COMSM0305 Assignment: CVRP

Your assignment is to choose and implement a solution to the Capacitated Vehicle Routing Problem (CVRP) described below.

The FruityBun Challenge

Imagine that the FruityBun bakery wants to optimize its delivery of buns to shops. The delivery problem is a CVRP. The bakery decides to run a public contest to find the most efficient solution to their current CVRP and they publish its specification. However, their orders will change in the future (e.g., adding new shops or losing shops), so they also want to know how the solution was found. Submissions will be evaluated on the quality of the solution (its cost) and how well the work was explained.

The Capacitated Vehicle Routing Problem

The CVRP is a hard combinatorial optimization problem. The scenario is that a set of depots contains goods to be delivered to customers by a fleet of trucks. Each customer wants a certain integer amount of goods called their *demand* and each truck can only carry a certain amount called its *capacity*. All trucks have the same capacity. The objective is to find a set of routes, one for each truck, which minimizes the total distance travelled by all trucks (the *cost*) and satisfies the demands of all customers. All routes begin and end at a depot. When a truck arrives at a customer it must deliver as much as the customer demands. A single customer cannot be served by more than one truck. A truck only makes one trip per solution, but the number of trucks is left as a parameter which you can optimize. A solution is only valid if no truck's capacity is exceeded and if all demand is satisfied. You cannot use invalid solutions when recording your best solution (see below), but invalid solutions can be given a fitness and participate in the search process, e.g., as parents in a genetic algorithm. (See the slides on constrained optimization for ideas.)

The fruitybun CVRP data file has a single depot and 249 customers, all of which exist in a 2-dimensional Euclidean world. The first line of the file specifies the number of depots and customers (the "DIMENSION" of the problem). The second line specifies the capacity of each truck (500). Following this the NODE_COORD_SECTION lists each node (depot or customer) on a separate line, in the format: node number, X coordinate, Y coordinate. The depot is node 1. Finally, the DEMAND_SECTION lists the demand of each customer (and 0 for the depot).

I do not know the optimal solution for this problem.

What to do

Your task is to write code to find the best solution it can to FruityBun's CVRP, and to write a poster which clearly explains what you did and why you did it that way. You can use any algorithm you like as long as it learns; you cannot just read a solution from a file, and you cannot use integer programming. You can use an algorithm mentioned in this unit, or do a literature search for specialized CVRP algorithms, or even create your own algorithm (which would most likely be a variation on some existing algorithm. It's unlikely you'll create a completely novel algorithm which is effective.)

The CVRP is a hard problem and that research is still being done on how best to solve it. As a rule of thumb, off-the-shelf learning algorithms are not as good as those which have been customized to the problem at hand, so the simple GA is not a good choice. Some of the representations and

operators discussed in the handouts are much better for this kind of problem than those used by the simple GA. Hillclimbers seem to be competitive with genetic algorithms on this problem.

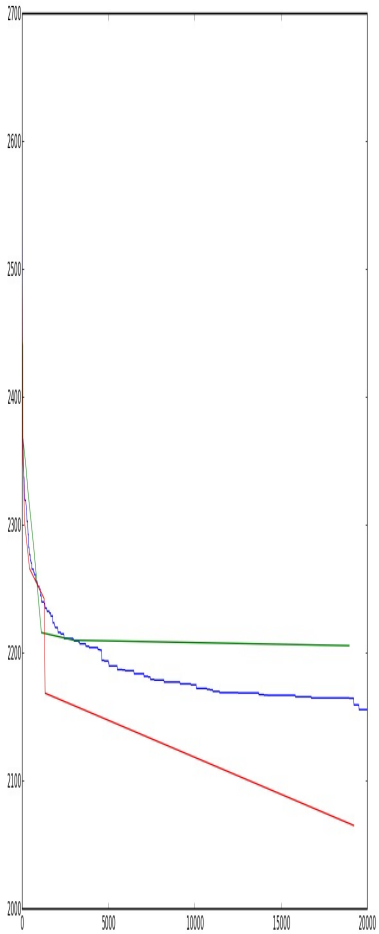
There are also some very specialized CVRP algorithms, some of which are genetic algorithms, which have been published. We will not cover them in the unit but you can search the literature for them. Specialized CVRP algorithms can exploit properties of the CVRP, for example the Euclidean geometry, and distance and demand information, which less specialized algorithms do not exploit (or do not exploit very effectively - a fitness function is a crude way to exploit Euclidean distances). I'm sure some of you will invent ways to modify existing algorithms to exploit this information.

