Problem Instance Generator for the SCPP: User Guide

This document explains how to compile and run the program for the problem instance generator used 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.3986636 and https://asylhawa.com.

The zip file contains two folders: one folder with six sub-folders containing problem instances 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:

- `-i`: Instance number. Default = 1.
- `-t`: Minimum scoring distance τ. Default = 70.
- `-n`: Number of items. Default = 100.
- `-a`: Minimum score width. Default = 1.
- `-b`: Maximum score width. Default = 70.
- `-m`: Minimum item width. Default = 150.
- `-M`: Maximum item width. Default = 1000.
- `-c`: Instance type. 1 = Artificial, 2 = Real. Default = 1.
- `-s`: Seed value. Default = 1.

All 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
```

This executes the program and creates an artificial problem instance using an instance number of 1, a minimum scoring distance τ of 70, 100 items, a minimum score width of 1, a maximum score width of 70, a minimum item width of 150, a maximum item width of 1000, and a random seed of 1.

```
myProgram -i 7 -t 50 -n 500 -a 10 -b 40 -m 200 -M 800 -c 2 -s 7
```

This executes the program and creates a real problem instance using an instance number of 7, a minimum scoring distance τ of 50, 500 items, a minimum score width of 10, a maximum score width of 40, a minimum item width of 200, a maximum item width of 800, and a random seed of 7.

5 Output

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

- `P`: Problem instance output file.
- `I`: Instance type. A = Artificial, R = Real.
- `n`: Number of items/100.
- `inst`: Instance number.

Each line in the output file contains the following information:

1. Instance number.
2. Number of items.
3. Instance type: 1 = Artificial, 2 = Real.
4. All scores: score widths in non-decreasing order.
5. Partners: score width indices and corresponding partner index.
6. Item widths: score width indices and corresponding item width.
7. Item number: score width indices and corresponding item number.
8. Total width of all items.

Lines 9 to 13 are only produced for real instance types.

9. Number of item types.
10. Number of items of each type.
11. Value of smaller score width for each type.
12. Value of larger score width for each type.
13. Item width for each type.
14. Type number: score width indices and corresponding type number.

6 Results

The six sub-folders each contain 1000 problem instance files for artificial and real instance types. All problem instance files were created using the same seed value `-s` as the instance number `-i`.

- The number of items `n` used are 1=100, 5=500, and 10=1000.
- The minimum scoring distances (τ) `t` used is 70.

7 Copyright Notice

Redistribution and use in source and binary forms, with or without modification, of the code associated with this document are permitted provided that up-to-date citations are made to both the reference at the top of this document and the zenodo reference. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. This software is provided by the contributors "as is", and any express or implied warranties, including, but not limited to, the implied warranties or merchantability and fitness for a particular purpose are disclaimed. In no

event shall the contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory or liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage. This software is supplied without any support services.

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)