به نام خدا

نام: فاطمه زهرا بخشنده **استاد: دکتر ناصر مزینی**

شماره دانشجویی: 98522157 **درس مبانی هوش محاسباتی**

گزارش تمرین 4:

سوال اول (امتیازی):

مراحل الگوریتم بهینه‌سازی کلونی مورچگان (ACO) به صورت زیر می‌باشد:

توصیف شهودی:

1. در مرحله اول تمامی مورچه‌ها در لانه خود می‌باشند و هیچ فرومونی در محیط وجود ندارد.
2. در مرحله دوم مورچه‌ها شروع به جست‌وجو میان تمامی مسیرها می‌کنند. احتمال جست‌وجوی همه مسیرها با هم برابر است. بدیهی است مسیرهایی که طولانی تر هستند مدت زمان بیشتری هم خواهند داشت.
3. در مرحله سوم مورچه‌هایی که مسیر کوتاه را انتخاب کرده اند سریع‌تر به غذا می‌رسند و رد فرومون آن‌ها در آن مسیر قوی‌تر خواهد بود.
4. در مرحله چهارم مورچه‌های بیشتری از مسیری که فرومون قوی دارد عبور می‌کنند. این امر باعث قوی‌تر شدن فرمومون در آن مسیر می‌شود. همچنین بدلیل پدیده تبخیر فرومون در مسیرهای طولانی کمتر می‌شود. در نتیجه احتمال انتخاب آن مسیر نیز کمتر می‌شود.

توصیف الگوریتمی:

```
Procedure AntColonyOptimization:
    Initialize necessary parameters and pheromone trials;
    while not termination do:
        Generate ant population;
```

Calculate fitness values associated with each ant;

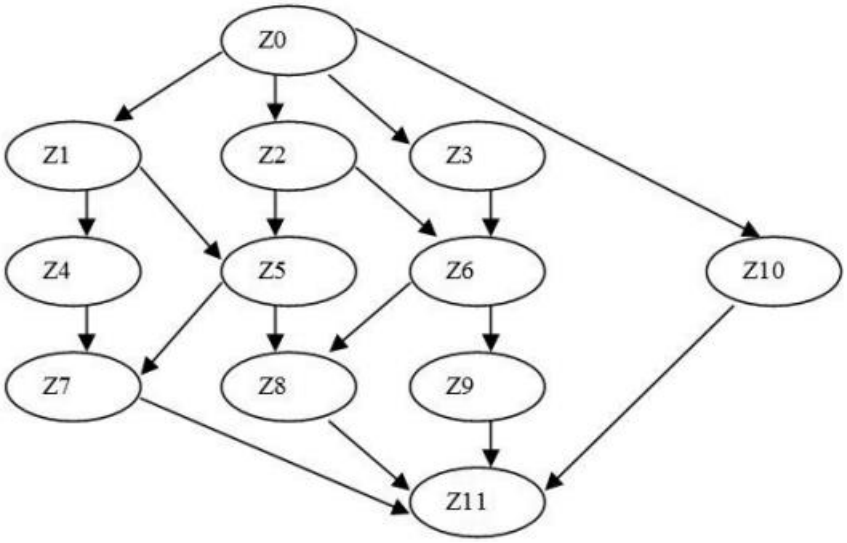
Find best solution through selection methods;

Update pheromone trial;

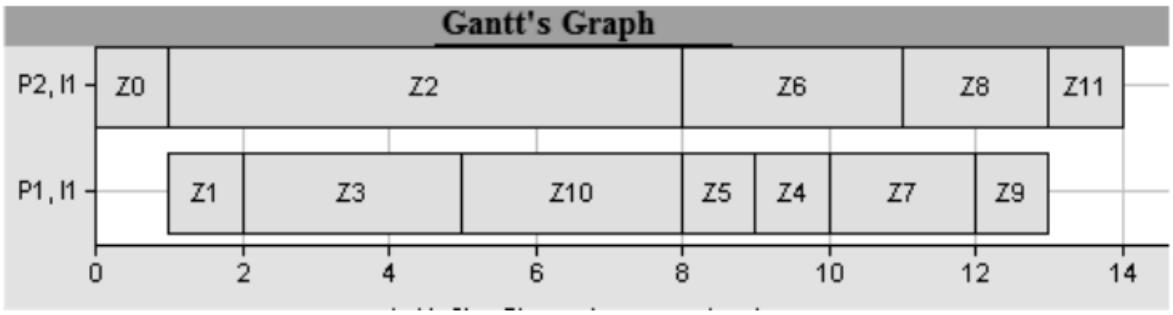
end while

end procedure

حال می‌خواهیم برای تسک زمان‌بندی (Scheduling) در پردازنده‌های چند هسته ای یک الگوریتم ACO ارائه دهیم. در این مسئله سیستم ما دارای دو پردازنده یکسان می‌باشد. همچنین 12 تسک به نام‌های Z0 تا Z11 داریم که زمان اجرای متفاوتی دارند. همچنین یک گراف تقدم تاخر اجرای تسک‌ها نیز مشابه شکل زیر وجود دارد.



