Programming Assignment 1

Student ID: 0660004

Name: 王順興

1. The idea of your program

**Serial version:**

num\_city: number of cities to connect

2opt\_swap(start, end);

for (i = 1; i < num\_city - 1; ++i)

for (j = i + 1; j < num\_city; ++j)

2opt\_swap(i, j);

**Can be changed to:**

start from (1) to (num\_city - 1)

2opt\_swap with \*depth\* (2) to (num\_city - i)

for (i = 1; i < num\_city - 1; ++i)

for (depth = 1; depth < num\_city - i; ++depth)

2opt\_swap(i, i + depth);

*depth* is the length between the 2opt\_swap indices

**Idea: Split *depth* across threads**

maximum\_depth = num\_city - 1

depth\_for\_each\_thread = maximum\_depth / num\_thread

for (depth = thread\_depth\_start[n]; depth < thread\_depth\_end[n]; ++depth)

for (i = 1; i < num\_city - depth; ++i)

2opt\_swap(i, i + depth);

2opt\_swap(start, end):

\_read\_lock

create new\_route

\_unlock

assert(distance(new\_route) < distance(current\_route))

\_write\_lock

current\_route = new\_route

\_unlcok

end

**Note:**

* A race condition may occur when a thread pass through assertion, but not yet change the current\_route; a better current\_route may be overwritten, so the assertion have to be done again after applying \_write\_lock
* The aforementioned race condition also prevents partial update of the current\_route, so a new array(new\_route) is created every operation.
* distance(current\_route) can be cached.
* Distance calculation can be optimized by calculate only the update segment of the route.
* There are also two ways to split the depth

Example 9 depth, 3 threads

Chunk:

Thread 1: 1/2/3 depth

Thread 2: 4/5/6 depth

Thread 3: 7/8/9 depth

Balanced:

Thread 1: 1/4/7 depth

Thread 2: 2/5/8 depth

Thread 3: 3/6/9 depth

1. Performance (run-time) analysis with 1, 2, 4, 8, 16 core(s)

|  |  |  |  |
| --- | --- | --- | --- |
| # of 2opt\_swap called in 10 mins | | | |
|  | before distance calculation is optimized | | Dist. Calc. optimized |
| Thread | 2opt\_call - **Chunk** | 2opt\_call - **Balanced** | 2opt\_call – **Balanced.opt** |
| 1 | 29511 | 22687 | 58793 |
| 2 | 55504 | 40865 | 118287 |
| 4 | 77554 | 78964 | 241688 |
| 8 | 137454 | 156811 | 516440 |
| 16 | 187609 | 263862 | 1027096 |

1. Record your distance every 30 seconds with **test11**

|  |  |  |  |
| --- | --- | --- | --- |
|  | before distance calculation is optimized | | optimized |
| Time | Distance - **Chunk** | Distance - **Balanced** | Distance – **Balanced.opt** |
| 0m 30s | 5236100228259.406250 | 5236717679532.094727 | 5234267109925.143555 |
| 1m 00s | 5231971534271.667969 | 5233515634808.969727 | 5228939589511.834961 |
| 1m 30s | 5227836991118.258789 | 5230367659596.432617 | 5223222758020.639648 |
| 2m 00s | 5223718046276.681641 | 5227120073294.151367 | 5217302527816.525391 |
| 2m 30s | 5219646075512.209961 | 5223826016575.713867 | 5211267723055.649414 |
| 3m 00s | 5215371936230.681641 | 5220503616187.361328 | 5205388533386.734375 |
| 3m 30s | 5211304150395.791016 | 5217188075359.485352 | 5199606625218.836914 |
| 4m 00s | 5207007363954.340820 | 5213956574863.473633 | 5193726923715.416992 |
| 4m 30s | 5202663341521.856445 | 5210696144937.516602 | 5187865127156.357422 |
| 5m 00s | 5198240863981.068359 | 5207478719493.100586 | 5181981346832.210938 |
| 5m 30s | 5193779622377.418945 | 5204159468348.786133 | 5176093032615.515625 |
| 6m 00s | 5189308262554.725586 | 5200850919899.376953 | 5170399317665.348633 |
| 6m 30s | 5184965645238.834961 | 5197510202195.328125 | 5164512765508.620117 |
| 7m 00s | 5180461832133.696289 | 5194199228798.541992 | 5158859514424.513672 |
| 7m 30s | 5175974494965.203125 | 5190893764368.701172 | 5152939158268.952148 |
| 8m 00s | 5171505812054.311523 | 5187529485820.927734 | 5146792177937.200195 |
| 8m 30s | 5167053654242.847656 | 5184208784862.052734 | 5140690823568.653320 |
| 9m 00s | 5162558193041.388672 | 5180851947875.826172 | 5134568784089.215820 |
| 9m 30s | 5158053381453.726562 | 5177470862782.342773 | 5128396490981.851562 |
| 10m 00s | 5153657131562.408203 | 5174167590672.028320 | 5122341273690.202148 |

1. Discussion

The number of race condition happening between rd\_lock and wr\_lock scales with thread count:

|  |  |  |  |
| --- | --- | --- | --- |
| % of 2opt\_call with race condition | | | |
|  | before distance calculation is optimized | | Dist. Calc. optimized |
| Thread | Race% - **Chunk** | Race% - **Balanced** | Race% – **Balanced.opt** |
| 1 | 0 | 0 | 0 |
| 2 | 1.03 | 4.00 | 11.82 |
| 4 | 7.87 | 8.33 | 19.19 |
| 8 | 17.18 | 11.06 | 23.27 |
| 16 | 30.64 | 11.40 | 24.88 |

1. Feedback