Subject:
Year Month Date

زهرا توکلی 401200 63 «سبحه تعالی» تکلیف اول AI

سوال ۱:

. الگوریتم local beam search با k=1 : Steepest ascent/descent Hill Climbing

. الگوریتم local beam search با یک استیت اولیه و نامحدود بودن تعداد استیت های نگهداری شده، تقریبا مثل bruteforce و حسب وجوی حجم بزرگی از search space برای یافتن جواب بهینه است.

. الگوریتم Simulated Annealing با T=0 برای همهٔ زمان ها بدون درنظر گرفتن شرطی برابر توقف حسب وجو: تقریبا مثل First-choice Hill climbing. همسایه های بهتر را می پذیرد (T=0) و به اولین همسایهٔ بهتر می رود.

. الگوریتم Simulated Annealing با T=∞ برای همهٔ زمان ها:

ابتدا یک state اولیه رندوم انتخاب می شود و همسایگی آن ساخته می شود سپس از بین همسایه ها یکی به صورت رندوم انتخاب می شود و اگر بهتر از solution فعلی باشد با احتمال ۱ و اگر نباشد با احتمال بالایی نزدیک ۱ در همهٔ زمان ها انتخاب می شود. پس می توانیم بگوییم از بین همسایه ها کاملا رندوم انتخاب می کند و به آنهایی رود (random search) و شرط پایان T=0 هم می توان برای آن گذاشت و باید شرط پایان برحسب تعداد iteration آن باشد. به تا حدودی به stochastic HC و stochastic local beam search با k=1 شباهت دارد.

. الگوریتم ژنتیک با N=1 که N اندازهٔ جمعیت است:

در این صورت crossover نمی تواند رخ دهد و نسل یک نقره بعدی به صورت تصادفی از طریق mutation ساخته می شود. اگر mutation را علی برای ساخت همسایگی مرد اولیه درنظر بگیریم این الگوریتم مثل انتخاب تصادفی از بین همسایه ها و در نهایت گزارش بهترین مورد دیده شده است. تقریبا random search است و تا حدودی به stochastic HC و stochastic local beam search با k=1 استیت نگهداری شده شباهت دارد.

سوال ۲: الف) تعریف state ها: state ها را می توانیم یک جدول 3x3 از اعداد ۱ تا ۸ و عدد ۰ برای خانهٔ خالی درنظر بگیریم.

تعریف همسایگی: همسایه های یک بورد 3x3 همهٔ بورد هایی هستند که از انتقال یکی از کاشی های اطراف خانه خالی به آن خانه به دست می آیند. بنابراین یک بورد می تواند 2 یا 3 یا 4 همسایه داشته باشد.

تابع هزینه: اگر بخواهیم تابع را minimum کنیم تابع هزینه f را به صورت تعداد خانه های متفاوت از بورد مورد نظر و بورد نهایی تعریف می کنیم. اگر بخواهیم تابع را maximum کنیم می توان f را به صورت تفاضل عدد ثابت با تعداد خانه هایی که با خانه متناظر در بورد نهایی تفاوت دارند درنظر بگیریم و این عدد ثابت می تواند تعداد بیشترین conflict ها که داشت باشد.

Subject:

Year. Month. Date. ( )

ب)

initial state

5	3	
8	7	6
2	4	1

cost=8

goal state

1	2	3
4	5	6
7	8	

neighbors:

5		3
8	7	6
2	4	1

cost=7

5	3	6
8	7	
2	4	1

cost=9

current → neighbor 2

neighbor 1

با توجه به شکل بالا none of above درست است.

سوال 4: الف) تعریف state: هر sdution را که عدد صحیحی از 0 تا 7 است به صورت معادل باینری آن در 3 بیت نمایش می دهیم پس نمایش عدد 2 به صورت 010 می شود.

تعریف همسایگی: همسایه های یک solution را همه solution هایی در نظر می گیریم که از flip کردن یکی از 10 بیت های بدست می آیند.

تعریف تابع هزینه: تابع هزینه برای هر solution را مقدار تابع f(x) به ازای آن در نظر می گیریم.