اکنون به دنبال یک زمان‌بندی بهینه مشابه شکل زیر هستیم.



راه حل مسئله:

مدل ریاضی مسئله: (S, f, Ω) که S مجموعه راه حل های احتمالی. f تابع هدف و امگا محدودیت های مسئله می باشد (دو پردازه همزمان وارد یک پردازشگر نمی شوند، تقدم تاخر تسک ها)

C = components = {c1, c2, …, cN}

X = states: x = (ci, cj, …, ch) and x is a member of X

هرجا که یال فرومون بیشتر بود احتمال بیشتری دارد تا طی شود.

Tij: بیانگر احتمال پردازه j مستقیما بعد از پردازه i است.

فرض کنیم در هر epoch ما m مورچه در حال ساخت راه حل باشند. پس بعد از هر اپاک مقدار فرومون روی هر یال به صورت زیر تبخیر می شود.

Tij = (1 – P) Tij

با توجه به اینکه در هر اپاک تعداد m مورچه داریم جواب بهینه ممکن است توسط یکی از این مورچه ها تولید شود (فرض کنیم مورچه k). پس رابطه فرومون به صورت زیر تعریف می شود.

$$T_{ij} = T_{ij} + \sum_{k=1}^{m} \Delta T_{ij}^{k}$$

مقدار دلتا برابر است با تغییری که مورچه k در این اپاک بر روی فرومون ایجاد کرده.

هر مورچه در پردازه ام پردازه j را با احتمال Pij(t) انتخاب میکند. t مقدار اپاک است. همچنین N^k مجموعه عملیات ممکن است که محدودیت های مسئله را داراست (امگا). رابطه زیر را داریم:

$$P_{ij}(t) = \frac{T_{ij}(t)^{\alpha} (n_j)^{\beta}}{\sum_{l \in N^{\mu}} T_{il}(t)^{\alpha} (n_l)^{\beta}}$$

مقدار nj یک مقدار heuristic است که بیانگر مقدار کار نسبی باقی مانده روی پردازه j است.

توصیف غیر ریاضی مسئله و مراحل همگرایی:

ابتدا Tرا تشکیل می دهیم که یک ماتریس n*n است و بیانگر فرومون است. n تعداد پردازه های مسئله است. Tij نیز تمایل انتخاب پردازه j بعد از i است. مقادیر اولیه این ماتریس نزدیک به صفر است.

به تعداد ایپاک های مسئله کارهای زیر را انجام می دهیم:

1- یک نسل مورچه ایجاد می کنیم.
2- برای هر مورچه یک ایپاک می زنیم تا زمانی که تمامی پردازه ها وارد گراف شوند.
3- با استفاده از مقادیر T پردازه بعدی را انتخاب می‌کنیم.
4- بر روی یک State ایجاد شده که قرار گرفتیم مقدار فرومون را آپدیت می‌کنیم.
5- با رابطه تبخیر (Evaporation) مقدار T را آپدیت می‌کنیم.
6- عملیات بالا را تا زمانی تکرار می کنیم که به جواب مطلوب برسیم یا تعداد ایپاک ما تمام شود.

سوال دوم:

```
Rock --> Weight --> Value
1 ------> 2 ------> 30
2 ------> 4 ------> 10
3 ------> 1 ------> 20
4 ------> 3 ------> 50
5 ------> 5 ------> 70
6 ------> 1 ------> 15
7 ------> 7 ------> 40
8 ------> 4 ------> 25
```

ابتدا به هر یک از این سنگ ها به ترتیب اعداد 1 تا 8 را نسبت می دهیم و درون کد با این اعداد کار می‌کنیم. threshold و آستانه ی ظرفیت را هم برابر با 25 در نظر می گیریم.

