بدین گونه عمل کردم که ابتدا میبینم که اگر هر دو همسایه، هردو شرط قابل قبول بودن(feasible) را برآورده یا هر دو شرط قابل قبول بودن را رد میکنند، همسایه با برازش بهتر را خروجی میدهیم. اگر هم یکی از آن ها شرط قابل قبول بودن را برآورده میکرد، فقط همان را خروجی میدهیم.

پس از انتخاب همسایه، نوبت به تابع اصلی الگوریتم، یعنی تبرید میرسد. در این تابع، در هر ایتریشن، یک همسایه از جواب موجود را میگیریم و فیتنس آن را محاسبه میکنیم. اگر فیتنس آن، از فیتنس قبلی بهتر بود، فیتنس جدید را جایگزین بهترین فیتنس میکنیم، در غیر این صورت با استفاده از یک پارامتر رندوم، آن را جایگزین بهترین فیتنس میکنیم که این عمل در واقع جهش را برای ما انجام میدهد. سپس دما را با استفاده از ضریب سرمایش، سرد میکنیم.

نکته قابل تامل در این الگوریتم این است که، با گذشت زمان دما بسیار کوچک می شود، و از انجا که اختلاف فیتنس تقسیم بر این عدد کوچک(کوچکتر از ۱) می شود، یک عدد با اندازه بزرگ میشود و سپس این عدد منفی شده، و e به توان آن میرسد که یک عدد بسیااارررر کوچک را به ما میدهد. در نتیجه با گذشت زمان، میزان پویش ما هم کم و کمتر میشود.

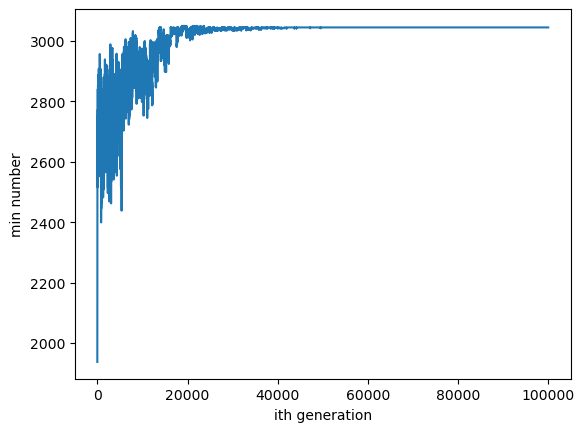
**نمودار:**

**تست کیس اول:**



همانطور که ملاحظه میکنیم با پارامتر های داده شده( میزان پارامتر دما در این مسئله اهمیت دارد، چرا که اختلاف فیتنس ها زیاد است، و متناسب با آن باید دمای اولیه را انتخاب نمود) در ایتریشن بسیار کمی به جواب میرسیم. این ایتریشن نسبت به ژنتیک الگوریتم بهتر بوده ولی از ممتیک الگوریتم ضعیف تر است. البته با توجه به اینکه الگوریتم های SA معمولا در ایتریشن بسیار بالایی به جواب میرسند، این جواب برای مسئله خیلی خوب است.

**تست کیس دوم:**

همانطور که ملاحظه میکنیم، با زیاد شدن اشیا تا ۵۰ عدد، برای رسیدن به جواب، حدودا ۵۰۰۰۰ ایتریشن باید طی شود که نسبت به حالت قبل خیلی بیشتر شده است.

هرچند که در الگوریتم تبرید، در ایتریشن های بیشتری به جواب میرسیم، اما از لحاظ زمانی، بسیار بسیار سریع تر از دیگر الگوریتم ها بهترین جواب پیدا میشود.