به نام خدا

گرآورنده: سروش میرشکاری

کد دانشجویی: ۴۰۲۳۰۰۰۳۲

گزارش پیاده‌سازی الگوریتم ژنتیک برای حل سیستم‌های معادلات

مقدمه

این گزارش به بررسی و تحلیل یک پیاده‌سازی از الگوریتم ژنتیک (GA) می‌پردازد که برای حل سیستم‌های معادلات جبری، شامل سیستم‌های خطی و غیرخطی، طراحی شده است. الگوریتم ژنتیک یک روش بهینه‌سازی الهام‌گرفته از فرآیند تکامل بیولوژیکی است که از مکانیزم‌هایی مانند انتخاب طبیعی، تقاطع (crossover) و جهش (mutation) برای یافتن راه‌حل‌های تقریبی برای مسائل پیچیده استفاده می‌کند.

هدف اصلی این پیاده‌سازی، یافتن مقادیر متغیرها (مانند t,z,y,x) است که به طور همزمان تمامی معادلات یک سیستم را برآورده می‌کنند. این کار با به حداقل رساندن خطای کل سیستم (که به عنوان تابع هدف یا تابع برازندگی معکوس در نظر گرفته می‌شود) انجام می‌پذیرد.

ساختار کد

کد ارائه شده به سه بخش اصلی تقسیم می‌شود:

۱. اجزای اصلی الگوریتم ژنتیک

این بخش شامل توابع پایه‌ای است که برای هر پیاده‌سازی GA ضروری هستند و مستقل از مسئله خاص عمل می‌کنند:

- create_individual(genome_length, variable_range):

این تابع یک فرد (یا راه‌حل کاندید) را ایجاد می‌کند. هر فرد یک ژنوم است که لیستی از مقادیر شناور تصادفی را نشان می‌دهد. genome_length تعداد متغیرها در راه‌حل (به عنوان مثال، ۲ برای یک سیستم ۲×۲) و variable_range محدوده‌ای است که مقادیر متغیرها می‌توانند در آن قرار گیرند (به عنوان مثال، (−۱۰۰,۱۰۰)).

- tournament_selection(population, fitnesses, k=۵):

این تابع یکی از والدین را برای نسل بعدی انتخاب می‌کند. در انتخاب تورنمنت، k فرد به صورت تصادفی از جمعیت انتخاب می‌شوند و فردی که بالاترین برازندگی را در این گروه دارد، به عنوان والد انتخاب می‌شود. این روش تضمین می‌کند که افراد با برازندگی بالاتر شانس بیشتری برای انتخاب شدن دارند.

- crossover(parent۱, parent۲, crossover_rate=۰٫۸):

این تابع تقاطع (crossover) را بین دو والد انجام می‌دهد تا دو فرزند جدید تولید کند. از روش تقاطع حسابی (Arithmetic Crossover۱) ستفاده می‌شود که در آن فرزندان ترکیبی خطی از ژن‌های والدین هستند.
crossover_rate احتمال وقوع تقاطع را کنترل می‌کند؛ اگر یک عدد تصادفی از این نرخ کمتر باشد، تقاطع رخ می‌دهد، در غیر این صورت والدین بدون تغییر به عنوان فرزند بازگردانده می‌شوند.

- mutate(individual, mutation_rate=۰٫۱, mutation_strength=۰٫۵):

این تابع جهش (mutation) را بر روی یک فرد انجام می‌دهد. جهش با اضافه کردن یک مقدار تصادفی کوچک به ژن‌های فرد صورت می‌گیرد. mutation_rate احتمال جهش برای هر ژن را کنترل می‌کند و mutation_strength حداکثر بزرگی تغییر تصادفی را تعیین می‌کند. جهش تنوع را در جمعیت حفظ می‌کند و به الگوریتم کمک می‌کند تا از حداقل‌های محلی فرار کند.

۲. توابع برازندگی (Fitness Functions) مختص مسئله

این بخش شامل توابعی است که میزان "خوبی" یک راه‌حل کاندید را برای هر سیستم معادله خاص محاسبه می‌کنند. تابع برازندگی بر اساس مجموع مربعات خطاها از هر معادله سیستم تعریف می‌شود. هرچه مجموع مربعات خطاها کمتر باشد، برازندگی بالاتر است. برای جلوگیری از تقسیم بر صفر و تسهیل تبدیل به یک مسئله بهینه‌سازی حداکثری، تابع برازندگی به صورت ۱/(ε+total_error) تعریف شده است، که ε یک عدد بسیار کوچک است.

