BloBB's Manual

Hayward Chan

September 29, 2003

Contents

1 Overview **1**

2 Usage and File Formats **2**

3 Basic Options **2**
- 3.1 Floorplan 2
- 3.2 Block Orientations 2
- 3.3 Algorithm 3

4 Advanced Options and Parameters **3**
- 4.1 Parameters for branch-and-bound, backtracking and enumeration 3
- 4.2 Parameters for slicing hierarchical packing 4
- 4.3 Interface options 4

5 Converters and Other Utilities **5**

6 Bug Reports **5**

1 Overview

BloBB (Block-packing with Branch-and-Bound) is an open-source software for rectangle block-packing. It handles a wide range of formulations of the problem, such as free/fixed-orientation, slicing/non-slicing and optimal/heuristic. In addition to be an optimal solver for small instances, it also handles large instances in a hierarchical, heuristic manner. Its performance is competitive with those state-of-the-art.

2 Usage and File Formats

BloBB is evoked as follows:

```
blobb <input-file> <output-file> [options and parameters]
```

The order of the options is not important. If multiple options of the same catagory appear, however, BloBB 's behavior is undefined.

The input file is expected to be in `txt` format, which describes the dimensions of the n blocks as in Fig.1. For simplicity, no comments is allowed in the first $n+1$ lines of the files, but everything beyond that is ignored. If BloBB searches for packing where blocks are free to rotate by $\frac{\pi}{2}$, then the order of the dimensions for each block does not matter.

The output file is expected to be in `bbb` format, which describes a packing as in Fig.2. The location of a block refers the location of its lower-left corner. Note that the width and height of a block depends on its orientation in the packing. Therefore, if a block is rotated by $\frac{\pi}{2}$, its width and height will be swapped as they are in the input files. Note that the extensions of the input and output files are not critical, but we will use extensions `txt` for input and `bbb` for output files to throughout this manual. Converters between `txt`, `bbb` and other formats such as `blocks/pl/nets` and YAL format are described in Section 5.

```
n
<width 1> <height 1>
<width 2> <height 2>
...
<width n> <height n>
```

Figure 1: `txt` format input file template

```
<width of the packing>
<height of the packing>
n
<width 1> <height 1>
<width 2> <height 2>
...
<width n> <height n>
<blank line>
<x-location 1> <y-location 1>
<x-location 2> <y-location 2>
...
<x-location n> <y-location n>
```

Figure 2: `bbb` format output file template

3 Basic Options

In each of the basic options, at most one option can be chosen from each catagory below. Multiple specification of a catagory leads to undefined behavior. All the options are optional. If no option from a catagory is specified, the default options would be chosen. Except `--hierarchical` and `--non-slicing`, all combinations of options from different catagories are supported.

3.1 Floorplan

Non-slicing packings The options `--general`, `--non-slicing` or `-n` set BloBB to look for non-slicing packings. Non-slicing here means *not necessarily slicing.* The underlying representation is O-Tree.

Slicing packings ***(default)*** The options `--slicing` or `-s` restrict BloBB to consider slicing packings only. The underlying representation is normalized Polish expressions.

3.2 Block Orientations

Free orientation ***(default)*** The options `--free-orient` or `-fr` relax BloBB to look for packings in which the blocks are allowed to rotate by $\frac{\pi}{2}$.

Fixed orientation The options `--fixed-orient` and `-fx` restrict BloBB to consider packings in which the blocks are *not* allowed to rotate.

