پروژه نهایی درس هم طراحی سخت افزار و نرم افزار

حل مسئله فروشنده دوره گرد (TSP) با استفاده از الگوریتم کلونی مورچگان (ACO)

صدف عابدی - ۹۸۲۳۴۵۳ هلیا قربانی - ۹۸۲۴۳۵۳

چکیده:

در این پروژه هدف ما پیاده سازی الگوریتم کلونی مورچگان به زبان SystemC و توزیع آن در قالب دو ماژول سخت افزار و نرم افزار است.

در ابتدا به شرح مختصری از این الگوریتم می پردازیم؛ الگوریتم کلونی مورچگان یا در حقیقت «بهینه‌سازی کلونی مورچگان» (Ant Colony Optimization) همانطور که از نام آن مشخص است، بر پایه رفتار طبیعی کلونی‌های مورچگان و مورچگان کارگر شاغل در آن‌ها بنا نهاده شده است. فرآیند

یافتن منابع غذایی در کلونی مورچگان بسیار بهینه است. زمانی که مورچه‌ها عملیات کاوش برای یافتن منابع غذایی را آغاز می‌کنند، به طور طبیعی یک مسیر «منطقی» و «بهینه» از آشیانه خود به منابع غذایی پیدا می‌کنند. به عبارت دیگر، جمعیت مورچگان به نحوی همیشه قادر هستند تا یک مسیر بهینه را برای تامین منابع غذایی مورد نیاز بیابند. شبیه‌سازی چنین رفتار بهینه‌ای، پایه و اساس بهینه سازی کلونی مورچگان را تشکیل می‌دهد.
مورچه‌ها در ضمن حرکت خود به سمت منبع غذایی، ردی از «فرومون» (Pheromone) در محیط منتشر می‌کنند که به‌طور طبیعی و با گذر زمان متلاشی می‌شود. مورچه‌ای که (به‌طور تصادفی) کوتاهترین مسیر به سمت منبع غذایی را انتخاب کرده، سفر برگشتی به سمت آشیانه را زودتر از دیگر مورچه‌ها آغاز می‌کند. در چنین حالتی، این مورچه در مسیر بازگشت به آشیانه، دوباره شروع به منتشر کردن فرومون در محیط می‌کند و از این طریق، رد فرومون به جا گذاشته در کوتاهترین مسیر را تقویت می‌کند.

مورچه‌های دیگر، به‌طور غریزی، قوی‌ترین مسیر فرومون موجود در محیط را دنبال و رد فرومون در این مسیر را تقویت می‌کنند. پس از گذشت مدت زمان مشخصی، نه تنها رد فرومون موجود در کوتاه‌ترین مسیر از بین نمی رود، بلکه، با انباشته شدن رد فرومون دیگر مورچه‌ها، بیش از پیش تقویت می‌شود. مسیری که قوی‌ترین رد فرومون در آن به جا گذاشته شده باشد، به مسیر پیش فرض برای حرکت مورچه‌ها از کلونی به منبع غذایی و برعکس تبدیل می‌شود.

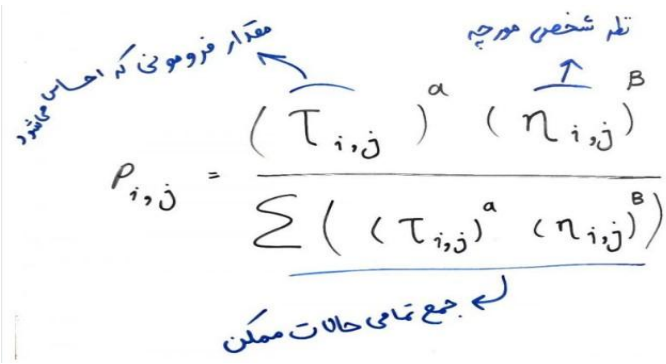
یکی از محبوب‌ترین روش‌ها برای نمایش چگونگی عملکرد روش فرا اکتشافی الگوریتم کلونی مورچگان، استفاده از آن در حل مسأله فروشنده دوره‌گرد است.