You can pass this assignment using algorithms mentioned in the lectures, but this CVRP problem is genuinely challenging and there is plenty of scope for you to go beyond the lectures. You are not expected to find the optimal solution.

A suggested plan:

1. Start by reading this Advice on starting the CVRP assignment.
2. Start working with the FruityBun CVRP data file. You may want to extract the data from the NODE_COORD_SECTION and plot it to get a sense of the layout of the nodes. You may want to generate some random solutions to get a sense of the difficulty of the problem. (If you generate 10,000 random solutions, what's the average cost? What's the lowest cost?) (Tip: you may find it easier to cut-and-paste the data file into arrays in your program than to write code to read the file. Alternatively you can use CVRPData.java to read in the data.)
3. Choose and implement an algorithm to solve the CVRP. It should beat random search!
4. Record the cost and route of the best solution your code ever finds in a file called `best-solution.txt`, which you will submit with your code in SAFE. The format of this file is described in the *What to submit* section below.
5. Run CVRPValidater.jar to verify that your `best-solution.txt` has the correct format, that your solution is valid, and that you have computed the cost correctly. To execute it from the unix prompt issue the command: `java -jar CVRPValidater.jar`. Your `best-solution.txt` file must be in the same folder. If you want to evaluate a file with a different name you can pass the name of your solution file like this: `java -jar CVRPValidater.jar myFile.txt`
6. Optimize the parameters of your algorithm (if it has any) and the number of trucks in the solution. You may optimize the number of trucks by hand, or it may be optimized by your algorithm.
7. Make and submit your poster.
8. Submit all your code.

Many algorithms will produce better results if you run them longer, at least up to a point. E.g., genetic algorithms tend to converge/stagnate, after which time they make little or no improvement. You may want to plot the cost of the best solution found so far at each time step to see how long it is useful to run your algorithm. For example, James Marshall generated 20,000 random solutions for a different CVRP ploblem and plotted the best-so-far. He repeated this 20 times. Here is a plot of the best (bottom red curve) and worst (top green curve) of these 20 repeats, and the average of all 20 (middle blue curve). You can see random search is still finding better solutions at a steady rate (on average) so it could be worth continuing it.



What to submit

Please do not submit .zip (or other archive) files. Submit the following files:

1. **best-solution.txt**

   Put the best-ever valid solution your code finds in this file. It will be read by a script so you must follow the format exactly. It must be an ASCII text file. Line 1 must say "login ", then your UoB username, then your candidate number (which you can find in your submission folder). Line 2 must say "name " then your name. Line 3 must say "algorithm " then a description of your algorithm of at most 100 characters. It's enough to identify the general type of algorithm (genetic algorithm, hillclimbing, random search ...). Line 4 shows the cost of the solution, which you can round off to 3 decimal places. (There will be a penalty if the cost you report does not match the cost computed by the marking program to within 0.01.) The following lines must show the solution, with the route of each truck on a separate line in the format 1->5->10->1, where 1 is the depot. Here's an incomplete example file showing 2 trucks:

   ```
   login xy1234 13851
   name John Zoidberg
   algorithm Genetic Algorithm with specialized crossover and mutation
   cost 3108.323
   1->5->10->1
   1->15->23->45->7->1
   ```

2. **Your code**

Submit all your source code. You can work in any language that is part of the the department's standard Linux installation (so the marker can run it). Your code must run on the standard Linux installation without installing other libraries. You must not use any optimization libraries (e.g., genetic algorithm libraries) but you can use generic library code (e.g., Java's ArrayList.)

