Trapezoid Packing Algorithm User Guide

This document contains descriptions on how to compile the algorithms presented in the paper:

- Lewis, R. and P. Holborn (2017) 'How to Pack Trapezoids: Exact and Evolutionary Algorithms'. *IEEE Transactions on Evolutionary Computation*, vol. 21(3), pp. 463-476.

This code is downloadable from http://rhydlewis.eu/resources/trapezoid.zip

Once downloaded and unzipped, you will see that the directory contains seven .cpp files and seven .h files. These will compile into a single executable. This algorithm is programmed in C++ and has been successfully compiled in Windows using Microsoft Visual Studio 2015 and in Linux using the GNU Compiler g++. Instructions on how to do this now follow. The folder also contains a further subfolder with three comma separated files. These correspond to Tables 1–3 in the above publication.

Compilation in Microsoft Visual Studio

To compile and execute using Microsoft Visual Studio the following steps can be taken:

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

The source code for this algorithm can then be viewed and executed from the Visual Studio application. Release mode should be used during compilation to make the programs execute at maximum speed.

Compilation with g++

To compile the source code using g++ at the command line, navigate to the directory and use the following command:

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

By default this will create a new executable program called **myProgram** that can then be run from the command line (you can choose your own name here). The optimisation option **-O3** ensures that the algorithms execute at maximum speed. A Makefile is also provided if this is preferred.

Usage

Once generated, the executable file can be run from the command line. If the program is called with no arguments, useful usage information is printed to the screen:

```
-------------------------
USAGE:
inputFile <filename>
---- Optional Run Parameters ------------------------------
-t  <int>         (Time limit. Default = 300s)
-i  <int>         (Iteration limit. Default = undefined)
-r  <int>         (Random seed. Default = 1)
-k  <int>         (Target num. groups. Default = TMin, otherwise stops when target is met or when -t or -i is met)
---- Optional Run Settings --------------------------------
-l  <int>         (Local search. Default = 1. If set to zero, no LS takes place; else -l rounds of LS are carried out per cycle)
-a  <int>         (Algorithm. Default = 1.
                        1: HillClimber;
                        2: GA with GGA recombination;
                        3: GA with GPX recombination;)
```

```
                        4: GA with GPX recombination where whole groups are eliminated.)
-s  <int>         (Subproblem algorithm. Default = 2.
                        1: Greedy + Flips;
                        2: Exact (Euler-Splice) Method.)
-o  <int>         (Order Heuristic. Default = 1.
                        1: Put items in random order
                        2: Random order, but with all items >50% of capacity first)
-g  <int>         (Genetic Repair Heuristic. Deafult = 1.
                        1: Choose bin with common item at random;
                        2: Choose bin with most common items;
                        3: Choose bin with common item that has the most spare capacity.)
-p  <int>         (Population size (GAs only). Default = 10.)
-np <int>         (Number of parents per recombination (GAs only). Default = 2.)
---- Optional Output Settings -----------------------------
-v                (Verbosity. If present screen output is produced)
-d                (Diversity. If present, diversity is output to a file and the screen at every iteration. Only used with EAs. Warning:
                      Makes the program slower.)
-------------------------------------------------------------
NOTES:
If -t is set to a value then iterations (-i) is not used as a stopping criteria.
If -i is set to a value then time (-t) is not used as a stopping criteria.
If both -i and -t are set then the run ends when BOTH have been met.
Order heuristic is used in the GAs only (for generating initial solutions and with the mutation operator).
Genetic repair heuristic is only used with GA recombination. It is used as part of the set covering sub-problem for deciding which item
    types to delete from which bin during repair.
Number of parents per recombination (-np) must be less than the population size defined by -p
```

The input file should contain the trapezoid packing problem to be solved. This is the only mandatory argument. A specification of the input files, together with all problem instances used in the above paper can be found at the following link: http://rhydlewis.eu/resources/trapBoxProbs.zip. For reference, an example input file called problem.txt has also been 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:

```
myProgram problem.txt
```

This will execute the algorithm on the problem given in the file problem.txt, using the default of 300 seconds (no iteration limit), a random seed of 1, the hill-climbing approach with the Euler-Splice algorithm, with one round of local search per cycle.

Another example command is:

```
myProgram problem.txt -r 1234 -t 100 -a 2 -p 20 -np 2 -d -v
```

This run will be similar to the previous, but will use the random seed 1234, a time limit of 100 seconds, the second algorithm (i.e., the evolutionary approach with the GGA recombination operator), a population of size 20, two parents per crossover, and outputting the diversity to the screen at each generation. Finally, the presence of –v ensures that output is also written to the screen.

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

Output

When a run of the program is completed, a number of text files are created. The most important of these are the files tEffort.txt (time effort) and resultsLog.txt. The first of these specifies how long (in terms of milliseconds respectively) solutions with a certain numbers of bins took to generate during the most recent run. For example, we might get the following time effort file after a run.

```
40   126
39   427
38   835
37   1187
36   1714
35   2685
34   6849
33   X
```

This is interpreted as follows: The first feasible solution observed used 40 bins and took 126ms to achieve. A solution with 39 bins was then found after 427ms, and so on. To find a solution using 34 bins, a total of 6,849ms was required. Once a row with an X is encountered, this indicates that no further improvements were made: that is, no solution using fewer bins than that indicated in the previous row was achieved. Therefore, in this example, the best solution found used 34 bins. For consistency, the X is always present in a file, even if a specified target has been met.

The resultsLog.txt file is used to store basic information on each run performed. A single line is added to this file after each run of the algorithm and contains the following information (in this order):

- Name of problem instance;
- Number of items in the problem;
- Number of different item types in the problem ;
- Target number of bins (-k);
- tMin (calculated during execution);
- Time limit in ms (value for -t multiplied by 1000);
- Iteration limit (-i);
- Random seed (-r);
- Algorithm type (-a);
- Local search (-l);
- Sub-problem Method (-s);
- Order heuristic (-o);
- Genetic repair heuristic (-g);
- Diversity on or off;
- Population size (if using an EA);
- Number of parents (if using an EA);
- Number of bins in the best solution seen during the whole run;
- Best solution cost observed across the whole run;
- Duration of the run (in ms);
- Number if iterations performed during the run.

A further six columns then contain information on how the algorithms for packing items into a single bin (the sub-problem algorithms) were used. These can be safely ignored.

Copyright notice

Please direct any queries/comments to Rhyd Lewis: web: www.rhydLewis.eu, email: LewisR9@cf.ac.uk

R. Lewis (Last updated Thursday, 27 September 2018)