هر کدام از مورچه‌ها، از یک شهر (یک رأس در گراف) کاملا تصادفی شروع می‌کنند. سپس، در هر گام از فرآیند تولید جواب، در راستای یال‌های گراف به حرکت می‌پردازند. هر مورچه، مسیر پیموده شده در گراف را به خاطر می‌سپارد و در گام‌های بعدی، یال‌هایی را برای حرکت در گراف انتخاب می‌کند که به مکان‌های (رأس‌های) از پیش پیموده شده منتهی نشوند. به محض اینکه تمامی رأس‌های گراف توسط یک مورچه پیمایش شد، یک جواب کاندید تولید می‌شود.

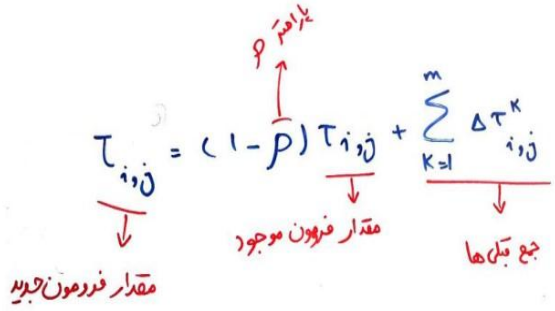
در هر گام از فرآیند تولید جواب، مورچه‌ها به طور احتمالی، از میان یال‌های در دسترس (یال‌های پیموده نشده و منتهی به رأس‌هایی که از آن‌ها گذر نکرده)، یک یال را برای پیمایش انتخاب می‌کنند. نحوه محاسبه احتمال انتخاب یال‌ها، به پیاده‌سازی انجام شده از الگوریتم کلونی مورچگان بستگی دارد. پس از اینکه تمامی مورچه‌ها یک جواب کاندید تولید کردند، فرومون روی یال‌ها، براساس «قانون به روز رسانی فرومون» به روز رسانی می‌شود.

شبه کد الگوریتم کلونی مورچگان در ادامه آمده است.

- پارامترهای الگوریتم کلونی مورچگان تنظیم شده و ردهای فرومون مقداردهی اولیه می‌شوند.
- تا زمانی که شرط توقف ارضا نشده باشد:
- **مرحله اول یا مرحله «تولید جواب‌های کاندید» (Construct Ant Solution) را شروع کن.**



احتمال انتخاب توسط فرمول بالا محاسبه می‌شود؛نحوه‌ی محاسبه به این صورت است که تصمیمِ یک مورچه در انتخاب یک مسیر به دو چیز بستگی دارد. اول فکری که خودش می‌کند یعنی نظر شخصیِ خود مورچه، دوم مقدار فرومونی که بر روی زمین حس می‌کند، یعنی پیام‌هایی که مورچه‌های دیگر برای او گذاشته‌اند. در فرمول بالا ۲ پارامتر **آلفا** و **بتا** (توان‌ها) وجود دارد که کاربر به الگوریتم به عنوان مقدار اولیه می‌دهد. اگر پارامتر آلفا بیشتر از بتا باشد، یک مورچه بیشتر به مقدار فرومون‌ها یعنی پیام‌های بقیه‌ی مورچه‌ها توجه می‌کند تا نظرِ شخصیِ خودش! و اگر مقدار بتا بیشتر از آلفا باشد، نظرِ شخصیِ یک مورچه بیشتر از مقدار فرومون‌ها (تجربه‌ی مورچه‌های قبلی) در انتخاب مسیر تاثیر می‌گذارد. توجه داشته‌باشید که با هر بار که الگوریتم جلو می‌رود، مقداری از فرومون‌ها تبخیر می‌شود.



- **مرحله دوم یا مرحله «جستجوی محلی جواب‌ها» (Local Search) را شروع کن.**

در این مرحله، از جواب‌های بهینه محلی استفاده می‌شود تا مشخص شود کدام یک از فرومون‌ها باید به روز رسانی شوند. این مرحله اختیاری است و در برخی از پیاده‌سازی‌های انجام شده از الگوریتم کلونی مورچگان وجود ندارد.

