درس هوش مصنوعی

تمرین شماره ۲: بررسی الگوریتم ژنتیکی مرکب (hybrid) با جستجوی تپه‌نوردی

هدف از این تمرین پیاده‌سازی یک الگوریتم جستجو است که با ترکیب الگوریتم ژنتیکی و جستجوی تپه‌نوردی به صورت توضیح داده شده در زیر اقدام به کاوش در فضای حالت‌های ممکن یک مسأله می‌کند.

توصیف الگوریتم

در این الگوریتم حالت‌ها یا افراد (individuals) جمعیت به صورت بیتی (دودویی) نمایش داده می‌شوند. جمعیت اولیه به صورت تصادفی با توزیع یکنواخت روی مقادیر هر یک از ژن‌ها (متغیرها) تولید می‌شود. در هر نسل از الگوریتم، برای هر فرد بر اساس برازندگی آن یک احتمال انتخاب محاسبه شده و سپس درصد مشخصی از افراد (#ParentPercent) بر اساس این احتمال‌های محاسبه شده برای بازتولید (reproduction) به عنوان والدین انتخاب می‌شوند. از بین افراد انتخاب شده به تعداد نصف فرزندان مورد نیاز (#OffspringPercent) زوج در نظر گرفته می‌شود که برای تشکیل هر زوج از بین والدین، دو فرد به صورت تصادفی انتخاب می‌شوند (جفت‌سازی تصادفی). پس از جفت‌سازی، بر روی هر زوج ابتدا با احتمال در نظر گرفته شده (#CrossoverProb) عملگر تقطیع دو نقطه‌ای (two-point crossover) اعمال شده و دو فرزند تولید می‌شود. نقاط تقطیع در هر زوج به صورت تصادفی انتخاب می‌شوند. سپس به هر یک از ژن‌های هر یک از فرزندان به صورت جداگانه با احتمال داده شده (#MutationProb) عملگر جهش اعمال می‌شود. پس از آن هر یک از فرزندان با استفاده از جستجوی تپه‌نوردی در صورت امکان بهبود داده می‌شود. برای اینکار جستجوی تپه‌نوردی با شروع از یک فرد در همسایگی آن به دنبال افراد بهتر می‌گردد و اینکار را تا جایی ادامه می‌دهد که فرد بهتری پیدا نکند یا به تعداد دفعات مشخصی (#MaxImprove) فرد مورد نظر بهبود داده شده باشد. منظور از همسایگان یک فرد، افرادی هستند که با تغییر مقدار فقط یک ژن بدست می‌آیند. جستجوی تپه‌نوردی دارای امکان حرکت به کنار (sideway moves) با بکارگیری یک لیست Tabu به اندازه مشخص شده (#TabuSize) را دارد و حداکثر حرکت‌های به کنار نیز به الگوریتم داده می‌شود (#MaxSideway). در پایان این نسل از الگوریتم، بهترین افراد از مجموعه فرزندان بهبود داده شده و افراد موجود در جمعیت بر اساس برازندگی انتخاب شده و به عنوان افراد جمعیت در نسل بعد در نظر گرفته می‌شوند (جایگزینی). این فرآیند تا برآورده شدن شرط توقف که همگرا شدن جمعیت (یکسان شدن تمام افراد جمعیت) یا رسیدن به حداکثر نسل‌های ممکن (#MaxGen) است، ادامه می‌یابد.

ورودی‌ها و خروجی‌های الگوریتم

پارامترهای تعیین کننده عملکرد الگوریتم به شکل یک فایل ورودی بنام "params.txt" به برنامه پیاده‌سازی شده داده می‌شود (برای اینکار امکان تهیه یک فرم در صورت وجود واسط گرافیکی برای برنامه نیز هست). سطرهای این فایل باید شامل پارامترهای مشخص شده در جدول1 باشد.

علاوه بر این، الگوریتم باید یک تابع برازندگی به صورت $y=f(x)$ برای ارزیابی راه‌حل‌های مختلف مسأله در اختیار داشته باشد (که در آن x نمایش بیتی یک فرد و y میزان برازندگی آن است). دقت کنید برای حفظ جامعیت الگوریتم پیاده‌سازی شده تغییر چنین تابعی برای مسائل مختلف باید با سهولت قابل انجام باشد و بنابراین تابع برازندگی باید یک پیمانه (module) جدا از الگوریتم اصلی بوده و فقط توسط آن الگوریتم فراخوانی شود. پیاده‌سازی که دارای این ویژگی نباشد فاقد ارزش خواهد بود.