3. **start-cvrp**

   This is a unix shell script that starts your code. It must run under the tcsh shell on the department's linux setup. It must run without any parameters and without any input from the user. Its output must be exactly in the format of `best-solution.txt`, but it must write its output to the standard output stream (not directly to a named file). Note: this must not take more than about 30 minutes to run. Here's an example `start-cvrp` shell script to compile and start a java program called `CVRP`:

   ```
   #!/bin/tcsh
   javac *.java
   java CVRP
   ```

4. **cvrp-info.txt**

   This is an ASCII text file with the following information: i) roughly how long in minutes it takes for `start-cvrp` to return results, to give the marker an idea of how long it will take. He will want to know whether it should take about 1, 5, 10, or 30 minutes so you don't need to be precise - choosing the nearest one of those values is enough. If your code does not compile enter -1 for the time. ii) A 1-line note for the marker about running your code. Most people will not have a note, but make sure the line begins with "note-for-marker" even if you don't have anything to say. Here's an example to show the format:

   ```
   cvrp-runtime-in-minutes 15
   note-for-marker My code crashes about 1/4 of the times it runs.
   ```

5. **Poster**

   Submit a 1-page poster in PDF format named `poster-xxxxxx.pdf` where xxxxxx is your login name. Put a title and your name at the top. There is not a lot of space available on a poster so the challenge is to choose what to include and what not to include. You should focus on what is likely to be novel and interesting to the reader. So, for example, do not explain what a genetic algorithm is, but if you have used an usual crossover operator do explain it briefly. (As a rule of thumb: if a concept is explained in the lecture slides then you don't need to explain it.) There are two styles of poster. The first relies on someone to stand in front of it and explain it, whereas the second is mean to be self-explanatory and so tends to have more text. You should make the second kind. There are many posters of the second kind on the 3rd floor of MVB that you may want to look at for inspiration.

   Include the following information:

   - Why you chose the algorithm.
   - Where you got the algorithm (you invented it, modified it, found it in a journal paper, on a web page...). Cite your sources.
   - Briefly explain the representation, operators, fitness function or any other significant details *if they are not covered in the handouts.*
   - List any parameter settings you used to generate any results you are showing.
   - Anything else you want to include.

You may want to use the following, but they're just ideas. You do NOT need to use them all.

- Pseudocode
- Diagrams (e.g., to show how an operator or algorithm works)
- Best-so-far plots during the learning run
- Plots of solutions

You will not have to present your poster. Just submit it in SAFE.

Marking

The marking scheme is:

Poster	10%
The cost of your `best-solution.txt`	90%

The poster mark will be based on its content, presentation, (layout, correct reference formatting...), novelty and difficulty of the solution, and evidence of reading and insight.

It's easier to improve bad solutions than good ones, so small improvements to bad solutions will result in a small increase in marks. But when you're close to the optimal solution it's harder to make improvements, so small improvements there will result in a bigger increase in marks. If you have trouble with the assignment, remember that if you submit nothing you will get a mark of 0, but if you submit something, however limited, you should get some marks. You may submit results for random search. You can get marks for submitting the poster without any code.

It's better to submit a working version of your code with less features than a non-working version with more features, so if you're unable to fix a bug or complete an upgrade please comment out the problem parts and submit something which runs. If you cannot submit runnable code then submit what you have and make a note for the marker in `cvrp-info.txt` to say it won't run.

Marks will be deducted for not following the specification (e.g., not using the correct file formats or file names), for not including all parameter settings, for code which does not compile, and so on.

Checklist

1. Submit your `best-solution.txt` ASCII file
2. Submit all your code
3. Make sure `start-cvrp` is working and does not run longer than specified
4. Submit `cvrp-info.txt`
5. Submit `poster-xxxxxx.pdf`
6. Do NOT submit .zip or other archive files.

Questions and Answers

Q: How often do crossover and mutation occur in a Genetic Algorithm?

A: After selecting two parents you generate a random number to decide whether to perform crossover. The random number is compared to the crossover rate parameter, which is a value you

can change to optimize performance. The crossover rate is usually high (e.g., 90%). If crossover does not happen then mutation may still happen. Using point mutation each gene has an independent chance of being mutated, which is determined by the mutation rate. This is another parameter you can adjust to optimize performance. The mutation rate is normally low (e.g., 3% per gene). In the simple GA each gene is mutated independently so it's possible for zero, one or more genes to be mutated when a chromosome is mutated.

Q: Can the same parent be selected twice in the same generation?

A: The simple GA samples with replacement, so yes if you're implementing the simple GA. You can change that if you want, but it may not be a bad thing for fitter parents to be selected more than once.

Q: What is the neighbourhood for hill climbing algorithms?