ب) $x_0 = 2 \Rightarrow x_0$ نمایش: 010 همسایه های x_0: 011، 000، 110

$f(x) = 0.5x^5 - 6x^4 + 2x^3 - 3x^2 + 2x$

$f(3) = -331.5$ ، $f(0) = 0$ ، $f(6) = -3552$ ، $f(2) = -72$

- Stochastic HC: این الگوریتم تمام همسایگی یک عنوان را تشکیل داده و از بین همسایه های بهتر یکی را به تصادف انتخاب کرده و به آن state می رود. احتمال انتخاب هر همسایه بستگی به این دارد که چقدر از حالت کنونی بهتر است. فرق آن با HC عادی این است که الزاماً بهترین همسایه انتخاب می شود.) در اینجا چون تنها همسایه بهتر $x=0$ است به $x=0$ می رود.
- First choice HC: این الگوریتم همسایه ها را یکی یکی می سازد و با مشاهده اولین همسایه بهبود دهنده ساخت همسایگی را متوقف کرده و به آن همسایه می رود. اگر اینجا با flip کردن بیت ها از راست به چپ همسایه ها را بسازیم اولین همسایه بهبود دهنده $x=0$ بوده و به آن می رود.
- Simulated Annealing: این الگوریتم از بین همسایه هایی را به تصادف انتخاب می کند. اگر همسایه بهتری بود قطعاً به آن می رود اگر بدتر بود با احتمال $e^{\frac{\Delta E}{T}}$ به آن می رود. اگر 0 تصادفاً انتخاب شود قطعاً به آن می رویم. اگر 6 انتخاب شود با احتمال $e^{-\frac{3552}{100}} = 3.7 \times 10^{-16}$ به آن می رویم که احتمال خیلی کمی است و احتمالاً الگوریتم سرانجام انتخاب رندوم یک همسایه دیگری رود.

Subject:
Year Month Date ( )

سوال 5 : نمایش افراد جمعیت : اگر n بیت داشته باشیم ، هر فرد جمعیت را به صورت جایگشتی از اندازه‌ٔ بیت ها در نظر می گیریم.

Crossover : چون نمایش افراد به صورت جایگشت است می توانیم crossover را به این صورت تعریف کنیم که کروموزوم های دو والد را از نقطه‌ای تصادفی به دو بخش تقسیم کرده و فرزند اول بخش اول از والد اول را ارث می برد. سپس در طول کروموزوم والد دوم حرکت می کنیم و اگر ژنی از آن در فرزند هست یا به تعداد لازم هست آن ژن را در فرزند قرار می دهیم. اینگونه فرزند حاصل هم یک جایگشت خواهد بود. فرزند دوم هم به همین شکل و با تعویض جای والد اول و دوم می سازیم.

mutation : برای mutation می توانیم 2-swap استفاده کنیم و دو ژن از فرزند را تصادفاً انتخاب و جابه جا کنیم.

تابع fitness : تابع fitness باید به گونه‌ای باشد که آن را ماکزیمم کنیم. در بدترین حالت اگر هر بسته در یک جعبه برود n جعبه نیاز خواهیم داشت. برای هر فرد و نمونه برای محاسبه‌ٔ تعداد جعبه های لازم از چپ به راست حرکت می کنیم و بسته ها را تا وقتی جمع اندازه‌ٔ آن ها کمتر مساوی 1 است در یک جعبه قرار می دهیم و اینگونه تعداد جعبه های لازم برای یک نمونه را محاسبه می کنیم. تابع fitness را به صورت تفاضل n و تعداد جعبه های لازم نمونه در نظر می گیریم و باید آن را maximum کنیم.

برای انتخاب والدی توانیم از روش fitness proportional selection یا روش های دیگر استفاده کنیم.
برای انتخاب بازمانده ها هم می توانیم از fitness based یا age based رفتار کنیم و افراد جدول کرا انتخاب کنیم.

سوال 3 : الف) افراد جمعیت را به صورت یک جایگشت روی اعداد 1 تا 5 در نظر می گیریم.

افراد جمعیت : 15243 1 , 234512 , 341253 , 431524 , 523415

ب) برای mutation می توانیم از عدد دوم تا یکی به آخر جایگشت دو عدد را تصادفاً انتخاب و جابه جا کنیم (2-swaps). یک mutation دیگری می تواند این باشد که از بین عناصر دوم تا یکی به آخر دو تا را تصادفاً انتخاب کرده و دوی را کنار اولی ببریم و عناصر میان آنها را شیفت دهیم: 3 4 1 2 5 3 → 3 4 5 1 2 3

یا می توانیم دو تا را به تصادف انتخاب و ترتیب عناصر میان آنها را هرطور خواستیم به هم بریزیم:
3 4 1 2 5 3 → 3 1 4 5 2 3

ج) تابع fitness را طول تور معرفی شده توسط هر نمونه در نظر می گیریم و باید آن را minimum کنیم.