حال 8 تا کروموزوم را در جمعیت در نظر میگیریم که در ابتدا به صورت رندوم آنها را مقداردهی می کنیم. به این صورت که هر کروموزوم شامل 8 تا ژن (gen) می باشد، یعنی یک آرایه ی 8 تایی فرض شده است که هر ژن 0 یا 1 می باشد که نشان دهنده انتخاب شدن یا نشدن آن سنگ متناظر می باشد، که یعنی به عنوان مثال اگر ژن درون خانه دوم کروموزومی 1 باشد، آنگاه آن سنگ دوم که سنگ نقره باشد، انتخاب میشود و دزد این سنگ نقره را درون کوله پشتی خود می گذارد. در غیر اینصورت اگر به عنوان مثال ژن درون خانه چهارم کروموزومی 0 باشد، آنگاه سنگ چهارم که سنگ الماس می باشد انتخاب نمی شود و دزد این سنگ الماس را درون کوله پشتی خود نمی گذارد. به این صورت 50 نسل جلو میرویم و در نهایت جواب را می یابیم.

```
Size Of Population is: (8, 8)
Initial Population is:
[[0 0 0 0 0 0 0 0]
 [1 0 0 1 0 0 0 0]
 [0 1 1 0 1 0 0 0]
 [1 0 0 1 1 0 0 1]
 [1 1 1 1 0 0 0 0]
 [0 0 1 0 1 1 0 1]
 [0 0 1 1 0 0 1 1]
 [0 0 1 0 1 0 0 0]]
```

میزان Fitness هر کروموزوم را براساس حاصل جمع حاصل ضرب های هر ژن (gen) در ارزش آن سنگ بدست می آوریم. البته حاصل جمع حاصل ضرب های هر ژن (gen) در وزن آن سنگ باید از threshold و آستانه ی ظرفیت یعنی 25 کیلوگرم کوچک تر باشد، چون در غیر اینصورت نمی توان در کوله پشتی سنگی را قرار داد و کوله پشتی کاملا پرشده است و ظرفیت آن تکمیل شده است، بنابراین Fitness آن کروموزوم برابر با صفر می شود.

در مرحله بعد مناسبترین و Fit ترین individual ها را انتخاب می کنیم و بعد crossover از نوع one-point crossover را روی این individual ها اعمال می کنیم.

در مرحله ی بعد روی این individual ها جهش اعمال می کنیم. به این صورت که یک ژن را به صورت رندوم و تصادفی در هر کروموزوم انتخاب کرده و مقدار آن را عوض می کنیم یعنی اگر 0 بود به 1 و اگر 1 بود به 0 تغییر می دهیم.

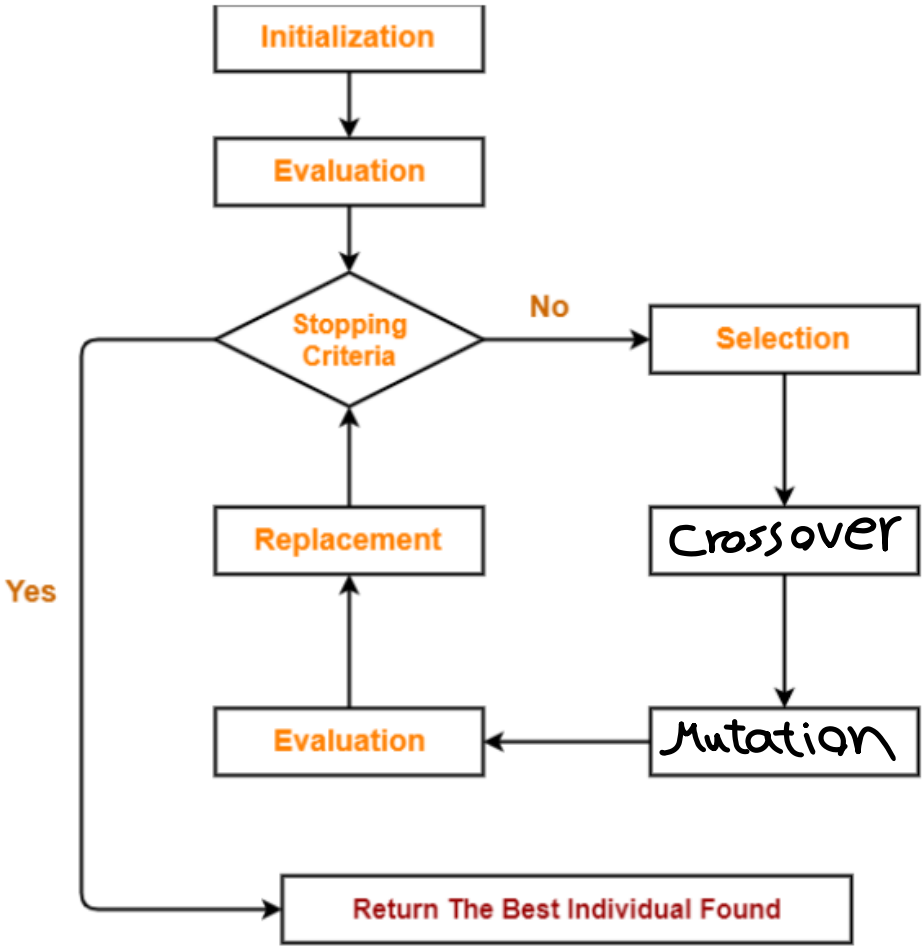
حال همه ی کارهای فوق 50 بار تکرار می کنیم. یعنی در واقع 50 نسل و generation جلو می رویم و در نهایت به جواب که یعنی این که کدام سنگها باید انتخاب شوند می رسیم.