3.3 Algorithm

Branch-and-bound The options `--optimal` or `-o` commands BloBB to search for an optimal packing (smallest area) of the given set of blocks. For example, the following sets BloBB to look for an optimal non-slicing packing of blocks given by `input.txt` and save it to `output.bbb`:

```
blobb input.txt output.bbb --non-slicing -o
```

Backtracking The options `--backtrack` or `-b` commands BloBB to search for a packing with deadspace less than a threshold. The threshold is specified by following `--deadspace_percent` or `-d`. For example, the following sets BloBB to look for a slicing packing with deadspace less than 4.30%:

```
blobb input.txt output.bbb -s -b -d 4.3
```

Enumeration The options `--enumerate` or `-e` commands BloBB to enumerate all non-symmetric solutions. The last solution is saved in the output file while all the others (possibly more) are saved in the intermediate files. The intermediate files share the same prefix and suffix `.bbb`, and numbered from 0. The last few intermediate files store the non-symmetric solutions. The prefix is specified by following `--filename_prefix` or `-f`. In the following example, BloBB enumerates all non-symmetric optimal slicing solutions and stores all the intermediate solution in files `temp0.bbb`, `temp1.bbb` etc.

```
blobb input.txt output.bbb -s -e -f temp
```

Hierarchical Packing ***(default)*** The options `--hierarchical` or `-h` commands BloBB to pack the blocks hierarchically. This does not guarantee optimal solutions but can handle much more blocks (more than 1000). Only hierarchical slicing packing is supported at this moment. The adjustable parameters are described in Section 4.2. The following example evoke BloBB to look for a sub-optimal slicing packing.

```
blobb input.txt output.bbb --slicing --hierarchical
```

4 Advanced Options and Parameters

In the following sections, we specify a numerical parameter `--PARAM` by appending it a space and then the desired value. Table 1 shows the adjustable parameter and their meaningful ranges. Note that BloBB is *not* responsible for all error-checking to ensure the parameters are meaningful. For example, we specify `--deadspace_percent` as follows.

```
blobb input.txt output.bbb -s -b --deadspace_percent 4.3
```

4.1 Parameters for branch-and-bound, backtracking and enumeration

One cannot know what the parameters do without understanding the algorithms. In these 3 algorithms, BloBB first sets a deadspace percentage upper bound b%, specified by `--ENG_INIT_DEADSPACE_PERCENT`, and considers only the solution with deadspace less than it. If no solution is found, the upper bound in area ($A \times (1 + b/100)$, where A is total block area) is multiplied by the deadspace increment α, specified by `--ENG_DEADSPACE_INCRE`. The same is continued until a solution is found.

4.2 Parameters for slicing hierarchical packing

Under `--hierarchical` mode, BloBB first groups the blocks into clusters, whose size is specified by `--HIER_CLUSTER_BASE` (s). Each cluster has size at most $s + 1$, and the deviation in total block area among clusters is control by `--HIER_CLUSTER_AREA_DEV`. Within each cluster, similar blocks (blocks with similar longer and shorter edgess) are considered symmetric. The degree of similarity accepted by BloBB is controlled the parameter `--HIER_SIDE_RESOLUTION` (β). For β to be meaningful, it must lie in the range $[0, 2]$.

Similar to branch-and-bound, when BloBB packs each cluster, it sets a upper bound in deadspace percentage and increases it until a solution is found. The deadspace percentage and increment are controlled by `--HIER_BEST_AREA_INCRE` (γ). At first, the area upper bound is set to be $A\gamma$. If no solution is found, it increases the area upper bound to $A\gamma^2$, and similarly until a solution is found. BloBB looks for optimal solution, whose aspect ratio is not extreme. The aspect ratio tolerance is controlled by `--HIER_AR` (R) and `--HIER_AR_INCRE` (δ). At first, the aspect ratio tolerance is set to be R. If no solution is found, the aspect ratio tolerance is increased to $R\delta$, and similarly until a solution is found. (This constraint is not impose at the top level of the hierarchy.)

4.3 Interface options

The boolean parameters `--PARAM` below are turned *true* by `--PARAM` and *false* by `--nPARAM`. Options `--verbose` or `-v` turn them all to be *true* while `--terse` or `-t` turn them *false*. All parameters below are set to be *true* by default.

`--INF_SHOW_INTERMEDIATES` shows the intermediate solutions found during runtime.