د) برای crossover می توانیم کروموزوم دو والد را از نقطه‌ای تصادفی برش دهیم. بخش اول فرزند اول مشابه والد اول است. سپس در طول کروموزوم والد دوم حرکت می کنیم و به هر ژنی که می رسیم و در فرزند اول نیست در فرزند اول قرار می دهیم. سپس ژن اول فرزند را به عنوان ژن آخر هم تکرار می کنیم. هم چنین در crossover برای مسئلهٔ TSP می توان از روش partially mapped crossover استفاده کرد:

34125 3 → 43 1254 → pmx
15243 1

روش تقریباً مشابه pmx

PMX

Subject:

Year Month Date ( )

سوال (3): ۵)

افراد: $S_1: 152431$ ، $S_2: 234512$ ، $S_3: 341253$ ، $S_4: 431524$ ، $S_5: 523415$

$f(S_1)=38$ ، $f(S_2)=42$ ، $f(S_3)=38$ ، $f(S_4)=38$ ، $f(S_5)=55$

parent selection (3_way tournament) : روش انتخاب والد

$f(S_1)=f(S_4)$

تورنمنت اول: (S_1, S_2, S_4) → انتخاب می‌شود $S_1: 152431$ ┐ crossover → 154321 → mutation (2-swap) → 134521

تورنمنت دوم: (S_2, S_4, S_5) → انتخاب می‌شود $S_4: 431524$ ┘ → 431524 → mutation → 421534

تورنمنت سوم: (S_1, S_2, S_3) → انتخاب می‌شود $S_3: 341253$ ┐ crossover → 341523 → mutation → 314523

تورنمنت چهارم: (S_1, S_4, S_5) → انتخاب می‌شود $S_4: 431524$ ┘ → 431254 → mutation → 432154

تورنمنت پنجم: (S_1, S_3, S_4) → انتخاب می‌شود $S_1: 152431$ ┐ crossover → 152431 → mutation → 153421

تورنمنت ششم: (S_3, S_4, S_5) → انتخاب می‌شود $S_3: 341253$ ┘ → 341253 → mutation → 342153

tournament selection انواع متفاوتی دارد. مثلا اینکه وقتی یک تورنمنت برگزار کردیم فقط یک نفر برتر را انتخاب کنیم یا چند نفر برتر. اگر بخواهیم یک نفر برتر را انتخاب کنیم به صورت عمل می‌کنیم که ابتدا تعداد بازیکنان لشکر ها (رسابقه (k) مناسب انتخاب می‌کنیم و سپس k نفر را به تعادف برای مسابقه انتخاب کرده و خانه‌ای که بهترین fitness را دارد به عنوان برنده انتخاب می‌کنیم. هرچه k بزرگ‌تر باشد، خانه‌های ضعیف‌تر با fitness کمتر شانس کمتری برای انتخاب شدن دارند. اینجا اگر $k=2$ بود، S_2 هم می‌توانست به عنوان والد انتخاب شود چون از S_5 بهتر است و اگر $k=1$، S_5 هم می‌توانست والد شود. برای survivor selection به روش tournament هم می‌توانیم مانند بالا عمل کنیم و 5 نفر را برای نسل بعدی انتخاب کنیم.

سوال (6): تعریف state: هر فرد و solution را به صورت جایگشتی از اعداد 1 تا n^2 در نظر می‌گیریم که n عدد اول سطر اول جدول، n عدد دوم سطر دوم الی و به همین ترتیب n عدد آخر سطر آخر جدول هستند.

تعریف تابع fitness: تابع fitness را می‌توانیم به صورت مجموع قدر مطلق اختلاف جمع اعداد هر سطر و ستون و قطر با magic constant در نظر بگیریم و زمانی به magic square رسیده‌ایم که مقدار این تابع 0 شود ⟸ minimization

$n \times n$ magic square در آن جمع اعداد هر سطر یا ستون یا قطر $\text{magic constant} = \frac{n(n^2+1)}{2}$

جمع اعداد ماتریس برابر با $\frac{n^2(n^2+1)}{2}$ است وی خواهیم n سطر با مجموع اعداد برابر داشته باشیم پس جمع اعداد هر سطر برابر با $M = \frac{n^2(n^2+1)}{2n} = \frac{n(n^2+1)}{2}$. اگر جمع اعداد سطر ها را با x_i نشان دهیم تابع fitness برابر است با: $f(S) = \sum |x_i - M|$ → fitness function

Crossover: چون به عزم جایگشت است می‌توانیم از PMX یا روش زیر استفاده کنیم: ابتدا والد ها را از نقطه‌ای تصادفی برش می‌دهیم، بخش اول والد اول فرزند اول می‌شود. سپس در طول کروموزوم والد دوم حرکت کرده و با رسیدن به هر ژنی که در فرزند اول نیست آن را در فرزند اول قرار می‌دهیم. فرزند دوم هم همین طور با تعویض نقش والد اول و دوم حاصل می‌شود

mutation: 2_swap