پروژه درس همطراحی سخت افزار و نرم افزار

استاد درس: دکتر رودکی

دانشجویان: سید علی تولیت(۹۸۲۰۷۱۳) – علی توکلی(۹۸۲۰۶۸۳)

موضوع: طراحی توأم سخت افزاری و نرم افزاری الگوریتم بهینه سازی کلونی مورچگان (Ant Colony)

پارتیشن بندی(Hardware software partitioning):

برای این پروژه از رویکرد dual استفاده شده است، به این صورت که ابتدا تمامی تسک ها به صورت نرم افزاری توسط یک Cpu پیاده سازی شده است و سپس برای بهینه سازی الگوریتم به کار گرفته شده، پیاده سازی گراف شهر ها و فاصله میان آن ها به صورت ASIC مجددا پیاده سازی شده است.

دلیل این امر پیچیدگی زمانی بالای (o(n^2)) محاسبه فاصله ها میان شهرها و مجموع طول مسیر طی شده توسط مورچه، محاسبه ی visibility است.

الگوریتم:

مسئله فروشنده دوره گرد به این صورت است که فروشنده دوره گرد باید از یک مکان شروع کند، در هر شهر فقط یک بار توقف کرده و در نهایت پس از بازدید از همه شهر ها به شهر اولیه بر گردد.

این مسئله تماما راجع به پیدا کردن کمترین مسیر با شرایط مطرح شده است.

برای حل این مسئله با استفاده از الگوریتم بهینه سازی ACO به صورت زیر عمل می کنیم:

گر n شهر وجود داشته باشد، مورچه ها به صورت رندوم از یک شهر شروع به حرکت می کنند.

نکته قابل توجه این است که این مورچه ها دو مزیت نسبت به مورچه های واقعی دارند:

۱. حافظه دارند (دوباره از شهرهایی که رفتند، دوباره بازدید نمی کنند. توسط فیلد `set<int, greater<int>> available;` در کلاس Ant پیاده سازی شده است.)
۲. فاصله بین شهرها را می دانند و شهر نزدیک تر را انتخاب می کنند.(اگر تمامی شرایط برابر بود. توسط متغیر `vector<vector<double>> cost;` در ماژول hardware پیاده سازی شده است.)
۳. اگر فاصله بین دو شهر یکسان بود، شهری با pheromone بیشتر را انتخاب می کند.

- P زیر احتمال حرکت مورچه k از شهر i به j می باشد.(که در تابع `double moveProbability(int i, int j, double norm)` پیاده سازی شده است.)

$$p_{ij}^{k} = \begin{cases} \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{s \in allowed_k} [\tau_{is}]^{\alpha} \cdot [\eta_{is}]^{\beta}} & j \in allowed_k \\ 0 & \text{otherwise} \end{cases}$$

- τ_{ij} شدت فرومون بین شهر i و j است.(پیاده سازی شده توسط T)
- آلفا میزان تاثیر τ_{ij} است.(alpha)
- μ در اصل visibility شهر j از شهر j است. که برابر $\frac{1}{distance(i,j)}$ است.
- بتا میزان تاثیر visibility است.(beta)

بعد از این که هر مورچه n بار الگوریتم بالا را اجرا کند، یک دور کامل طی شده است. فرومون ها به طوری آپدیت می شوند که مسیر کوتاه تر فرومون بیشتری از مسیر بلندتر دارد، فرمول آپدیت شدن فرومون به صورت زیر است:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}$$

$$\Delta\tau_{ij} = \sum_{k=1}^{l} \Delta\tau_{ij}^{k}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{if ant } k \text{ travels on edge } (i,j) \\ 0 & \text{otherwise} \end{cases}$$

- ρ مساوی است با نرخ تبخیر فرومون که در کلاس Ant با متغیر `double` `evaporation` نمایش داده می شود.
- Q مقداری ثابت است که در تابع `void pheroRemain()` و در متغیر Q قرار دارد.
- L_k برابر طول مسیری است که مورچه طی کرده است و در تابع `void` `pheroRemain()` با متغیر `tourDist` تعریف شده است.

- $\Delta\tau_{ij}^{k}$ میزان افزایش فرومون روی مسیر بین شهر i و j توسط مورچه k است.
- $\Delta\tau_{ij}$ میزان افزایش فرومون روی مسیر بین شهر i و j توسط تمام مورچه ها است.

بر اساس این فرمول ها و الگوریتم توضیح داده شده و با توجه به تاثیر فرومون مسیر خود را مورچه انتخاب کرده و دور کمینه بعد چندین مرتبه طی شدن مسیر توسط مورچه های متعدد به دست می آید.

توضیحات جزئی تر در کامنت های پروژه موجود می باشد.