- **مرحله سوم یا مرحله «به روز رسانی فرومون» (Pheromone Update) را انجام بده.**

1. کاهش مقادیر فرومون متناظر با تمامی جواب‌های کاندید از طریق فرآیند «تبخیر فرومون»
2. افزایش مقادیر فرومون متناظر با جواب‌های کاندید عضو مجموعه «جواب‌های خوب»
3. در صورتی که شرط توقف ارضا شده باشد، اجرای الگوریتم را متوقف کن؛ در غیر این صورت، مراحل را مجددا انجام بده.

شرح کد SystemC :

کلاس ACO را جهت تعریف پارامتر ها و توابع مورد نیاز برای پیاده سازی الگوریتم در فایل ACO.h تمامی توابع را تعریف شده و سپس توابع و وظایف را بر اساس محاسبات بیشتر و امکان موازی سازی به سخت افزار و محاسبات سبک تر در حد چک کردن بر اساس نتایج حاصل از محاسبات و پرینت مقادیر به نرم افزار داده شده است.

Hardware.h :

ابتدا ثابت ها و مقادیر مورد نیاز برای محاسبات احتمالات در توابع را تعریف کردیم.

Cities : شهر هایی که برای طی کردن مسیر ها استفاده می شوند .

Q : ضریب فرمون ریخته شده توسط مورچه در مسیر در فرمول محاسبه احتمال آن

NumberOfCities : تعداد شهر ها

NumberOfAnts : تعداد مورچه ها

pheromones : مقدار فرومون

Cityi, Cityj : ایندکس شهر ها

Distance : فاصله بین شهر ها

Routes : مسیر بین شهرها

calc_distance() : فاصله دو شهر را بر اساس فرمول فاصله محاسبه می کند

calc_phi() : احتمال انتخاب مسیر را بر اساس فرمولی که بالاتر ارائه شد،محاسبه می کند

calc_length() : مسیر ها را پیدا می کند،با هم جمع می کند و طول نهایی را بر می گرداند

update_pheromones() :

مقدار فرومون را از شهری به شهر دیگر آپدیت می کند که با استفاده از فرمول نوشته شده،در صورتی که مورچه از مسیری عبور کند،با استفاده از تابع محاسبه فاصله،در طول آن مسیر فرومون را زیاد می کند.

از ماژول hardware یک شیء می سازیم.

در کلاس ACO توابع را تعریف می کنیم و محاسبات مربوط به فرمول ها را با استفاده از hardware انجام می دهیم. Event های مورد نیاز برای توابع تعریف شده اند و محاسبات در صورت امکان با wait و notify به صورت همزمان انجام می شوند.

علاوه بر تعریف مقادیری که در بالا معرفی شد، متغیر Ro نیز برای مقدار تبخیر فرومون با گذشت زمان نیز تعریف می کنیم.

initialCity نیز شهر مبدا که مورچه از آن شروع می کند را مشخص می کند

از تابع init برای تشکیل گراف مسیر های بین شهر ها استفاده می کنیم که با استفاده از 0 و 1 عدم وجود یا وجود مسیر بین شهر ها را مشخص می کند.

این تابع همچنین مقدار فرومون در هر شهر، احتمال شهری مهمورچه می رود،بهترین مسیر(routes) و بهترین طول را مشخص می کند.

در تابع **init** آرایه های مقادیر را ساخته و فضایی برای آن ها در نظر میگیریم.

در تابع مجازی **ACO~** فضاهای داده شده را حذف می کنیم.

probs : احتمال شهری که مورچه می رود،مشخص می کند

ACO::length(int antk) : طول مسیر مورچه را محاسبه می کند

تابع **route** مسیر مورچه را بر اساس اینکه به کدام شهر برود و آیا در به آن شهر مسیری وجود دارد یا نه و اینکه آیا قبلا از آن عبور کرده است یا نه، تعیین می کند.

تابع **optimize** حرکت مورچه در شهر های مختلف را نمایش می دهد،مسیر را پرینت می کند،پایان مسیر را اعلام میکند،فرمون را آپدیت می کند و با گذر زمان کم می کند و ماتریس آن را پرینت کرده و همچنین طول بهترین مسیر را حساب می کند و در route قرار می دهد و در نهایت مسیر را پاک می کند.

