دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

هوش مصنوعی
بهار ۱۴۰۰
استاد: محمدحسین رهبان

مباحث فصل اول تا چهارم | **پاسخ‌نامه‌ی میانترم** | مهلت ارسال: ـ

- مسائل لزوما یک پاسخ ندارند و نمره‌ی کامل به هر پاسخ صحیحی به جز پاسخ ذکر شده در این مستند تعلق خواهدگرفت.

مسائل (۱۰۰+۱۰ نمره)

۱. (۳۲ نمره)

(آ) صحیح، فرض کنید در iteration شماره k ام هستیم و تا به حال m حالت را دیده ایم. پس جست‌وجوی کامل تا عمق $k-1$ انجام شده است. بدترین حالت این است که مستقیم از راس اصلی (root) به یک گره در عمق k رفته باشیم و $O(bk)$ تا حالت در frontier باشند. این مقدار برحسب k بوده در حالی که در سوال مقدار m در اختیارمان گذاشته شده‌است. از آن جایی که IDS اگر در عمق d باشد تعداد $O(b^d)$ تا را مشاهده کرده پس می‌توان گفت k از مرتبه‌ی $O(log(m))$ است. بنابراین با جایگزاری اندازه‌ی frontier از مرتبه‌ی $O(b\,log(m))$ خواهدبود.

(ب) غلط، در صورتی که heuristic مناسبی در اختیار الگوریتم A^* گذاشته نشده‌باشد، عملکرد دو الگوریتم می‌تواند یک‌سان بوده و هر دو تعداد گره‌های یکسانی را بررسی کنند.

(ج) صحیح، تصور کنید مقدار هزینه‌ی واقعی باقی‌مانده تا هدف برای حالت i برابر با $h^*(i)$ باشد. هر دو تابع h_1 و h_2 admissible هستند، بنابراین خواهیم داشت:

$$h_1(i), h_2(i) \leq h^*(i)$$

$$\alpha h_1(i) \leq \alpha h^*(i)\ \&\ (1-\alpha)h_2(i) \leq (1-\alpha)h^*(i)$$

بنابراین تابع heuristic مطرح شده نیز admissible می‌باشد، چرا که داریم:

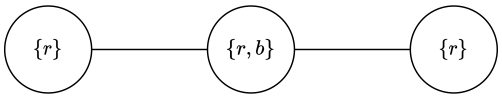
$$\alpha h_1(i) + (1-\alpha)h_2(i) \leq h^*(i)$$

(د) صحیح، با کاهش دما عمل‌کرد الگوریتم simulated annealing به گونه‌ای خواهد بود که در هر گام خود یا به حالتی بهتر رفته و یا در همان‌حالت باقی می‌ماند. این عملکرد مشابه عملکرد الگوریتم hill climbing بوده که خود حالت خاصی از الگوریتم local beam-search با $k=1$ می‌باشد.

(ه) غلط، برای مثال تصور کنید می‌خواهیم تابع $f(x)=x^2$ را با شروع از نقطه‌ی $x_0=1$ به کمک الگوریتم gradient descent بهینه‌کنیم. در صورتی که ضریب یادگیری برابر ۱ در نظر گرفته شود این با توجه به آنکه مشتق این تابع نسبت به x برابر $f'(x)=2x$ است، الگوریتم بهینه‌سازی در هر مرحله دو واحد x را تغییر داده و خواهیم داشت $x_t=(-1)^t$. بنابراین هرگز به نقطه‌ی کمینه‌ی تابع یعنی $x^*=0$ دست نمی‌یابد و هرگز همگرا نخواهدشد.

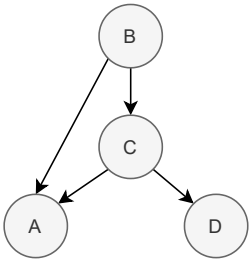
(و) غلط، الگوریتم alpha,beta-pruning تنها روشی برای محاسبه‌ی سریع‌تر مقدار راس اصلی (root) بوده و اعمال آن، با توجه به احتمال رخ‌داد هرس شاخه‌های مختلف، حد پایین و یا بالایی از مقدار زیرشاخه‌های درخت minimax محاسبه می‌کند که لزوما با مقدار دقیق یکسان نیست.