الگوریتم پیاده‌سازی شده باید یک فایل ثبتی (log) ایجاد کند (و در صورت امکان در خروجی نشان دهد) که برای هر نسل شامل: بهترین فرد جمعیت، برازندگی بهترین فرد، میانگین برازندگی افراد جمعیت و تعداد دفعات فراخوانی تابع برازندگی تا آن لحظه را نشان دهد. زبان برنامه‌نویسی برای پیاده‌سازی الگوریتم به دلخواه است.

جدول1 . پارامترهای ورودی الگوریتم

• IndivSize xxx	یک مقدار صحیح نشان دهنده طول رشته بیتی
• PopSize xxx	یک مقدار صحیح نشان دهنده اندازه جمعیت مورد استفاده در الگوریتم
• ParentPercent xxx	یک مقدار اعشاری در (۰, ۱) نشان دهنده نسبت والدین انتخاب شده به اندازه جمعیت
• OffspringPercent xxx	یک مقدار اعشاری در (۰, ۱) نشان دهنده نسبت فرزندان تولید شده به اندازه جمعیت
• MaxGen xxx	یک مقدار صحیح نشان دهنده حداکثر تعداد نسل‌های (تکرارها) الگوریتم
• CrossoverProb xxx	یک مقدار اعشاری در (۰, ۱) نشان دهنده احتمال تقطیع
• MutationProb xxx	یک مقدار اعشاری در (۰, ۱) نشان دهنده احتمال جهش
• TabuSize xxx	یک مقدار صحیح نشان دهنده اندازه لیست تابو
• MaxSideway xxx	یک مقدار صحیح نشان دهنده حداکثر تعداد حرکت‌های به کنار
• MaxImprove xxx	یک مقدار صحیح نشان دهنده حداکثر تعداد دفعات بهبود هر فرزند

مسأله کوله‌پشتی

چنین الگوریتمی می‌تواند برای حل مسائل مختلفی مورد استفاده قرار گیرد. به عنوان یک کاربرد ساده و برای بررسی عملکرد الگوریتم در این تمرین، از مسأله معروف کوله‌پشتی (knapsack) استفاده می‌شود که در آن باید از بین تعدادی کالا (item) که هر یک دارای ارزش (value) و وزن (weight) بخصوصی است، زیر مجموعه‌ای انتخاب شود که دارای بیشترین ارزش باشد ولی وزن آن از حداکثر وزن قابل حمل توسط کوله‌پشتی تجاوز نکند. بنابراین تابع هدف در نظر گرفته شده برای تعیین برازندگی افراد در این مسأله به صورت زیر خواهد بود:

$$y = f(\vec{x}) = \begin{cases} \sum_{i=1}^{M} x_i \cdot value(i), & \sum_{i=1}^{M} x_i \cdot weight(i) \le MaxWeight \\ 0, & otherwise \end{cases}$$

که در آن M تعداد کالاها و برابر با طول رشته‌های بیتی در نظر گرفته شده برای نمایش افراد است. $MaxWeight$ نشان دهنده حداکثر وزن قابل قبول برای کوله‌پشتی است. سه نمونه از این مسأله که باید در این تمرین مورد استفاده قرار گیرد در فایل‌های پیوست آورده شده است. در هر فایل سطر اول به ترتیب تعداد کالاها (M) و حداکثر وزن مجاز ($MaxWeight$) را نشان می‌دهد. سایر سطرها که تعداد آنها M است به ترتیب نشان دهنده ارزش و وزن هر یک از کالاها خواهد بود. مقادیری که باید برای پارامترهای الگوریتم در هر یک از این نمونه مسائل در نظر گرفته شود در جدول2 آورده شده است.

