Evolutionary Algorithm for the SCPP: User Guide

This document explains how to compile and run the program for the evolutionary algorithm (EA) in the thesis:

- Hawa, A. L. (2020). *Exact and Evolutionary Algorithms for the Score-Constrained Packing Problem.* PhD Thesis, Cardiff University.

This code is available from https://doi.org/10.5281/zenodo.3986640 and https://asylhawa.com.

The zip file contains four folders: three folders containing results files and one folder containing the .cpp and .h files for the program. The program is implemented in C++ and can be compiled using different IDEs or the GNU compiler.

1 Compilation using Microsoft Visual Studio

The program can be compiled and executed using Microsoft Visual Studio as follows:

1. In Visual Studio, go to **File** > **New** > **Project from Existing Code**.
2. In the dialogue box, select **Visual C++** and click **Next**.
3. Select the directory containing the above .cpp and .h files and give the project a name, then click **Next**.
4. Select **Console Application Project** for the project type, and then click **Finish**.

The source code can then be viewed and executed within Visual Studio. Release mode should be used during compilation to make the program execute at maximum speed.

2 Compilation using CLion

The program can be compiled and executed using CLion along with the provided CMakeLists.txt file as follows:

1. In the Welcome dialogue box, select **Open or Import**.
2. Select the directory containing the above .cpp and .h files and click **OK**.

The source code can then be viewed and executed within CLion. Note that the CMakeLists.txt file must be in the folder along with the .cpp and .h files. Release mode should be used during compilation to make the program execute at maximum speed.

3 Compilation using g++

The program can be compiled and executed with g++ using the commmand line by navigating to the directory and using the following command:

```
g++ *.cpp -O3 -o myProgram
```

This creates an executable program called `myProgram`. Note that any name can be used in place of `myProgram`.

4 Running the Program

The executable file can be run from the command line. To run the executable on Unix machines use the command `./myProgram <arguments>`. On Windows, use the command `myProgram <arguments>`.

If the program is called with no arguments, program information and usage is provided on screen. The user input arguments are as follows:

- `.txt`: SCPP instance file must be the first argument (`PAn_xx.txt` or `PRn_xx.txt`).
- `-e`: Time limit for the EA. Default = 600s.
- `-t`: Minimum scoring distance τ. Default = 70.
- `-W`: Bin capacity. Default = 5000.
- `-p`: Number of solutions in the population. Default = 25.
- `-x`: Recombination operator. 1 = GGA, 2 = AGX, 3 = AGX$'$. Default = 1.
- `-s`: Seed value. Default = 1.

The first argument must be the input file of the SCPP problem instance. A problem instance generator for the SCPP along with a specification of the input files and the problem instance files used for experiments in the above publication can be found at https://doi.org/10.5281/zenodo.3986636 and https://asylhawa.com. For reference, an example input file called `PA1_1.txt` is provided. The remaining arguments for the program are optional and are allocated default values if left unspecified. Here are some example commands, using `myProgram` as the program name (on Windows):

```
myProgram PA1_1.txt
```

This executes the program on the problem instance given in the file `PA1_1.txt` using a run-time of 600 seconds, a minimum scoring distance τ of 70, bins of size 5000, 25 solutions in the population, the GGA recombination operator, and a random seed of 1.

```
myProgram PA1_1.txt -e 300 -t 50 -W 2500 -p 10 -x 3 -s 7
```

This executes the program on the problem instance given in the file `PA1_1.txt` using a run-time of 300 seconds, a minimum scoring distance τ of 50, bins of size 2500, 10 solutions in the population, the AGX′ recombination operator, and a random seed of 7.

Interpretations of the run-time parameters for the various algorithms can be found in the above publication.

5 Output

The output file is a single text file, `EInWx_t.txt`, created by the program. The name of the output file should be interpreted as follows:

- `E`: Evolutionary algorithm output file.
- `I`: Instance type. A = Artificial, R = Real.
- `n`: Number of items/100.
- `W`: Bin capacity/1000 (rounded down to the nearest integer).
- `x`: Recombination operator. 1 = GGA, 2 = AGX, 3 = AGX′.
- `t`: Minimum scoring distance τ.

The instance type (artificial or real) and the number of items are taken from the given input file. The recombination operator, bin capacity, and minimum scoring distance τ are taken from the program arguments.

Each line in the output file contains the following results for each instance (in order):

- Instance number.
- Lower bound - theoretical minimum (t).
- Number of bins in the best solution (#S).
- Solution quality - #S/t (Q).
- Fitness value of the best solution.
- Number of EA iterations.
- Proportion of sub-SCPP instances with feasible solutions.
- Time taken to find the best solution.

The instance number is taken from the given input file. For the experiments in the above publication, batch files are used for 1000 problem instances with different program arguments.

Another file, `TEInWx_t_inst.txt`, is also created for each problem instance that records when a better solution is found during the EA. Line 1 of each file contains the following information for the problem instance:

- Instance number.
- Number of items.
- Instance type (1 = Artificial, 2 = Real).
- Bin capacity.
- Recombination operator (1 = GGA, 2 = AGX, 3 = AGX′).
- Lower bound - theoretical minimum (t).

Line 2 states the column headings, and Lines 3 onwards state the number of bins in the best-so-far solution, the fitness of the best-so-far solution, and the time at which the solution is found.

6 Results

In the results folders, EA51, EA70, and EA92, there are six spreadsheets named `EIn_t.xlsx`. Each spreadsheet consists of nine sheets, named in the same format as the output files, which contain results for 1000 problem instances using the stated input arguments. These results correspond to Tables 5.1 to 5.3 in the above publication.

- The number of items `n` used are 1=100, 5=500, and 10=1000.
- The bin capacities `W` used are 1=1250, 2=2500, and 5=5000.
- The minimum scoring distances (τ) `t` used are 51, 50, 70 and 92.

Note that in all of our experiments in the above publication, the seed value `-s` is set equal to the instance number of the problem instance file.

- $\tau = 92$: $\delta = 0.25$.
- $\tau = 70$: $\delta = 0.5$.
- $\tau = 51/50$: $\delta = 0.75$, where 51 is used for artificial instance types and 50 is used for real instance types.

7 Copyright Notice

Please direct any queries or comments to Asyl Hawa: email: HawaA@cardiff.ac.uk, web: https://asylhawa.com.

A. L. Hawa (Last updated Friday, 28th August 2020)