در آخر، سنگهای زمرد، یاقوت، الماس، برلیان، فیروزه، عتیق و کهربا انتخاب میشوند و کروموزوم نهایی به صورت [1, 0, 1, 1, 1, 1, 1, 1] می باشد.

```
Chromosomes of Last generation are:
[[1 0 1 1 1 1 1 1]
 [1 0 1 1 1 1 1 1]
 [1 0 1 1 1 1 1 1]
 [1 0 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1]
 [1 0 1 1 1 1 1 1]
 [1 0 1 1 1 1 0 1]
 [1 0 1 0 1 1 1 1]]

Result Choromosome is: [array([1, 0, 1, 1, 1, 1, 1, 1])]
Selected Rocks are:
1: زمرد
3: یاقوت
4: الماس
5: برلیان
6: فیروزه
7: عتیق
8: کهربا
```

سوال سوم:



مسئله به صورت زیر است:

Maximize $f(x) = x^2$,

over {0, 1, 2, ... , 50}

Initial x values of (13, 24, 8, 19)

یک دور الگوریتم ژنتیک را اجرا می کنیم.

Initialization & Evaluation:

We create initial random population, and evaluate fitness for each one.

String number	x	Initial population (randomly generated)	Fitness f_i $f(x) = x^2$	p select i $\frac{f_i}{\sum f}$	Expected count $\frac{f_i}{\bar{f}}$
1	13	01101	169	0.14	0.56
2	24	11000	576	0.49	1.97
3	8	01000	64	0.06	0.24
4	19	10011	361	0.31	1.24
Sum			1170	1	4
Average			293	0.25	1
Max			576	0.49	1.97

Stopping criteria:

- Number of generations
- Population member(s) with > specified fitness
- No change in max fitness in *n* generations

None has happened. (We should specify values for "Number of generations" and our goal "fitness")

Selection:

We select string number 2 that has higher expected count. And the least expected count was for string number 3. So we eliminate it and replace it with number 2.

Crossover:

We start with relatively high crossover rate = 100%

If we continued, We would reduce it during the run.

We select two parents, from population, for crossing.

String number	Mating pool after reproduction	Mate	Crossover site (randomly selected)	New population	x	Fitness f_i $f(x) = x^2$
1	01101	1	4	01100	12	144
2	11000	2	4	11001	25	625
2	11000	4	2	11011	27	729
4	10011	2	2	10000	16	256
Sum						1754
Average						439
Max						729

Now we have total fitness of 1754 that is greater than 1170 we had before.

Mutation:

The bits are changed from 0 to 1 and from 1 to 0, from randomly chosen position of randomly selected strings.

String number	after crossover	Apply Mutation (randomly selected)	x	Fitness f_i $f(x) = x^2$
1	01100	11100	28	784
2	11001	11001	25	625
2	11011	11011	27	729
4	10000	10100	20	400
Sum				2538
Average				635
Max				784

Now we have total fitness of 2538 that is greater than 1754 we had before.

Replacement:

We got better fitness, so we replace new population with previous population.

We check stopping criteria:

- Number of generations
- Population member(s) with > specified fitness
- No change in max fitness in *n* generations

Number 3 haven't happen, yet. But number 1 and number 2 depend on what we consider for "Number of generations" and "specified fitness".

Anyway, we stop here. Because we were asked to run one round of the generic algorithm.

منابع: مثال اسلاید و لینک و لینک