(ز) غلط، به مثال نقض شکل ۱ توجه کنید.



شکل ۱: در صورتی که محدودیت مسئله آن باشد که رؤوس مجاور نمی‌توانند هم رنگ باشند، فرض بیان شده در این گراف محدودیت مشخصا برقرار نیست.

(ح) غلط، به مثال نقض شکل ۲ توجه کنید.



شکل ۲: در این شبکه‌ی بیز تنها استقلال‌های یادشده برقرار اند.

۲. (۱۵ نمره)

(آ) اگر k تعداد خانه های دیوار کشی شده باشد تعداد کل حالت های ممکن برابر می‌شود با:

$$۲^{MN-k}(MN-k)(MN-k-۱)$$

در واقع هر یک از خانه‌هایی که دیوارکشی نشده‌اند یا فروریخته‌اند یا نه پس دو حالت دارند. برای خانه ای هایی که افراد در آن قرار دارند نیز $(MN-k)(MN-k-۱)$ حالت وجود دارد. پس فضای حالت مسئله از مرتبه‌ی $O(۲^{MN}(MN)^{۲})$ است.

(ب) در ابتدا انتخاب ها بین بالا پایین چپ راست و ثابت ماندن است (۵ حالت برای هر کدام). پس از آن همیشه یکی از جهات غیرقابل دسترس خواهد بود چرا که خانه ای که فرد قبلا در آن قرار داشته فروریخته (۴ حالت برای هر کدام) پس ضریب انشعاب برابر است با $۴ \times ۴ = ۱۶$.

(ج) کمینه‌ی فاصله یا همان manhattan distance هر کدام از افراد تا نقطه خروجشان را در نظر می‌گیریم. قطعا این هیورستیک overestimate نمی‌کند چرا که به هر حال هر دو قرار است خارج شوند و حداقل به اندازه فاصله تا مقصد باید حرکت کنند. بنابراین این heuristic، admissible است.

۳. (۱۵ نمره) فرض کنید متغیرهای منطقی مساله $v_۱, v_۲, ..., v_k$ باشند. هر کروموزوم را به صورت رشته k بیتی حاصل از چسباندن متغیرهای منطقی مساله به یکدیگر به صورت $\overline{v_۱v_۲...v_k}$ تعریف می‌کنیم که $\forall_{۱\leq i\leq k} : v_i \in \{۰, ۱\}$.

دقت کنید تناظری یک به یک میان کروموزوم‌ها و حالت‌های متفاوت مقداردهی به متغیرهای منطقی وجود دارد.

با توجه به تعریف فوق ۲^k کروموزوم متفاوت خواهیم داشت و بنابراین سایز فضای جستجوی مساله ۲^k است.

در ابتدا به عنوان جمعیت اولیه، تعدادی (پارامتر طراحی) کروموزوم را از مجموعه تمام کروموزوم‌های ممکن با احتمال یکسان برمی‌گزینیم. برای این کار کافی است رشته‌ای k بیتی تولید کنیم که هر بیت از آن با احتمال ۰.۵ مقدار صفر و با احتمال ۰.۵ مقدار یک را به خود بگیرد (طراح مدل می‌تواند امکان تکراری بودن یا نبودن کروموزوم تولید شده را انتخاب نماید).

می‌دانیم هر عبارت k-sat به صورت ترکیب عطفی تعدادی عبارت فصلی نوشته می‌شود. حال تابع $fitness$ را برای هر کروموزوم به صورت تعداد عبارت‌های فصلی ارضا شده توسط مقداردهی متناظر متغیرهای منطقی به ازای آن کروموزوم در نظر می‌گیریم.

دقت کنید مطلوب ما آن است که به کروموزومی دست یابیم که $fitness$ آن دقیقا برابر تعداد عبارات فصلی شود. بنابراین به ازای هر کروموزوم هرچه مقدار $fitness$ بهتر باشد، کروموزوم پایدارتر بوده و احتمال انتخاب آن (پارامتر طراحی) برای زاد و ولد بیشتر خواهد بود.