Software.h :

در این قسمت متغیر ها را مقدار دهی می کنیم.

برای تمامی توابع یک event تعریف می کنیم و در sc_ctor به آن ها sensitive می کنیم.

یک عدد رندوم بین 0 و 1 تولید می کنیم.

از آنجایی که نرم افزار نسبت به سخت افزار کند تر است یک delay برای آن در نظر می گیریم.

connect_cities(int cityi, int cityj) :

اگر بین دو شهر مسیری موجود باشد،گراف شهر ها را اجرا می کند و مقداری رندوم بین 0 و 1 تولید کرده و به فرومون می دهد.

تابع**set_city_pos(int city, double x, double y)** مکان شهر ها را مشخص می کند.

تابع **city_selector()** بر اساس فرمول احتمال هر شهر را حساب می کند و یک مقدار رندوم تولید می کند.اگر احتمال از مقدار رندوم بیشتر باشد،و مسیر وجود داشته باشد،آن شهر را

انتخاب می کند.اگر شرط برقرار باشد،احتمال phi را حساب می کند و شهر بعدی را در route قرار می دهد.

تابع **route_maker(int antk)** شهر مبدا را می گیرد و بر اساس آن روی شهر هایی که بینشان مسیر وجود دارد، جلو می رود.

validator(int antk, int iteration) : چک می کند که مسیر بین شهر ها وجود دارد یا خیر

از ماژول نرم افزار نیز شیء می سازیم.

توابع exist , visited به ترتیب وجود مسیر و عبور کردن از یک شهر را نشان می دهند.(از هر شهر یکبار باید عبور کرد)

تابع**() ACO::printPHEROMONES** مقادیر فرومون را پرینت می کند.

تابع **() ACO::printGRAPH** گراف مسیر ها را رسم می کند.

تابع **() ACO::printRESULTS** نتیجه نهایی را پرینت می کند؛بهترین مسیر،بهترین طول

:Main

از ACO شیء می سازیم و مقداردهی اولیه توسط init انجام می شود.

در نهایت اگر error نداشته باشیم،تابع sc_start را صدا می زنیم.

نمونه ای از خروجی کد به شکل زیر است:

```
Microsoft Visual Studio Debug Console

        SystemC 2.3.3-Accellera --- Apr 15 2022 01:26:47
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
 GRAPH:
   | 0 1 2 3 4 5 6 7
 - | - - - - - - - -
 0 | x 1 1 1 0 0 0 1
 1 | 1 x 0 1 0 1 0 1
 2 | 1 0 x 0 1 1 1 0
 3 | 1 1 0 x 1 0 0 0
 4 | 0 0 1 1 x 1 0 1
 5 | 0 1 1 0 1 x 0 0
 6 | 0 0 1 0 0 0 x 1
 7 | 1 1 0 0 1 0 1 x

Number of connections: 28

 @ 500 ms pheromones updated...
 PHEROMONES:
   |     0       1       2       3       4       5       6       7
 - | ---------------------------------------------------------------
 0 |     x     1.624   0.571   1.641     0       0       0     0.188
 1 |   1.624     x       0     1.078     0     1.472     0     0.844
 2 |   0.571     0       x       0     0.107   0.807   1.987     0
 3 |   1.641   1.078     0       x     0.012     0       0       0
 4 |     0       0     0.107   0.012     x     0.376     0     1.995
 5 |     0     1.472   0.807     0     0.376     x       0       0
 6 |     0       0     1.987     0       0       0       x     0.065
 7 |   0.188   0.844     0       0     1.995     0     0.065     x

@ 500 ITERATION 1 HAS STARTED!
```