A: The neighbourhood is the set of solutions that can be reached by applying an operator once to the current solution. For point mutation on a binary bitstring the neighbourhood for 0000 is 0001, 0010, 0100 and 1000. However, for the CVRP other mutation operators will work better. You would not use crossover because hillclimbers only have one current solution and crossover needs (at least) two chromosomes to use as parents.

Q: How much of one generation is kept in the next generation?

A: In the Simple GA none of the old population is deliberately kept, although it's possible for parents to be cloned or regenerated by chance. Other GAs keep the best N% of the last generation in the next one, where N is a parameter you can adjust. This is called "elitism" and it means you don't lose the best solution found so far.

Q: What should chromosomes look like for the CVRP?

A: Different representations are possible. This answer will not discuss specific solutions because designing a representation is an important part of the assignment.

Each truck should visit a customer at most once, so its route will be a subset of a permutation of customers. No customer can be visited by more than one truck, so if the set of routes overlaps the solution is not valid. See the section on Constrained Optimization in handout 5 "Issues in evolutionary computing" for ways to handle invalid solutions in evolutionary algorithms. For example, you can insert an invalid solution in the population but penalize its fitness.

Two trucks may follow routes of different lengths. There are two basic ways to encode routes of different lengths. i) All chromosomes have the same length, but some are decoded into longer routes than others. Note that not all genes need to contribute to the route; genes can be ignored if they're not needed. ii) Alternatively, your representation may use "variable-length" chromosomes, i.e., some are longer than others. Crossover is more complex with variable-length chromosomes - consider the issue of lining up cut points - but it can be done.

In addition to coding routes of different lengths, you may want to specify different numbers of trucks in different chromosomes. Again, you can handle this either by making all chromosomes the same length or giving them different lengths.

Note that the length of a route and the number of trucks are just integers, and that chromosomes can include genes which are integers. A simple mutation operator for integers is just to add or remove one from the current value.

Your choice of representation can have a big impact on how successful your algorithm is.

Q: Can I use matlab?

A: Yes. Write a shell script called `start-cvrp` which calls matlab.

Q: Can I use the demand information to improve my search?

A: Yes.

Q: How do I calculate the cost (distance) of a route?

A: Compute the Euclidean distance. If customers p and q are at coordinates p = (p1, p2) and q = (q1, q2) then the distance between them is: $\mathrm{d}(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$.

Q: What should `start-cvrp` do if I implemented more than one algorithm, or variations on an algorithm?

`start-cvrp` should run the best algorithm you found.

Q: How can I plot routes?

A: There are many options including gnuplot and these files, which you can reuse for your assignment, if you wish. Unfortunately, however, each truck's route must be in a separate file and the gnuplot scripts must explicitly plot each file. This means that you need a different script for each number of trucks in the solutions you want to plot, which can be a little inconvenient.

Here are two other ways of plotting routes:

- Barnabas Szabolcs wrote this matlab code.
- Tom Pickering wrote this java code.
- Mike Slade wrote this python code.

If you find a new and useful plotting solution you can post it on the forum, and if I think it's worth it, you will get a small bonus mark.

Q: Can I use a multiobjective approach to optimize the number of trucks?

A: In other words, can I use the route cost as one objective and the number of trucks as another?

This would make sense if you wanted to minimize the number of trucks (which seems plausible in the real world: it costs money to buy trucks, to maintain them, to store them, and if you have more trucks you need more drivers.)

But in this assignment we're only optimizing the cost of the route. It's an inherently single-objective problem so a multi-objective approach is not necessary. We don't care how many trucks

there are, as long as we get the lowest route cost we can find. We're not trying to minimize or maximize the number of trucks.

So a multi-objective approach is not an obvious one to take. However, it could be used to handle invalid solutions as discussed in handout 5. It might also be useful because it increases the diversity of the solutions or, put another way, it does more exploration. It will try to find the lowest cost solution for each number of trucks, and that could turn out to be a good way to find the lowest overall cost. I don't know how well it would work.

A multi-objective solution is useful in another way: it adapts the number of trucks. I would expect it to find a better solution than a single-objective method with a randomly-chosen fixed number of trucks, but of course choosing the number of trucks at random is not a very good method. But there are other ways of choosing or adapting the number of trucks. For one thing, the search space for the number of trucks is quite small. We know we don't need more trucks than customers, and there are only 249 customers. For such a small search space a genetic algorithm is definitely overkill.

Note on solution costs

Empty.

Note on language choice

Empty.