حال فرض کنید دو کروموزوم $\overline{u_1u_2...u_k}$ و $\overline{v_1v_2...v_k}$ برای مرحله $crossover$ برگزیده شده‌اند. برای $crossover$ کردن این دو کروموزوم با توجه به اینکه ترتیب متغیرهای منطقی چیده شده در این رشته بیت معنادار نیست، زیرمجموعه ای مانند I (طراح مدل می‌تواند ناتهی یا سره بودن این زیرمجموعه را شرط کند) از مجموعه $\{1, 2, ..., k\}$ را با احتمال یکسان انتخاب می‌کنیم و سپس دو کروموزوم فرزند را با جابجا کردن مقادیر بیت‌های با اندیس‌های عضو I از دو کروموزوم پدر و مادر تولید می‌کنیم.

برای نمونه: فرض کنید کروموزوم‌های پدر و مادر به ترتیب ۰۱۰۱۰ و ۱۱۰۰۱ باشند. همچنین فرض کنید $I = \{2, 4, 5\}$ باشد. در این صورت کروموزوم‌های فرزند به صورت ۰۱۰۰۱ و ۱۱۰۱۰ خواهند بود.

برای محدود کردن انتخاب I می‌توانیم شرط‌هایی را تعریف نماییم. برای نمونه برای آنکه در انتخاب I به گونه‌ای عمل کنیم که سعی کنیم اندیس‌های نزدیک‌تر به هم (با توجه به حضورشان در کنار یکدیگر در ترکیب‌های فصلی) همزمان یا در I حضور داشته باشند یا نداشته باشند می‌توانیم گراف همسایگی متغیرهای منطقی (متغیرهای منطقی رئوس گراف هستند و دو راس به هم یال دارند اگر و تنها اگر در یک عبارت فصلی با یکدیگر آمده باشند) را رسم کرده و در هر گام به صورت زیر عمل کنیم.

- یک عضو از I را به صورت تصادفی با احتمال یکسان انتخاب کرده و مجموع فاصله آن با اعضای درون I و اعضای خارج از I را محاسبه کرده و در صورتی که مجموع اول از مجموع دوم بیشتر بود، آن را از I خارج نماییم.
- یک عضو خارج از I را به صورت تصادفی با احتمال یکسان انتخاب کرده و مجموع فاصله آن با اعضای درون I و اعضای خارج از I را محاسبه کرده و در صورتی که مجموع دوم از مجموع اول بیشتر بود، آن را به I اضافه نماییم.

حداکثر تعداد انجام این گام پارامتر طراحی است. این محدودسازی را می‌توان با صرفا در نظرگیری گراف همسایگی متغیرهای منطقی در ترکیب‌های فصلی ارضا شده در یکی از کروموزوم‌های پدر یا مادر یا هر دو، محدودتر نیز کرد. (انتخاب میان I بدون محدودیت یا با محدودیت توسط طراح مدل انجام می‌شود)

جهش در کروموزوم را به صورت $flip$ کردن یک بیت تعریف می‌کنیم. بدین ترتیب با یک احتمال (پارامتر طراحی) مشخص ممکن است ژن های یک فرزند تغییر بکنند. این تغییر به صورت $flip$ شدن یک یا چند بیت از کروموزوم فرزند خواهد بود. دقت شود احتمال آنکه n ژن در یک فرزند دچار جهش شوند با افزایش n نزولی است. جهش در جمعیت را می‌توانیم به صورت افزودن یک کروموزوم جدید (تولید شده همانند مرحله تولید جمعیت) به جمعیت در انتهای تولید هر نسل، با یک احتمال مشخص (پارامتر طراحی) تعریف نماییم.

این انواع جهش برای اجتناب از افتادن در مقادیر بهینه محلی غیر سراسری هستند.