- fitness_function_part(genome):۱

برای یک سیستم خطی ۲×۲:

- ٤=y۲+x
- ۱۲=y٤+x٤ مقادیر x و y از ژنوم استخراج شده و خطای هر معادله محاسبه می‌شود.
- fitness_function_part(genome):۲

برای یک سیستم غیرخطی ۳×۳:

- ۲۰=z۸+y۲-x٦
- ۱-=xz۸+y
- ٦=y۱٫۵+x/z۱۲ این تابع شامل یک بررسی برای x≈۰ است تا از خطای تقسیم بر صفر جلوگیری شود و در صورت وقوع، برازندگی بسیار پایینی (پنالتی) اعمال می‌شود.
- fitness_function_part(genome):۳

برای یک سیستم خطی ٤×٤:

- ۳=t(۵/٤)-z۱۵-y۲-x(۱۵/۱)
- ۱۷=t-z۱۲+y۲٫۲۵-x۲٫۵-
- ۱۷=t۰٫٤-z٦-y۰٫۳+x۱۳-
- ۹-=t(۳/٤)+z۱٫۷۵+y۲+x۰٫۵ خطاهای مربوط به هر چهار معادله محاسبه می‌شوند.

۳. موتور اصلی حل‌کننده GA

run_genetic_algorithm(...):

این تابع هسته اصلی الگوریتم ژنتیک را پیاده‌سازی می‌کند. مراحل اصلی عبارتند از:

- ۱. مقداردهی اولیه: یک جمعیت اولیه از افراد تصادفی ایجاد می‌شود.
- ۲. حلقه اصلی: برای تعداد مشخصی از نسل‌ها تکرار می‌شود:
  - ارزیابی برازندگی: برازندگی هر فرد در جمعیت محاسبه می‌شود.
  - حفظ بهترین راه‌حل: بهترین فرد یافت شده در نسل جاری و در طول کل اجرای الگوریتم نگهداری می‌شود.
  - ۳. ایجاد نسل بعدی:
    - نخبه‌گرایی ((Elitism: بهترین فرد نسل قبلی مستقیماً به نسل بعدی منتقل می‌شود تا از از دست دادن بهترین راه‌حل اطمینان حاصل شود.

- تولید فرزندان: با استفاده از انتخاب تورنمنت، دو والد انتخاب می‌شوند. سپس تقاطع بین آنها انجام شده و فرزندان تولید می‌شوند. در نهایت، جهش بر روی فرزندان اعمال شده و به جمعیت بعدی اضافه می‌شوند تا زمانی که جمعیت جدید به اندازه مورد نظر برسد.
  - گزارش پیشرفت: در طول اجرای الگوریتم، گزارش‌های دوره‌ای از بهترین برازندگی و خطا چاپ می‌شود.
- ٤. نتیجه نهایی: پس از اتمام تمامی نسل‌ها، بهترین راه‌حل یافت شده، مقدار خطای مربوطه و در صورت وجود، راه‌حل صحیح (جهت مقایسه) چاپ می‌شود.

اجرای الگوریتم

در بخش if __name__ == '__main__':، الگوریتم برای بخش ۱: سیستم خطی ۲×۲ اجرا می‌شود. پارامترهای مختلفی مانند generations (تعداد نسل‌ها)، pop_size (اندازه جمعیت) و variable_range (محدوده متغیرها) برای هر بخش تعریف شده‌اند تا بهینه‌سازی مناسبی صورت گیرد. بخش‌های ۲ و ۳ به صورت کامنت شده‌اند و می‌توان با حذف کامنت‌ها آنها را نیز اجرا کرد.

پارامترهای پیش‌فرض استفاده شده برای بخش ۱:

- genome_length: ۲ (برای متغیرهای x و y)
- generations: ۲۰۰
- pop_size: ۱۰۰
- variable_range: (−۱۰۰,۱۰۰)
- correct_solution: [۲/۰,۱/۰]

نتایج (مثال برای بخش ۱)

پس از اجرای الگوریتم، خروجی شامل گزارش پیشرفت هر چند نسل و در نهایت بهترین راه‌حل یافت شده توسط GA، مقدار خطای نهایی و راه‌حل صحیح (در صورت ارائه) خواهد بود. به عنوان مثال برای بخش ۱، انتظار می‌رود که الگوریتم به مقادیری بسیار نزدیک به x=۲/۰ و y=۱/۰ با خطای نزدیک به صفر همگرا شود.

مثال خروجی برای بخش ۱:

```
========== Running GA for Part 1: 2x2 Linear System ==========
Generation   0: Best Fitness = 0.00, Error = 96.00000000
Generation  20: Best Fitness = 0.02, Error = 52.88049089
Generation  40: Best Fitness = 0.04, Error = 26.68965825
...
Generation 180: Best Fitness = 1000000000.00, Error = 0.00000000

--- Final Result for Part 1: 2x2 Linear System ---
GA finished after 200 generations.
Best solution found: x = 2.0000, y = 1.0000
Correct solution is: x = 2.0000, y = 1.0000
Final Error (Sum of Squares): 0.00000000
================================================
```

نتیجه‌گیری

این پیاده‌سازی نشان می‌دهد که الگوریتم ژنتیک می‌تواند ابزاری قدرتمند برای حل سیستم‌های معادلات، حتی معادلات غیرخطی، باشد. با تنظیم مناسب پارامترهای (GAمانند اندازه جمعیت، تعداد نسل‌ها، نرخ جهش و تقاطع)، می‌توان به راه‌حل‌های بسیار دقیق نزدیک شد. این روش به ویژه در مسائلی که روش‌های تحلیلی پیچیده یا ناممکن هستند، کاربرد پیدا می‌کند.