دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

هوش مصنوعی
بهار ۱۴۰۰
استاد: محمدحسین رهبان
گردآورندگان: کیمیا یزدانی

پاسخ تمرین دوم بخش اول — **جست و جوی محلی** — مهلت ارسال:

سوالات جست‌وجو (یادآوری) (۳۰ نمره)

۱. (۳۰ نمره)

آ) غلط. $h_۲$ لزوما admissible نیست. زیرا فقط یک طرف نامساوی مربوطه دو برابر شده است.

ب) صحیح. در جست و جوی درختی A^* می‌دانیم در صورتی که مسیر بهینه پیدا نشده باشد حتما یک پیشوند از مسیر بهینه در fringe است. بنابراین در زمانی که یک جواب نابهینه در حال pop شدن از fringe است می‌دانیم یک مسیر پیشوندی از مسیر بهینه نیز در fringe قرار دارد.
هزینه‌ی جواب نابهینه‌ی پیدا شده توسط الگوریتم برابر با $f(G') = g(G') + h_۲(G') = g(G')$ است (زیرا $h_۲(G')$ برابر با صفر است) حال اگر مسیر پیشوندی بهینه را a در نظر بگیریم داریم:

$$g(G') \leq f(a) = g(a) + h_۲(a) = g(a) + ۲h_۱(a) \leq ۲(g(a) + h_۱(a)) \leq ۲C^*$$

در نتیجه هزینه‌ی G' از دو برابر جواب بهینه کمتر است.

ج) غلط. $h_۲$ لزوما admissible نیست پس consistent هم نیست و جست و جوی گرافی برای بهینه بودن نیاز به consistency دارد.

سوالات جست‌وجوی محلی (۷۰ نمره)

۱. (۲۲ نمره) TSP یک مسئله‌ی NP-Hard است پس حل آن با A^* با پیچیدگی زمانی نمایی قابل انجام است. اگر در حل مسئله رسیدن به بهینه‌ی سراسری لازم باشد، باید از A^* استفاده شود. اگر تعداد گره‌ها بسیار کم باشد هم استفاده از A^* عملی است.

اگر رسیدن به یک بهینه‌ی محلی کافی باشد این کار در زمان کم با Hill Climbing ممکن است. و هم‌چنین اگر شرایط مسئله تغییر کند – مثلا راهی خراب یا اضافه شود – در Hill Climbing کافی است ادامه‌ی الگوریتم اجرا شود و الگوریتم با سرعت بالا از بهینه‌ی محلی فعلی به یک بهینه‌ی محلی نزدیک جدید جابه‌جا می‌شود، اما در صورت استفاده از A^* باید الگوریتم از ابتدا اجرا شود. جمع‌بندی: در صورت محدود بودن تعداد گره‌ها، ثابت بودن شرایط مسئله و اهمیت بالای رسیدن به بهترین جواب از A^* استفاده شود و در صورت بالا بودن تعداد گره‌ها، قابل تغییر بودن شرایط مسئله و کافی بودن یک جواب خوب از الگوریتم‌های جست‌وجوی محلی مثل Hill Climbing استفاده شود.

۲. (۴۸ نمره)

آ) $k = ۱$: الگوریتم hill climbing ، تنها یک استیت برای بررسی هست در نتیجه بین successor های تولید شده تنها بهترین آن‌ها انتخاب می‌شود. تعریف hill climbing دقیقا همین است.

ب) $k = \infty$:الگلوریتم BFS، ابتدا از تمام nodeها آغاز می‌کنیم و تمام successorها را تولید می‌کنیم و از بین successorهای تولید شده، همه‌ی آن‌ها را انتخاب می‌کنیم. در واقع در این الگوریتم اگر به جای شروع از همه‌ی nodeها از یک node مبدا خاص شروع کرده‌بودیم و یک آرایه‌ی visited نگه می‌داشتیم تا چند بار یک نود را گسترش ندهیم، الگوریتم شبیه به BFS می‌شد.

ج) $T = 0$: الگوریتم first choice hill climbing، اگر به state بهتری از state فعلی بربخوریم که انتخاب می‌شود، در غیر این صورت با صفر بودن دما e به‌توان $-\infty$ می‌رسد و صفر است. پس هیچ‌گاه وضعیت ضعیف‌تری انتخاب نمی‌شود و همیشه اولین وضعیت بهتر که پیدا شد به آن سمت حرکت می‌کنیم.

د) دما در SA برای ایجاد randomness و خروج از بهینه‌ی محلی هست، پس اگر خیلی سریع کاهش پیدا کند و به یک بهینه‌ی محلی برسیم، الگوریتم در همان وضعیت می‌ماند و تنها در ابتدای اجرای الگوریتم random walk در الگوریتم دخیل است.

اگر دما مقدار زیاد و ثابتی داشته‌باشد، الگوریتم بسیار کند عمل می‌کند و converge نمی‌شود. اگر در مقدار کوچکی ثابت باشد، الگوریتم اکثرا شبیه به first choice hill climbing رفتار می‌کند و بعد از رسیدن به بهینه‌ی محلی مدت بسیار زیادی طول می‌کشد تا از آن خارج شود. پس در هر دو صورت کند می‌شود در صورت زیاد بودن بخش convergence کند می‌شود و در صورت کم بودن بخش خروج از بهینه‌های محلی.