حال حداکثر تعداد نسل های تولید شده (پارامتر طراحی)، درصد جمعیت تولید شده در هر مرحله با استفاده از $crossover$ که به آن $crossover\ rate$ می‌گویند (پارامتر طراحی) و سایر پارامتر های طراحی را تعیین می‌کنیم.

شرط خاتمه، ارضا شدن تمام عبارات فصلی و یا رسیدن به حداکثر تعداد نسل‌های تولید شده خواهد بود. با نگهداری کروموزوم با حداکثر مقدار $fitness$ در صورت ارضا شدن تمام عبارات فصلی به پاسخ مساله خواهیم رسید.

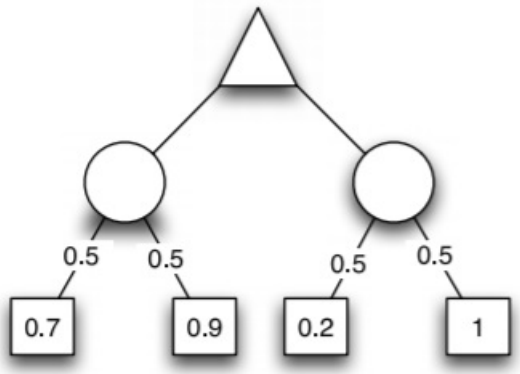
۴. (۱۶ نمره)

(آ) مساله CSP در حالت کلی یک مساله NP-Hard است. البته می توان از heuristic هایی مثل forward checking، least constraining value و ... برای سریعتر کردن الگوریتم استفاده کرد اما در بدترین حالت همچنان از مرتبه زمانی $O(d^n)$ است.
در حالتی که گراف بدون دور باشد، این مساله در زمان چندجمله‌ای قابل حل است. به این صورت که ابتدا با DFS ترتیب topological درخت را پیدا می کنیم و از انتها به ابتدا یالها را consistent می کنیم. چون گراف دور ندارد این کار در یک بار پیمایش درخت ممکن است. سپس از ابتدا به انتها گراف را مقدار دهی می کنیم. این الگوریتم از مرتبه زمانی $O(nd^2)$ است.
** d تعداد کل مقادیر مجاز و n تعداد یالها یا همان متغیرها است.

(ب) واضح است که راس وسط یک cutset است و با حذف آن ۴ درخت به دست می آید. پس ابتدا راس وسط را مقدار دهی می کنیم و با راس های همسایه‌اش مقایسه می کنیم. سپس هر کدام از در خت ها را در مرتبه زمانی $O(nd^2) = O(5d^2) = O(d^2)$ حل می کنیم. چون خود راس وسط نیز d حالت دارد پس مرتبه زمانی کل الگوریتم $O(d^3)$ است.

۵. (۱۶ نمره)

(آ)
- در درخت max هرس کردن امکان‌پذیر نیست. چون در انشعاب بعدی درخت همواره ممکن است مقدار بیشتری از آن چه قبلا دیده‌ایم داشته باشد. در درخت expectimax نیز مشابه حالت قبل هرس کردن امکان‌پذیر نیست.
- در درخت max و expectimax هرس کردن امکان‌پذیر نیست. در نظر گرفتن کران پایین مشکل قسمت الف را حل نمی‌کند.
- برای درخت max فرض کنید یک درخت ماکس با دو فرزند ۱ داشته باشیم. رأس دوم هرس می‌شود. برای درخت expectimax یک مرتب‌سازی از کوچک به بزرگ و از چپ به راست در نظر بگیرید؛ در درخت زیر راست‌ترین برگ هرس می‌شود.



(ب) فرض می‌کنیم راس‌های قبلی که بررسی کرده‌ایم، بیشترین امیدریاضی بدست آمده برابر با m می‌باشد حال برای بدست آوردن مقدار راس جدید هنگامی که به فرزندهای آن نگاه می‌کنیم برای prune کردن تصور می‌کنیم که فرزندهای دیده نشده دارای مقدار ۱ می‌باشند (بیشترین مقدار ممکن) و یک مقدار تخمینی برای بیشینه مقدار راس بدست می‌آوریم. حال اگر این مقدار از m کمتر باشد می‌توان نتیجه گرفت که به هیچ عنوان نمی‌توانیم از m مقدار بیشتری را بدست آوریم و می‌توانیم محاسبات را متوقف کنیم. با توجه به این استدلال هر چه زودتر رئوس با احتمال بیشتر را ببینیم تخمین بهتری از کران بالا بدست می‌آوریم و فرضیاتی که می‌کنیم به دلیل احتمال کمتر رئوس باقی‌مانده به واقعیت نزدیک‌تر می‌شود.

