ECE 459: Programming for Performance
Assignment 3

Jon Eyolfson (adapted by Patrick Lam)

March 4, 2014 (Due: March 21, 2014)

This assignment is done in groups of at most 2. The group coordinator must create a group at `http://ecegit.uwaterloo.ca/ece459/1141/a3`. You have to do this even if you are working in a group of 1.

git clone URL (recommended in the event of any changes from our side)
We will fork the provided repo for you once you create your group on `ecegit`. `ecegit` will tell you your repo path. Group members may then clone that repo.

Reminder: keep your repositories private.

Archive URL (don't use this if you're in the class)
`http://patricklam.ca/p4p/files/provided-assignment-03.tar.gz`

Handy References

- `http://www.cplusplus.com/reference/`
- `http://en.cppreference.com/w/cpp` (C++11)
- `http://en.wikipedia.org/wiki/Genetic_algorithm`
- `http://en.wikipedia.org/wiki/Selection_(genetic_algorithm)`

Important Notes

- You may modify any code or build flags, but you must use gcc/g++/clang.
- Make sure you run your program on `ece459-1.uwaterloo.ca`.
- I will replace your `solver.cpp` file when running your code.
  - Do not try to modify this file.
- You must not change any high level operations of the algorithm.
- You must not change the interface between `solver` and `gatsp`.
- If you have any questions about your limitations, email me.

Background

Lecture 15 contains the background information. The source is as high level as possible with no premature optimizations. There are many ways to improve the performance of this program. You may improve the serial execution, parallelize it as much you can and/or use compiler flags. However, you cannot change any high level operation. Do not make any modifications that change the output of the program (beyond the random aspect of the algorithm).

Report

Baseline Performance (30 marks)

First, we need a performance baseline. Take the provided application and run it for at least 10 seconds on the `berlin52.tsp` problem. The solver has two flags: `s` for specifing how many seconds your program should run and `i` for specifing how many iterations your program should run.

```
% bin/solver tsps/berlin52.tsp -s 10
```

Look at the output file (`berlin52.tour`). In the comment section, the program reports the number of iterations and the quality of the result. Now, profile your application using this set number of iterations with the profiler(s) of your choice.

```
% bin/solver tsps/berlin52.tsp -i <natural number>
```

Note: use this same number of iterations for all profiling runs, so your results are comparable. This is your baseline performance. Record which profiler you are using along with the results.

Identify which parts of the code take up the majority of the execution time. Record your observations. All of your observations should be in the context of the code in `gatsp.cpp` or `gatsp.hpp`; if built-in functions take a majority of the execution time, identify the caller, from one of those files.

Improvements (at least 2, 60 marks)

Now, it's time for improvements. You'll probably want to target the functions taking up the most time. Make at least 2 changes that will improve the performance of the program. For your most significant changes do the following:

- Provide a code listing of the diff of the change from the provided code.
- Profile only this change (using your original baseline, or another baseline for comparsion) showing a large improvement.
- Discuss why this change improves performance and how it does not change the fundamental operation of the program.

Note: a compiler flag is only considered valid for this section if it offers large speedup (at least 20%). You may use as many improvements as you want in order to get the best performance. Just briefly outline any other changes you made to improve performance.

Your Performance (10 marks)

You are going to be evaluated against the other groups in your class. It's simple; the faster you make your program, the more marks you'll get. We will put up a leaderboard on the class website, so you can see how you're doing. Your program will be tested on `berkeley`, which is a machine with `gcc`, `g++`, `clang` and CPU identical to `ece459-1`. Make sure your program runs on `ece459-1` as you expect it to. You may use any optimizations specifically targeting this architecture, although it may not help. (Remember: `ece459-1` is a Intel Core i7-2600K).