Assignment 3: The Traveling Thief Problem (multi-objective)

Due date: 11:55pm, 2 November 2014

1 Overview

Assignments should be done in groups consisting of four students. Please use the Student Discussion Forum to post the details of your group (names, student numbers, and email addresses). If you have problems finding a group use the forum to search for group partners or contact the lecturer.

Each assignment has to be handed in and reasonable effort has to be made to complete each assignment in order to pass the course. For Assignment 3 the minimum requirement is that the different modules of multi-objective evolutionary algorithms for the TTP need to be implemented as described below and that multi-objective evolutionary algorithms for the TTP have to be designed and tested on the benchmarks given below.

2 Assignment

This assignment deals with the Traveling Thief Problem (TTP). You have to design different multi-objective evolutionary algorithms for this problem. In the original formulation, the goal is to maximise the profit. Here, your goals will be (1) to minimise the raw travelled distance, (2) to maximise the knapsack usage, and also (3) to maximise the profit - all three objectives are to be optimised at the same time.[1]

The instances that you should test your approach on are the nine instances listed in the *Instances and Code*-Section of the competition page http://cs.adelaide.edu.au/~optlog/CEC2014Comp/.

All programming work should be done in JAVA. You have to use Java SE 7 JDK and jMetal (http://jmetal.sourceforge.net/). jMetal implements most of the important evolutionary multi-objective algorithms and you are able to use this framework for Assignment 3. You must evaluate the TTP solutions by using the fitness function provided[2]: from there you can extract the raw travel distance and also the use of the knapsack.

You should make sure that your algorithms focus on the best solutions according to the given two objectives and achieve a good spread along the approximated Pareto front for placements in between these two extremes.

Notes:

[1] For the definitions, please revisit Assignment 2.

[2] http://cs.adelaide.edu.au/~optlog/research/ttp/TTP-JavaForDistribution.zip

- You can complete this assignment even if you have not completed Assignment 2: you can use the simple bit-flip operator used in our package for the packing plan and you can modify the TSP part using any operator presented so far (swap, inversion, scramble, ...).
- This time, there is **no** time limit for the optimisations.

Exercise 1 *Your first multi-objective optimisation (20 marks)*

Download and install jMetal. Follow the case study in Section 3.3 from the jMetal user manual (available from the jMetal website). Run NSGA-II for 10.000 generations on ZDT 2 and ZDT 3 with population sizes 10, 100, and 1.000. Visualise the six final populations.

Exercise 2 *Implementation of the TTP problem (20 marks)*

Given our objective function, write wrapper code so that you can use it inside jMetal. Examples are available in the package jmetal.problems.

Important note: jMetal assumes the minimisation of all objectives by default. This means that you need to convert the "maximisation of the knapsack usage" into a minimisation problem.

Note: you can use IntSolutionType as provided in jmetal.encodings, or you can implement your own jMetal representation based on your Assignment 2.

Exercise 3 *Implementation of the TTP variation operators (24 marks)*

Given your variation operators (mutation/crossover/repair/local search/...), write wrapper code so that you can use at least two of them inside jMetal. Examples are available in the package jmetal.operators.

Exercise 4 *Multi-objective TTP optimisation with NSGA-II (18 marks)*

Use NSGA-II as a basis for the following experiment. Given the three instances `a280*` with 280 cities, run NSGA-II once with population sizes 10 and 100 and for 100, 1.000, and 10.000 generations. Visualise the 18 final populations.

Note: it is visually pleasant to visualise multiple populations in a single plot, e.g. "NSGA-II on one instance with population size 10" and then you show the three different populations at t=100, 1.000, and 10.000 generations.

Exercise 5 *Comparison of different multi-objective optimisation algorithms (18 marks)*

Compare the performance of the algorithms NSGA-II, SPEA2 and IBEA on the instances `a280_n279_bounded-strongly-corr_01.ttp`, `fnl4461_n4460_bounded-strongly-corr_01.ttp`, and `pla33810_n33809_bounded-strongly-corr_01.ttp`. You decide on the algorithmic setup. Justify your configuration choices made.

Where are your results of your Assignment 2? Visualise your best Assignment 2 results in these plots as well.

Your progress

Over the next weeks, please post visualisations of your final populations on the Student Discussion Forum https://forums.cs.adelaide.edu.au/mod/forum/view.php?id=14630. This way we can compare visually the performance of the different approaches.

We strongly recommend that you post your first results by 19 October 2014. This encourages you to start early.

3 Marking

Marking of implementations will be done according to the following criteria:

- correct overall implementation - 20%
- design of the algorithms and justification of the operators and algorithms - 40%
- quality of the results achieved - 40%

4 Procedure for handing in the assignment

Work should be handed in using Moodle forum, with one submission per group. The submission needs to include:

- all source files
- all configuration files
- a README.txt file containing
  - instructions to run the code (yes, we will run your code)
  - the names, student numbers, and email addresses of the group members
  - the URL of your repository
- one document that (1) contains the plots of the final populations and (2) contains interpretations of the results
- the log-files containing the results generated by the code
  - please generate short log-files that still show the overall optimisation
  - if you do not provide log-files, you will receive 0 marks for the "results reports"