۶. (۱۶ نمره)

(آ) در روش Rejection sampling در هر مرجله از نمونه‌برداری، اگر نمونه انتخاب شده مغایر با evidence مساله باشد آن را رد می‌کنیم.
باتوجه به شبکه داده شده و فرضیات، برای محاسبه احتمال رد داده نمونه‌برداری شده، احتمال متمم evidence ها یعنی احتمال رخداد $P(-a)$ را محاسبه می‌کنیم:

$$P(-a) = ۰/۹$$

بنابراین در ۹۰ درصد حالات نمونه برداشته شده رد می‌شود.

(ب) در روش Likelihood weighting برای جلوگیری از دور ریختن داده‌های نامرتبط در محاسبه query برای هر داده یک وزن باتوجه به evidence داده شده در نظر گرفته می‌شود که این وزن از رابطه زیر محاسبه می‌شود:

$$P(E|parents(E)) = \prod_i P(e_i|parents(e_i))$$

که برای نمونه‌های داده شده، مقدار وزن به صورت زیر می باشد:

$$\omega_{(+a,-b,+c,+d)} = P(+a) \times P(+d|+c) = ۰/۱ \times ۰/۵ = ۰/۰۵$$
$$\omega_{(+a,-b,-c,+d)} = P(+a) \times P(+d|-c) = ۰/۱ \times ۰/۸ = ۰/۰۸$$
$$\omega_{(+a,+b,-c,+d)} = P(+a) \times P(+d|-c) = ۰/۱ \times ۰/۸ = ۰/۰۸$$

حال برای محاسبه $P(-b|+a,+d)$ دو مجموعه زیر را تعریف می‌کنیم:

$$R_۱ = \{+c, -c\}$$
$$R_۲ = \{(+b,+c), (+b,-c), (-b,+c), (-b,-c)\}$$

سپس داریم:

$$P(-b|+a,+d) = \frac{P(+a,-b,+d)}{P(+a,+d)} = \frac{\sum_{i=۱}^{M} \sum_{r_۱ \in R_۱} \left(\omega_{(+a,-b,r_۱,+d)} \times \mathbb{1}\left(x_i = (+a,-b,r_۱,+d)\right)\right)}{\sum_{i=۱}^{M} \sum_{r_۲ \in R_۲} \left(\omega_{(+a,r_۲,+d)} \times \mathbb{1}\left(x_i = (+a,r_۲,+d)\right)\right)}$$

اگر تنها عبارت بالا نیز نوشته شده باشد نمره کامل تعلق می‌گیرد. در نهایت می‌توان با نمونه‌های داده شده مقدار احتمال را به صورت زیر محاسبه کرد:

$$P(-b|+a,+d) = \frac{۰/۰۵ + ۰/۰۸}{۰/۰۵ + ۰/۰۸ + ۰/۰۸} = \frac{۱۳}{۲۱}$$

(ج) برای محاسبه احتمال $P(C|+a)$ چون evidence داده شده یکی از ریشه‌های گراف(راس بدون پدر) می‌باشد، باتوجه به فرض محدود بودن تعداد نمونه‌برداری روش Likelihood weighting بهتر می‌باشد چون روش gibbs نیازمند تعداد نمونه‌برداری بالایی برای هر sample می‌باشد.
اما برای محاسبه احتمال $P(C|+d)$ چون الگوریتم Likelihood weighting تنها بر انتخاب متغیرهای پائین و downstream اثر می‌گذارد ولی الگوریتم gibbs sampling این مشکل را ندارد، بهتر است از روش gibbs sampling استفاده نماییم.