پروژهٔ هوش مصنوعی، گزارش الگوریتم ژنتیک

تکین جزایری، ۹۸۱۳۰۰۶

قسمت اصلی برنامه، تابعی به نام genetic است که الگوریتم ژنتیک را پیاده سازی می کند. پیش از اجرای این تابع، آدرس فایل ورودی از کاربر گرفته می شود و از روی فایل دریافت شده، ماتریس مجاورت گراف (adj_matrix) تشکیل داده می شود. سپس تابع genetic فراخوانی می شود. مقادیری که به عنوان پارامتر ورودی به تابع ارسال می شود، شامل مقدار جمعیت مدنظر (popu)، تعداد تکرار (iter_num) و احتمال رخداد جهش در هر ژن (mutation_possibility) نیز می شود.

در تابع genetic، ابتدا لیست chro_list برای نگهداری کروموزوم ها ساخته می شود. اعضای این لیست، نمونه هایی از کلاس Chromosome هستند. هر کروموزوم نشان دهندهٔ یک روش جداسازی اجتماعات است و یک لیست (به طول یکی بیشتر از تعداد گره ها) به نام genes دارد. به طوری کلی i ام عضو این لیست (genes[i]) شمارهٔ یکی از همسایه های گرهٔ i است که با هم در یک اجتماع قرار می گیرند و به این طریق اجتماعات ساخته می شوند (به جز genes[0] که مقدار آن همواره برابر 0 است).

علاوه بر این کلاس Chromosome دارای سه متد زیر هم هست:

- Q: این متد با گرفتن ماتریس مجاورت، مقدار Q را با فرمول زیر محاسبه می کند:

$$Q = \frac{1}{2m}\sum_{i,j}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j)$$

یک مرحله مهم در این محاسبه، پیدا کردن اجتماعات پدید آمده از روی genes است. این مرحله با ایجاد یک کلاس Partition انجام شده است که یک نمونهٔ ساده از ساختمان دادهٔ Disjoint-set است.
لازم به توجه است که با توجه به احتمال وجود خطا در محاسبات ماتریسی، ممکن است مقدار Q به جای صفر، یک مقدار منفی بسیار کوچک بگیرد (مثلاً از مرتبهٔ 10^{-16})؛ بنابراین در انتهای محاسبه مطمئن می شویم مقدار Q حاصل کمتر از صفر یا بیشتر از یک نشود.

- crossover: این متد با دریافت دو کروموزوم، یک کروموزوم جدید که حاصل بازترکیب آنهاست را برمی گرداند. روش بازترکیب به این صورت است که هر ژن کروموزوم فرزند، به ازای احتمال برابر از یکی از ژن های متناظر در یکی از دو والد انتخاب می شود.
- mutation: این متد یک کروموزوم را دریافت می کند و روی آن جهش انجام می دهد. به این صورت که هر یک از ژن هایش به احتمال معینی (برابر با mutation_possibility) تغییر می کند و شمارهٔ یکی دیگر از همسایه هایش را می گیرد.

در تابع genetic، پس از ایجاد لیست chro_list، از تابع random_chro برای ساخت رندوم اعضای اولیهٔ آن استفاده می شود. سپس تابعی به نام choose_parent را فراخوانی می کنیم.

این تابع مقدار تابع برازندگی (Q) را برای تمام کروموزوم های موجود محاسبه می کند. سپس به هر یک از این متغیر ها یک احتمال انتخاب شدن به عنوان والد می دهد و در لیست possibility ذخیره می کند. این احتمال برای هر کوروموزوم متناظر با نسبت مربع Q برای آن کروموزوم به مجموع مربعات مقدار Q برای کل جمعیت است. سپس به طور رندوم بر اساس احتمالات محاسبه شده به تعداد 2-(2*popu) والد انتخاب می کند تا از هر جفت از آنها یک کروموزوم فرزند تولید شود.

این تابع همچنین متغیری به نام maximum را مقداردهی می کند. این متغیر نشان دهندهٔ شماره کوروموزومی با بیشترین مقدار Q در جمعیت است. دلیل مشخص کردن این کروموزوم، حفظ آن برای تشکیل جمعیت بعدی است. به عبارتی، الگوریتم همواره بهترین کروموزوم خودش را در جمعیت حفظ می کند و به همین دلیل به تعداد popu-1 کروموزوم جدید ایجاد می کند.

در گام بعد، بین جفت والدهای انتخاب شده، متد crossover را اجرا می کنیم و کروموزوم حاصل را در لیست new_chro_list می ریزیم. سپس روی کروموزوم های جدید متد mutation را اجرا می کنیم. در آخر، تمام کروموزوم های جدید (اعضای لیست new_chro_list) را در لیست chro_list می ریزیم.

پس از اتمام مراحل، تابع genetic مقدار max_value و best_partitioning را بر می گرداند که best_partitioning یک لیست است که هر یک از اعضای آن یکی از اجتماعات تشخیص داده شده است و به کمک کلاس Partition ساخته شده است.

کد این برنامه در لینک گیت هاب زیر هم موجود است:

https://github.com/TakeenJazayeri/Community-Detection.git