بخش اختیاری با نمره اضافی

در صورت علاقه دانشجویان می‌توانند نتیجه اعمال الگوریتم A* را نیز بر روی سه نمونه پیشتر گفته شده از مسأله کوله‌پشتی بررسی کرده و با نتایج بدست آمده از الگوریتم پیاده‌سازی شده مقایسه کنند. در این صورت حالت اولیه می‌تواند برابر با کوله‌پشتی خالی باشد و هر کنش شامل اضافه کردن یکی از کالاهای انتخاب نشده یا حذف یکی از کالاهای انتخاب شده در کوله پشتی باشد. آزمایش هدف می‌تواند بررسی برابری ارزش کوله پشتی با مقدار بهینه ارزش (و رعایت محدودیت وزن کوله‌پشتی) بوده و

هزینه مسیر وزن فعلی کوله پشتی باشد. یکی از گزینه‌های ممکن برای تابع اکتشافی (heuristic) می‌تواند قدرمطلق تفاوت ارزش فعلی کوله‌پشتی از مقدار بهینه ارزش نسبت به حداکثر وزن مجاز کوله‌پشتی ($|value(knapsack) - value^*|/MaxWeight$) باشد (تعریف ارائه شده فقط یک تعریف پیشنهادی برای مسأله است و هر تعریف صحیح دیگری از مسأله که به حل مؤثرتر آن توسط الگوریتم A^* کمک کند می‌تواند در این بخش در نظر گرفته شود).

جدول2. مقادیر پارامترهای الگوریتم روی نمونه‌های مختلف مسأله کوله پشتی

	نمونه اول (ks_20_878)	نمونه دوم (ks_100_997)	نمونه سوم (ks_200_1008)
IndivSize	20	100	200
PopSize	50	150	150
ParentPercent	0.5	0.5	0.5
OffspringPercent	0.5	0.5	0.5
MaxGen	300	500	500
CrossoverProb	0.8	0.8	0.8
MutationProb	0.05	0.01	0.01
TabuSize	5	5	5
MaxSideway	5	10	15
MaxImprove	10	30	50
ارزش جواب بهینه	1024	2397	1634

نکات مدنظر در تحویل تمرین

- برای انجام این تمرین دانشجویان باید پس از پیاده‌سازی الگوریتم ترکیبی توصیف شده، آن را حداقل پنج بار روی هر یک از نمونه‌های مسأله داده شده اعمال کرده و پس از آن میانگین عملکرد الگوریتم در تمام پنج اجرا را برای هر نمونه مسأله بر اساس موارد زیر گزارش دهند:
  - میانگین بهترین برازندگی بدست آمده
  - میانگین متوسط برازندگی بدست آمده (بر اساس جمعیت آخرین نسل الگوریتم)
  - میانگین دفعات فراخوانی تابع برازندگی
- این تمرین باید به صورت انفرادی انجام شود و هرگونه کپی برداری از کار دیگر دانشجویان به منزله تقلب است.
- دانشجویان می‌توانند جهت پیاده‌سازی الگوریتم از زبان برنامه‌نویسی دلخواه استفاده کنند. همچنین استفاده از کدها و بسته‌های نرم‌افزاری موجود به شرطی امکان پذیر است که مرجع آن ذکر شده و دقیقاً الگوریتم توصیف شده در این تمرین (با تمام جزئیات) پیاده‌سازی و بررسی شود. اجرای صحیح و کامل برنامه پیاده‌سازی شده در ارزشیابی این تمرین نقش اساسی خواهد داشت.
- دانشجویان باید تمام فایل‌های کد و اجرایی برنامه را به همراه یک فایل گزارش در قالب pdf که حداقل حاوی اطلاعات زیر باشد به صورت فشرده شده (به صورت یک فایل zip) تحویل دهند:
  - مشخصات فردی (نام، شماره دانشجویی)
  - توضیحات در مورد عملکرد میانگین الگوریتم (و در صورت وجود مقایسه آن با الگوریتم A^*)
  - توضیحات کافی در مورد نحوه پیاده‌سازی و اجرای الگوریتم

- در صورت لزوم شرح برنامه‌های مورد نیاز برای اجرا و چگونگی نصب، راه‌اندازی و اجرای برنامه

هر گونه سوال در این زمینه را از طریق ایمیل professor.karshenas@gmail.com مطرح کنید.

موفق باشید

کارشناس