تکلیف سری اول درس هوش مصنوعی — "هوالعلیم" — مریم سعیدی ۹۴۲۹۴۷۲

سوال اول :

$X = abcdefgh$

$f(x) = (a+b)-(c+d)+(e+f)-(g+h)$

$x_1 = 65413532$ ، $x_2 = 87126601$ ، $x_3 = 23921285$ ، $x_4 = 41852094$

الف) $f(x_1) = (6+5)-(4+1)+(3+5)-(3+2) = 9$

$f(x_2) = (8+7)-(1+2)+(6+6)-(0+1) = 23$

$f(x_3) = (2+3)-(9+2)+(1+2)-(8+5) = -16$

$f(x_4) = (4+1)-(8+5)+(2+0)-(9+4) = -19$

برحسب برازندگی $x_4 < x_3 < x_1 < x_2$

$\Rightarrow$ برازندگی کل $= -3$

ب) single-point crossover : دو برازنده ترین ترکیب شوند.

$x_2 = 871$	26601	$y_1 = 871$	13532
$x_1 = 654$	13532	$y_2 = 654$	26601

two-point Crossover : خرد دوم و سوم ترکیب شوند.

$x_2 = 871$	26	601	$y_3 = 871$	21	601
$x_3 = 239$	21	285	$y_4 = 239$	26	285

Uniform Crossover : خرد اول و سوم ترکیب شوند (برای هر بیت، سکه انداختم)

$x_1 = 6$	5	4	1	3	5	3	2	
$x_3 = 2$	3	9	2	1	2	8	5	
$y_5 = 6$	5	4	1	3	2	8	5	
$y_6 = 2$	3	9	2	1	5	3	2	

پ)

$f(y_1) = (8+7)-(1+1)+(3+5)-(3+2) = 16$

$f(y_2) = (6+5)-(4+2)+(6+6)-(0+1) = 16$

$f(y_3) = (8+7)-(1+2)+(1+6)-(0+1) = 18$

$f(y_4) = (2+3)-(9+2)+(6+2)-(8+5) = -11$ ×

$f(y_5) = (6+5)-(4+1)+(3+2)-(8+5) = -2$

$f(y_6) = (2+3)-(9+2)+(1+5)-(3+2) = -5$ ×

$\Rightarrow$ برازندگی کل $= 48$

بله برازندگی کل جمعیت جدید (48) بیشتر از جمعیت اولیه (3-) افزایش چشمگیری داشته است.

ت) $\max(f(x)) = (9+9)-(0+0)+(9+9)-(0+0) = 36$

بیشینه $X = 99009900$

ث) از 2-swap میتوان استفاده کرد. همچنین میتوان Uniform-mutation (در جایی که real-value باشد) را به گونه ای تعریف کرد که $\{1,2,3,\dots,9,0\} = n$ و حالا یک موضع به تصادف انتخاب شده و یک عدد از دامنه ی n انتخاب شده و در موضع منتخب جایگزین رقم فعلی شود.

ج) خیر. در single-point crossover قرار است یک برش داشته و چهار قطعه ایجاد شده را طبق تعریف، به جای جایگزین یکدیگر کنیم. حالا یک نگاه به ژن اول (a) جمعیت اولیه کامنیت تا بفهمم اصلا در این ژن، رقم 9 نداشتیم و حالا هرچند هم این single-point Crossover را اجرا کنیم، محال است در ژن a، 9 ظاهر شود پس محال است به جواب بهینه یعنی 99009900 برسیم و فقط mutation لازم است!

سوال دوم:

الف) الگوریتم شبیه‌سازی نوب فلزات با $T=0$: $e^{\frac{\Delta E}{T}} \to 0$ درنتیجه کاملاً حریصانه داریم رفتار می‌کنیم. مثل الگوریتم HC روش steepest descent/ascent.

ب) الگوریتم شبیه‌سازی نوب فلزات با $T=\infty$: $e^{\frac{\Delta E}{T}} \to 1$ درنتیجه کاملاً تصادفی داریم رفتار می‌کنیم. مثل الگوریتم random walk.

پ) الگوریتم جست‌وجوی پرتوی محلی با $K=1$: همان HC است روش steepest descent/ascent

ت) الگوریتم جست‌وجوی پرتوی محلی با $K=\infty$: این دیگه عملاً داره کل فضای حالت را جست‌وجو می‌کند :D مثل BFS (سطح‌به‌سطح جلو، همه‌ی همسایه‌ها را می‌بینه و بعد دوباره برای تمام نوه‌های قبلی، تمام همسایه‌ها ... الی آخر.)

ث) الگوریتم ژنتیک وقتی که هر نسل فقط شامل یک نفر باشد: چون فقط یک نفر در هر نسل داریم اصلا crossover معنا ندارد و الگوریتم با جهش ادامه پیدا می‌کند و چون جهش یک حرکت تصادفی است و عملاً انتخابی برای نسل بهتر یا بدتر نداریم، پس عملاً نشستیم به امید جهش (یک حرکت تصادفی) ← می‌شه الگوریتم random walk.

