پروژهٔ هوش مصنوعی، گزارش الگوریتم فاخته

تکین جزایری، ۹۸۱۳۰۰۶

پیش از اجرای خود الگوریتم فاخته، برنامه فایل ورودی را دریافت کرده و از روی آن دو لیست adj_matrix و ngh_matrix را تشکیل می دهد که اولی ماتریس مجاورت گراف ورودی و دومی لیست همسایه های هر گره در گراف است. سپس تابع cuckoo_algorithm که حاوی الگوریتم فاخته است اجرا می شود.

کلاسی که فاخته ها را در آن ذخیره کرده و با استفاده از آن روی فاخته ها کار انجام می دهیم Cuckoo نام دارد. هر object از این کلاس دارای یک لیست habitat است. لیست به این شکل ساخته می شود که habitat[0] برابر 0 است و برای بقیه اعضا، habitat[i] = k به این معنا است که گرهٔ i و k امین همسایه اش (در ngh_matrix) در یک اجتماع قرار می گیرند. کلاس Cuckoo دارای متدهای زیر است:

- Q: با استفاده از adj_matrix و کلاس Partition که مشابه ساختمان دادهٔ Disjoint-set عمل می کند، مقدار Q را برای فاخته مدنظر محاسبه می کند.
- difference: برای بررسی مهاجرت و همچنین محل های مناسب برای تخم گذاری فاخته ها، نیاز است معیاری برای فاصلهٔ آنها ایجاد شود. گسسته بودن فضا و همچنین تفاوت دامنه در اعضای habitat، در محاسبهٔ فاصله ایجاد مشکل می کنند. با توجه به این مسائل، متد difference اختلاف دو فاخته را به صورت مجموع اختلاف اعضای habitat آنها محاسبه می کند.
- spawn: این متد با داشتن تعداد تخم های فاخته و همچین ELR آن (فاصله ای که فاخته تخم هایش را در آن فاصله از خود قرار می دهد) ، به تعداد معین برای فاخته تخم ایجاد می کند. روش کار این است که برای هر تخم، ابتدا habitat را همان habitat فاخته والد قرار می دهد و سپس روی هر عضو به احتمال مشخصی تغییر ایجاد می کند. در پایان اگر فاصله تخم با فاختهٔ والد کمتر از ELR بود (یعنی تخم در محدودهٔ مجاز بود)، آن تخم را به لیست تخم های فاخته اضافه می کنند و تا زمان رسیدن به تعداد تخم معین این کار را تکرار می کند.

بخش اصلی برنامه، که همان تابع cuckoo_algorithm است، به این شکل عمل می کند که در ابتدا لیست cuc_list را ایجاد می کند. این لیست در هر مرحله شامل تمام فاخته های بالغ می شود. قبل از شروع مراحل الگوریتم، به تعداد جمعیت تعیین شده (popu) فاختهٔ رندوم ایجاد شده و در لیست cuc_list قرار می گیرد. همچنین دو متغیر best_cuc و max_Q ساخته می شود تا تابع بتواند بهترین فاخته مشاهده شده در طول مراحل و همچنین مقدار Q آن را ذخیره کند و در پایان به عنوان خروجی برگرداند.

مراحل الگوریتم به تعداد iter_num (که از پارامترهای ورودی تابع است) تکرار می شود. در ابتدای هر مرحله تعداد تخم ها و همچنین ELR متناسب با هر یک از فاخته ها مشخص می شود. این محاسبات توسط تابع set_egg_and_ELR انجام می شود. این تابع بجز popu، سه متغیر دیگر را به عنوان پارامتر دریافت می کند: متغیرهای var_low و var_high که حد بالا و پایین تعداد تخم های هر فاخته است و متغیر alpha که یکی از

پارامتر های تأثیرگذار در محاسبهٔ ELR است. تابع set_egg_and_ELR به صورت رندوم تعداد تخم های هر فاخته را مشخص می کند و سپس ELR را با فرمول زیر حساب می کند:

$$\mathrm{ELR} = \alpha \times \frac{\text{Number of current cuckoo's eggs}}{\text{Total number of eggs}} \times (var_{\mathrm{hi}} - var_{\mathrm{low}})$$

و در نهایت، دو لیست شامل تعداد تخم ها و ELR همهٔ فاخته ها را بر می گرداند.

در قدم بعد، الگوریتم با استفاده از دو لیست بدست آمده از تابع set_egg_and_ELR و همچنین متد spawn، برای هر یک از فاخته های موجود در جمعیت به تعداد مشخص شده تخم ایجاد می کند. تخم ها به احتمال مشخصی از بین می روند و تخم های از بین نرفته در لیست egg_list قرار خواهند گرفت.

سپس برنامه با استفاده از تابع choosing_best_egg با حذف تخم های معیوب تعداد تخم ها را به popu می رساند. برای این کار برای تمام تخم ها Q را محاسبه می کند و مقادیر را به ترتیب در لیست egg_list_Q قرار می دهد. در ادامه تابع two_sort را روی egg_list_Q و egg_list اجرا می کند. این تابع egg_list_Q را با روش marge-sort مرتب می کند با این تفاوت که تمام تغییرات روی egg_list_Q در egg_list هم اعمال می شود. به عبارتی در آخر اعضای هر دو لیست به ترتیب از Q بیشتر به Q کمتر مرتب شده اند و می توانیم به تعداد popu از این تخم ها جدا کنیم.

در آخرین گام لیست تخم ها جایگزین لیست فاخته ها می شود (به عبارتی تخم ها بالغ می شوند). فاخته ای که بهترین Q را دارد (cuc_list_Q[0]) با best_cuc مقایسه می شود و اگر مقدار Q آن بیشتر بود، جایگزین best_cuc می شود. البته اگر در آخرین تکرار نباشیم، نیاز است که مهاجرت این فاخته ها را هم اجرا کنیم که به وسیلهٔ دو تابع migration_dest و migration پیاده سازی می شود.

تابع migration_dest از الگوریتم KMeans به منظور خوشه بندی فاخته ها به سه خوشه استفاده می کند. پس از اجرای این الگوریتم، در بین خوشه ها، خوشه ای که فاخته های آن بیشترین مجموع Q دارند انتخاب می شود و مرکز آن خوشه به عنوان مقصد مهاجرت برگردانده می شود.

در تابع migration مهاجرت فاخته ها به سمت مقصد به دست آمده پیاده سازی می شود. روش شبیه سازی مهاجرت به این شکل است که هر یک از عضوهای habitat فاخته با احتمال مشخصی (که به فاصلهٔ فاخته با مقصد وابسته است) با عضو متناظرش در habitat مقصد جایگزین می شود و habitat بدست آمده، محل استقرار فاخته پس از مهاجرت است.

در پایان اجرای مراحل الگوریتم فاخته، با استفاده از کلاس Partition اجتماعات تشکیل شده در گراف توسط بهترین فاخته مشخص می شوند و به همراه مقدار Q این فاخته برگردانده می شوند.

در روش حل این مسئله از مقاله زیر استفاده شده است:

Mahmoudi, Shadi; Lotfi, Shahriar; Modified cuckoo optimization algorithm (MCOA) to solve graph coloring problem; 2014

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

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