`--INF_SHOW_LANDMARKS.` Algorithms with non-slicing packings are generally slow. The landmarks help the user to locate where BloBB is at in the whole solution space. It is only relevant in `--non-slicing` mode.

`--INF_SHOW_PRUNED_TABLE` shows the number of nodes visited and left after pruning at each layer of the search tree. It is ignored in `--hierarchical` mode.

`--INF_SHOW_SIMILARITY_TABLE` shows which blocks are symmetric to each other. It is ignored in `--hierarchical` mode.

`--INF_SHOW_POLISH_EXPRESSION` shows the Polish expression of an optimal slicing solution. Ii is ignored in `--non-slicing`, `--hierarchical` and `--enumerate` modes.

Table 1: Adjustable parameters and their default values

Parameter	Example (default value are used)	Range
`--deadspace_percent`	`--deadspace_percent 4.00`	$[0, \infty)$
`--ENG_DEADSPACE_INCRE`	`--ENG_DEADSPACE_INCRE 1.05`	$(1, \infty)$
`--ENG_INIT_DEADSPACE_PERCENT`	`--ENG_INIT_DEADSPACE_PERCENT 5.0` *(fixed orient)*	$[0, \infty)$
	`--ENG_INIT_DEADSPACE_PERCENT 10.0` *(free orient)*	
`--HIER_CLUSTER_BASE`	`--HIER_CLUSTER_BASE 8`	$\{2, 3, 4, \ldots\}$
`--HIER_AR`	`--HIER_AR 1.5`	$[1, \infty)$
`--HIER_AR_INCRE`	`--HIER_AR_INCRE 1.5`	$(1, \infty)$
`--HIER_BEST_AREA_INCRE`	`--HIER_BEST_AREA_INCRE 1.05` *(fixed orient)*	$(1, \infty)$
	`--HIER_BEST_AREA_INCRE 1.10` *(free orient)*	
`--HIER_CLUSTER_AREA_DEV`	`--HIER_CLUSTER_AREA_DEV 2`	$(1, \infty)$
`--HIER_SIDE_RESOLUTION`	`--HIER_SIDE_RESOLUTION 1.9`	$[0, 2]$

5 Converters and Other Utilities

Since BloBB needs only the dimensions of the blocks, `txt` and `bbb` formats are designed to simplify I/O processes. The following utilities support useful convertions to or from formats commonly used in VLSI design. More information about the `blocks/pl/nets` and YAL format can be found in `http://www.cse.ucsc.edu/research/surf/GSRC/MCNC/recipe.html`. In addition to the converters, programs that sort the blocks according to area are also included. Unless otherwise specified, the syntax to evoke each program is as follows.

```
<program> <input-file> <output-file>
```

txt2blocks converts a `txt` input file to a set of three files in `blocks`, `nets` and `pl` formats respectively. The following example creates `output.blocks`, `output.nets` and `output.pl`.

```
txt2blocks input.txt output
```

txt2yal converts a `txt` input file from to YAL format.

blocks2txt converts a `blocks` input file to `txt` format.

bbb2pl converts a `bbb` file to a `pl` file.

txtascend sorts the blocks in `<input-file>` in ascending order of area, and saves the arrangement in `<output-files>`. Both files are in `txt` format.

txtdescent. Similar to `txtdescent`, it sorts the blocks in descending order of area.

plotout is evoked as follows.

```
plotout <input-file> <postscript-file-name>
```

It reads in `<input-file>` which is in `bbb` format and a text file `out`. `out` saves the information of `<input-file>` in a format readable by `gnuplot`. Evoking `gnuplot` with `out` produces the postscript file as desired. In other words, the following sequence of commands produces a plot of packing without labels in `output.ps` specified by `output.bbb`. The flag `-n` plots the blocks with no labels, while `-l` indexes the blocks from 0 to $n - 1$. Both flags are optional.

```
plotout output.bbb output.ps -n
gnuplot out
```

6 Bug Reports

We gratefully appreciate bug reports or suggestions to make BloBB a better software. Please contact us at `hhchan@umich.edu`.