سوال سوم:

الف) با الگوریتم HC
- هدف: یافتن مسیر با کمترین هزینه.
  - در این مسیر ارائه شده، هزینه برابر با 12 که یک Global Optima هست و بهینه‌ترین جواب ممکن.
- هدف: یافتن مسیر با بیشترین پاداش:

$A_6 \to B_3 \to G_2 \to G_1 \to F_0$

$A_6 \xrightarrow{2} B_3 \xrightarrow{2} C_3 \xrightarrow{2} D_3 \xrightarrow{3} G_1 \xrightarrow{5} E_1 \xrightarrow{6} F_0$

$D_3 \xrightarrow{3} G_2 \xrightarrow{1} G_1 \xrightarrow{5} E_1 \xrightarrow{6} F_0$

از D_3 به طور تصادفی یکی مسیر را پیش می‌گیرد. اگر فرض کنیم شاخه پایین را انتخاب کند پس:

$A_6 \xrightarrow{2} B_3 \xrightarrow{2} C_3 \xrightarrow{2} D_3 \xrightarrow{3} G_2 \xrightarrow{1} G_1 \xrightarrow{5} E_1 \xrightarrow{6} F_0$

که در این مسیر پاداش برابر با 21 است و یک local Optima است.
بهترین پاداش $+\infty$ است. (اگر از گره E_1 با بال صفر به I_4 می‌رفت در حلقه‌ای شامل نوده‌ای $(E_1, I_4, H_5, G_2, G_1)$ گیر می‌افتاد!)

ب) با الگوریتم TS
- هدف: یافتن مسیر با کمترین هزینه. دقیقاً مشابه مثال قبل است $A6 \to B3 \to G2 \to G1 \to F0$ و در Global Optima متوقف می‌شود.
- هدف: یافتن مسیر با بیشترین پاداش. این حالت هم مشابه قبل است. باز از دو مسیر فوق به تصادف انتخاب می‌شود. فقط احتمال افتادن در لوپی که گفتم از بین نمیره کاملاً. در local Optima متوقف نمی‌شود.

پ) ماکزیمم محلی یا سراسری؟ در صورت مشخص نشده.

ت) الگوریتم برای ارضای هر دو شرط به صورت همزمان: باید دقیقاً تابع هدف مشخص شود. بین این دو trade off وجود دارد و شاید مجبور باشیم اولویت رعایت کنیم. به هرحال باید تابع هدف مشخص باشد؛ ولی اگر منظور این است که مسیری همزمان بهترین باشد از لحاظ هزینه‌ی کم و پاداش زیاد، وجود ندارد. بدیهیست که کم‌هزینه‌ترین مسیر (هزینه 12) یک Optima سراسری بود

ادامه قسمت (ت) سوال سوم: در حالی که بهترین مسیر از نظر پاداش، حاوی یک لوپ است و پاداش را این مسیر بهینه برابر است با $2+2+2+3+5+(0+2+4+1+5)n+6 = +\infty$.

اما اگر مثلاً تابع هدف را به صورت: $f(x) = \frac{\text{پاداش}}{\text{هزینه}}$ معرفی کنیم که در این صورت مسئله maximization است. و کافیست با الگوریتم های local search و تابع هدف $f(x)$ که قرار است max شود، مسئله را حل کنیم.

یا مثلاً تابع هدف را به صورت: $f(x) =$ پاداش − هزینه معرفی کنیم و هدف max کردن $f(x)$ باشد.

در هر حال برای تابع هدف $\frac{\text{پاداش}}{\text{هزینه}}$، HC و TS هر دو یک جواب می دهند:

$A_6 \to B_3 \to C_3 \to D_3 \to G_1 \to F_0$

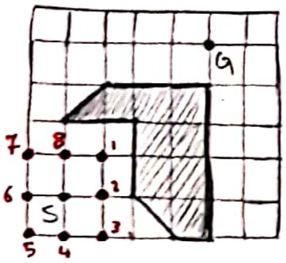
برای تابع هدف دوم (هزینه − پاداش)، با اینهم HC و TS یک جواب می دهند:
(جاهایی که مسیر دو شاخه می شود، و نمی دانم تقسیم می کنیم به کجا برویم.)

$A_6 \to B_3 \to C_3 \to D_3 \to G_1 \to F_0$
  $G_1 \to E_1 \to F_0$
  $B_3 \to G_2 \to G_1 \to F_0$
  $G_1 \to E_1 \to F_0$

سوال چهارم:

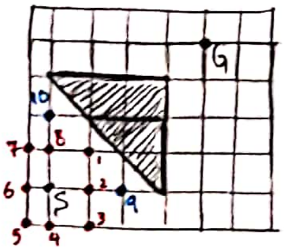
الف) چگونگی عملکرد الگوریتم HC: متناسب با تعریفی که از همسایگی برای مسئله در نظر می گیریم، هر بار می آید همسایه ها را پیدا کرده و آنی که به مقصد (G) نزدیک تر است را انتخاب می کند. مثلاً اگر منظور از همسایگی، آن هایی باشد که با فاصله ی یک از نقطه فعلی هستند، 8 همسایه برای S پیدا می کنیم که از این بین آنی که به G نزدیک تر است را انتخاب می کنیم و به آن نقطه می رویم.