ماتریس GRATH وجود مسیر بین شهر ها را نمایش می دهد و PHEROMONES یک مقدار اولیه فرومون به صورت رندوم نمایش می دهد.

```
Microsoft Visual Studio Debug Console

@ 500 ITERATION 1 HAS STARTED!

@ 510 : ant 0 has been released!
@ 520  :: releasing ant 0 again!
0 3 1 5 4 7 6 2
@ 530  :: route done
@ 530 ms  : ant 0 has ended!
@ 530 : ant 1 has been released!
@ 540  :: releasing ant 1 again!
0 1 3 4 5 2 6 7
@ 550  :: route done
@ 550 ms  : ant 1 has ended!
@ 550 : ant 2 has been released!
@ 560  :: releasing ant 2 again!
0 3 1 5 4 7 6 2
@ 570  :: route done
@ 570 ms  : ant 2 has ended!
@ 570 : ant 3 has been released!
@ 580  :: releasing ant 3 again!
0 1 5 2 6 7 4 3
@ 590  :: route done
@ 590 ms  : ant 3 has ended!
@ 590 : ant 4 has been released!
@ 600  :: releasing ant 4 again!
@ 610  :: releasing ant 4 again!
@ 620  :: releasing ant 4 again!
0 1 5 2 6 7 4 3
@ 630  :: route done
@ 630 ms  : ant 4 has ended!
@ 630 : ant 5 has been released!
@ 640  :: releasing ant 5 again!
0 1 5 2 6 7 4 3
```

۸ مورچه و ۸ شهر داریم که این مورچه ها طبق ITERATION تعریف شده ۵ با به صورت رندوم از یک شهر شروع به حرکت می کنند.

```
Microsoft Visual Studio Debug Console

updating PHEROMONES . . . done!

 @ 1220 ms pheromones updated...
 PHEROMONES:
  |     0       1       2       3       4       5       6       7
- | ------------------------------------------------------------
0 |     x     5.882   0.457   2.751     0       0       0     0.151
1 |   5.882     x       0     3.116     0     6.383     0     0.676
2 |   0.457     0       x       0     0.086   5.229   7.610     0
3 |   2.751   3.116     0       x     4.593     0       0       0
4 |     0       0     0.086   4.593     x     2.554     0     6.802
5 |     0     6.383   5.229     0     2.554     x       0       0
6 |     0       0     7.610     0       0       0       x     6.073
7 |   0.151   0.676     0       0     6.802     0     6.073     x


@ 1270 ms ITERATION 1 HAS ENDED!

@ 1270 ITERATION 2 HAS STARTED!

@ 1280 : ant 0 has been released!
@ 1290  :: releasing ant 0 again!
@ 1300  :: releasing ant 0 again!
@ 1310  :: releasing ant 0 again!
@ 1320  :: releasing ant 0 again!
@ 1330  :: releasing ant 0 again!
0 1 5 2 6 7 4 3
@ 1340  :: route done
@ 1340 ms  : ant 0 has ended!
@ 1340 : ant 1 has been released!
@ 1350  :: releasing ant 1 again!
0 1 5 2 6 7 4 3
```

پس از هر ITERATION ماتریس فرمون update و چاپ می شود.

```
Microsoft Visual Studio Debug Console
@ 3900  :: releasing ant 7 again!
@ 3910  :: releasing ant 7 again!
0 1 3 4 5 2 6 7
@ 3920  :: route done
@ 3920 ms  : ant 7 has ended!

updating PHEROMONES . . . done!

 @ 4420 ms pheromones updated...
 PHEROMONES:
  |     0       1       2       3       4       5       6       7
- | ---------------------------------------------------------------
0 |     x    17.564   0.678   3.786     0       0       0     0.062
1 |  17.564     x       0     9.146     0    11.425     0     5.051
2 |   0.678     0       x       0     0.035  17.296  21.421     0
3 |   3.786   9.146     0       x    19.357     0       0       0
4 |     0       0     0.035  19.357     x    13.690     0     9.275
5 |     0    11.425  17.296     0    13.690     x       0       0
6 |     0       0    21.421     0       0       0       x    20.792
7 |   0.062   5.051     0       0     9.275     0    20.792     x


@ 4470 ms ITERATION 5 HAS ENDED!

@ 4520 BEST ROUTE:
0 1 3 4 5 2 6 7
length: 132.75

G:\4002\hs\HW1\HWSW\Debug\HWSW.exe (process 19288) exited with code 0.
Press any key to close this window . . .
```

پس از ۵ بار تکرار بهترین مسیر که بر اساس الگوریتم محاسبه شده که نشان می دهد که از چه شهر هایی باید عبور کنیم و طول مسیر نمایش داده می شود.