CS2101 Due: Tuesday 2nd of December 2014

Problem Set 3

CS2101 *Submission instructions on last page*

PROBBLEM 1. [100 points]

In class, we went through a tutorial on the subject of *genetic algorithms*. Here a few introductory pointers.

- `http://en.wikipedia.org/wiki/Genetic_algorithm`
- `http://www.cs.colostate.edu/~genitor/MiscPubs/tutorial.pdf`
- `http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/tcw2/report.html`
- `http://www.scholarpedia.org/article/Genetic_algorithms`
- `http://www.egr.msu.edu/~goodman/GECSummitIntroToGA_Tutorial-goodman.pdf`
- `https://www.lri.fr/~hansen/proceedings/2012/GECCO/companion/p917.pdf`
- `http://www.theprojectspot.com/tutorial-post/creating-a-genetic-algorithm-for-beginn` `3`

We will follow the terminology of Darrell Whitley in *A Genetic Algorithm Tutorial*, that is: `http://www.cs.colostate.edu/~genitor/MiscPubs/tutorial.pdf`

The goal of this problem is to realize a `Julia` implementation of a genetic algorithm solving the *Traveling Salesman Problem*. See for instance:

- `http://en.wikipedia.org/wiki/Travelling_salesman_problem` `http://www.dbai.tuwien.ac.at/staff/musliu/ProblemSolvingAI/Class9GATutorial.ppt`

Let N be the number of cities to be visited by the Traveling Salesman. We assume that the distances between each pair of cities are known and recorded in a 2D array C which can be constructed as follows (here for $= 10$) in a `Julia` session:

```
jjulia> C = [[mod(rand(Int32), N^2) for j=1:(N-i)] for i=1:N]
10-element Array{Array{Int64,N},1}:
 [64,84,58,67,93,28,98,37,48]
 [22,57,1,58,67,51,63,33]
 [70,11,13,37,26,31,89]
 [68,34,51,10,90,85]
 [33,10,7,95,53]
 [2,72,38,56]
 [37,34,84]
 [0,10]
 [78]
 []
```

In the above, the value of `C[i][j]` is the distance between the `i`-th and the `(i+j)`-th cities. Note that the coefficients in the array `C` are obtained randomly by calling `mod(rand(Int32), N^2)`. Using N^2 is important, a smaller value would make the problem too easy while a larger one would make it harder.

We denote by $\mathcal{P}$ the current generation and by P be the cardinality of the $\mathcal{P}$. Each individual in $\mathcal{P}$ is a permutation of the numbers $1, 2, \ldots, N$ and represents a possible travel for the Salesman, as discussed in class. Initialzing the population can be done as follows. For a given value of N define

```
v = [i for i=1:N]
```

then each individual $x \in \mathcal{P}$ can be initialized by calling

```
shuffle(v)
```

Indeed, each call to `shuffle(v)` produces a random permutation of the input vector `v`.

The *evaluation function* ℓ maps any individual x to the total length $\ell(x)$ of the trip encoded by x. Let $\widehat{\ell}$ be the average of a trip in $\mathcal{P}$. The *fitness function* f maps any individual x to the ratio $f(x) = \frac{\widehat{\ell}}{\ell(x)}$.

Crossover and mutation may be performed as described in one of the following tutorials which include a presentation of a genetic algorithm implementation of the *Traveling Salesman Problem*.

- `http://www.dbai.tuwien.ac.at/staff/musliu/ProblemSolvingAI/Class9GATutorial.ppt`
- `http://www.theprojectspot.com/tutorial-post/applying-a-genetic-algorithm-to-the-tra5`
- `https://www.acc.umu.se/~top/travel_information.html#GATSS_alg`
- `http://www.generation5.org/content/2001/tspapp.asp`

The selection of the individuals participating to the *recombination phase* is performed as follows:

1. $c(x) := \lfloor f(x) \rfloor$ copies of x are placed randomly in the *intermediate generation*, for each $x \in \mathcal{P}$, where $\lfloor f(x) \rfloor$ is the integer floor part of $f(x)$. Let Q be the total number of copies selected in the intermediate generation, thus we have
$$Q = \sum_{x \in \mathcal{P}} c(x).$$
2. Choose $P - Q$ individuals from $\mathcal{P}$ with the constraint that each $x \in \mathcal{P}$ has a probability $f(x)$ to be chosen if $f(x) < 1$ holds and probability $1 - 1/f(x)$ otherwise. Add those $P - Q$ individuals to the intermediate generation.

We explain below how to implement the second phasis in the selection. Define

$$p(x) = \begin{cases} f(x) & \text{if } f(x) < 1 \\ 1 - \frac{1}{f(x)} & \text{otherwise} \end{cases}$$

for each individual $x \in \mathcal{P}$. Then, for each individual x in the population:

1. choose a random number $0 \leq q \leq 1$;
2. if $q \leq p(x)$ then add x to the selected individuals.

Repeat the above procedure until the total number of selected individuals in P.

We now turn our attentiion to the stopping criterion. We suggest the following condition:

$$\frac{\max\{\ell(x) \mid x \in \mathcal{P}\} - \min\{\ell(x) \mid x \in \mathcal{P}\}}{\max\{\ell(x) \mid x \in \mathcal{P}\}} \leq \varepsilon \tag{1}$$

where $0 \leq \varepsilon \leq 1$ is prescribed in advance (the smaller the better). In practice ε could be in the order of 1%.

Question 1. [30 points] Write a serial `Julia` program implementing the above genetic algorithm for solving the Traveling Salesman Problem. The input parameters of the main function must be N, C and the initial value of $\mathcal{P}$ meanwhile its return value must be an individual (from the final generation) with optimum fitness.

Question 2. [50 points] Write a parallel `Julia` program implementing the above algorithm. Using shared arrays for encoding the current generation and one for the intermediate generation. Meanwhile, each processor will have a local copy of the array A for speeding up access time.

Clearly, the main challenge in this parallel program is the concurrent construction of the intermediate generation.

You should explain the design of your implementation and clearly document each of the functions of your code.

Question 3. [20 points] Collect running times and the number of generations for both the serial and the parallel programs for $N = 10, 15, 20, 25, 30, 35, 40$ and $P = N, 10N, 100N$. You should organize the collected data in tables or charts and discuss the observed performance results.

Submission instructions.

Format: Questions 1 and 2 involve programming with Julia: they must be submitted as two input files to be called `Q1.jl` and `Q2.jl`, respectively. Each of these two files must be a valid input file for Julia. In addition, each user defined function must be documented

and the measured running times and answers to Question 3 should be presented in a separate PDF file called `Q3.pdf`.

To summarize, each assignment submission consists of three files: `Q1.jl`, `Q2.jl` and `Q3.pdf`.

Submission: The assignment should be returned to the instructor **and** the TA by email.

Collaboration. You are expected to do this assignment *on your own* without assistance from anyone else in the class. However, you can use literature and if you do so, briefly list your references in the assignment. Be careful! You might find on the web solutions to our problems that are not appropriate. For instance, because the cache memory model is different. So please, avoid those traps and work out the solutions by yourself. You should not hesitate to contact the instructor or the TA if you have any question regarding this assignment. We will be more than happy to help.

Marking. This assignment will be marked out of 100. A 10 % bonus will be given if your answers are clearly organized, precise and concise. Messy assignments (unclear statements, lack of correctness in the reasoning, many typographical or language mistakes) may give rise to a 10 % malus.