ب) مثالی برای گیر کردن در بهینه محلی با موانع غیرمحدب:



در مثال روبه رو یک مانع غیرمحدب داریم. همسایگی را هم نقاط با فاصله ی یک از نقطه فعلی تعریف کردم. می بینیم که 8 همسایه داریم. که از بین این هشت تا، همسایه شماره 1 به G نزدیک تر است و اگر به آن برویم بعد 3 همسایه داریم که همین شماره های 8 و 2 و S خواهند بود و هیچ کدام ارتقادهنده نیستند و درنتیجه الگوریتم در این local optima گیر می افتد.

پ) گیر افتادن با موانع محدب؟

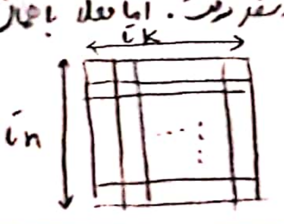


در مثال روبه رو یک مانع محدب داریم. همسایگی را هم نقاط با فاصله ی یک از نقطه ی فعلی تعریف کرده ام. می بینیم که 8 همسایه داریم که از این بین، همسایه شماره 1 به G نزدیک تر است و اگر به آن برویم بعد 5 همسایه داریم (9، 2، S، 8، 10) که ارتقادهنده نیستند و الگوریتم همین جا متوقف می شود.

ت) آیا SA کمک کننده است برای فرار از lo ها؟ از آنجایی که این الگوریتم در مواضعی که همسایه ارتقادهنده نداریم، به صورت تصادفی عمل می کند، به ما امکان خارج شدن از local optima را می دهد تا به این ترتیب باز هم شانس رسیدن به بهینه سراسری را داشته باشیم.

سوال پنجم:

الف) Representation: یک ماتریس باینری $n \times k$ (n راس ها و k خوشه ها) به شرطی که حداقل یک "یک" در هر ستون و دقیقاً یک "یک" در هر سطر باشد. (هر راس فقط متعلق به یک خوشه است و حتماً باید k خوشه داشته باشیم) می توان شرط ها را برداشت و جریمه برای نقض آن در نظر گرفت. اما فعلاً با همان شرط ها ادامه می دهیم.



ادامه‌ی سوال پنجم :

ب) تعداد کل اعضای فضای جست‌وجو : اول با K راس ، در هر خوشه یک راس می‌گذاریم که مطمئن شویم دقیقاً k خوشه داریم . حالا از (n-k) راس باقی‌مانده ، هر راس می‌تواند در یکی از K خوشه باشد . لذا برای این n-k راس باقی‌مانده K^{n-K} حالت داریم . (تا اینجا $\binom{n}{K} K^{n-K}$) . حالا از بین این حالات باید مواردی که دو سرال در خوشه‌های متفاوت باشند را کم کنیم (فرض کنیم UV یک یال باشد ، یکبار U در خوشه یک و V در خوشه دو است در حالت دیگری V در خوشه دو و U در خوشه یک است ولی در دو حالت از آنجایی که دسته‌ها حائز اهمیت نیستند یکسان محسوب می‌شوند پس باید این موارد از حالات کلی کم شود) خلاصه تعداد کل فضای جست‌وجو : $\frac{\binom{n}{K} K^{n-K}}{2}$

ب) تعریف همسانگی : به نوعی همسانگی جدیدی تعریف می‌کنیم به این شکل که :
خوشه‌ی راس را عوض کنیم به این شرط که تعداد خوشه‌ها از K کمتر نشود (اگر خوشه‌ای تنها یک راس داشت ، این راس بلاک می‌شود و حق تغییر خوشه ندارد .)
در مورد representation گفته شده ، یک سطر را (یعنی یک راس را) انتخاب کرده و اول چک می‌کنیم تک‌راس در خوشه نباشد (در ستون مربوط به آن اگر تنها یک مورد یعنی نباید به آن راس دست بزنیم .) و پس خوشه مربوط آن را عوض می‌کنیم اگر راس بلاک شده نبود .

ت) ماکزیمم و مینیمم تعداد همسایه‌ها :
* کمینه تعداد همسانگی : شرایطی را فرض می‌کنیم که (K-1) خوشه هرکدام با یک راس داریم و یک خوشه با (n-k+1) راس . در این شرایط قطعا از بین (n-k+1) راس می‌توان یک راس را برای تغییر خوشه انتخاب کنیم پس کلا : (n-k+1)x(K-1) همسانگی داریم .
* بیشینه تعداد همسانگی : شرایطی که می‌توانیم هر راس دلخواهی را تغییر خوشه دهیم ، مطلوب را می‌دهد . پس در کل تعداد بیشینه همسانگی برابر است با : n x (K-1) .

ث) در این حضور (البته به راهنمایی TA) در مورد الگوریتم‌های Graph clustering تحقیق کردم و متوجه شدم این مسئله ، یک مسئله‌ی NP-Complete هست و الگوریتم‌های Exact ندارند . همچنین مسئله‌ی این سوال هم بسیار شبیه Graph clustering است و از آن هم الگوریتم Exact ندارد .

سوال ششم

کد