3DPlacement Documentation

计53 唐适之 2015011308

计53 张耀楠 2015011295

目录

[Introduction 1](#_Toc453958770)

[Program Overview 1](#_Toc453958771)

[Build and Run 1](#_Toc453958772)

[Main workflow 2](#_Toc453958773)

[Detailed reference 2](#_Toc453958774)

[Experiment Result 3](#_Toc453958775)

Introduction

3DPlacment is a program to place several 3D cubes (or blocks) in an as small as possible volume, using the T-tree algorithm from Temporal Floorplanning Using the T-tree Formulation [iccad04]. We used it to experimentally generate a placement that keeps that wasted volume rate less than 10% for maximized number of cubes.

Program Overview

Build and Run

To build the program, go to directory src, and execute make. It will generate the binary executable 3DPlacement\_release. This program can also be built in debug mode via executing make debug, so as to activate correctness verification in every step of the algorithm. Enabling debug mode will not interfere the original build and it will generate 3DPlacement\_debug. One can also build a test program via executing make test to get an executable test. test will do several random modifications on the T-tree to check the correctness.

Run 3DPlacement with 3 arguments, i.e.

3DPlacement\_release <inputFile> <configFile> <outputFile>

to input the 3D cubes from *inputFile* to calculate the minimized placement and output to *outputFile*, using the configurations from *configFile*. For example,

*3DPlacement\_release blocks.in blocks.conf blocks.out*

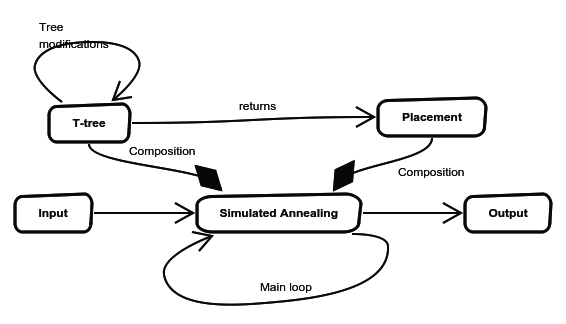
The format of the *inputFile* is as below: The first line contains the number of the cubes. Each line below contains three numbers indicating the length (on x axis), width (on y axis) and height (on z axis) of that cube. A helper program *gen* should be built at the same time, and can be used to generate *n* random cubes with lengths, widths and heights in range [1,10]. Variable *n* is input from the command line.

The format of *configFile* can be referred to *testcase/blocks.conf*. This file is able to contain comments do describe the details.

The *outputFile* is of the format below: Each line contains 7 number which describe one cube in the placement. The first number is *id*, which indicates which one it is in the *inputFile*. It counts from zero. 3 following number indicate the coordinates of the nearest lower-left corner of that cube. The last 3 number indicate the current length, width and height of that cube, for cubes can rotate in a placement.

When running *3DPlacement*, it will report some parameters in each 100 steps of the simulated annealing, which make manual optimization easier.

Main workflow

The main workflow can be described in the figure below:

Each loop of the simulated annealing calls a tree modification, and then generated a placement from the tree. The placement is evaluated and decided whether to accept. If the placement is rejected by simulated annealing, the tree modification will be undone.

Detailed reference

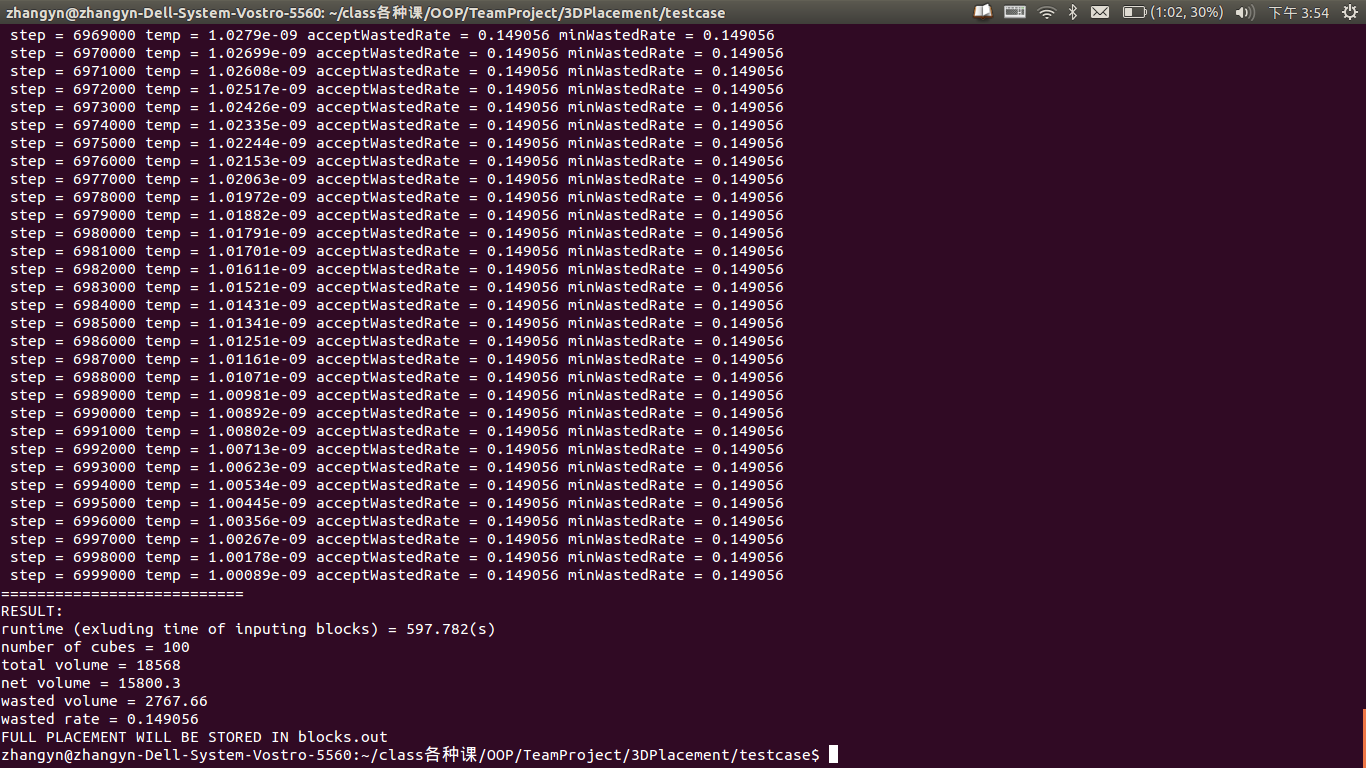
Detailed reference for each classes and functions can be referred to *doc/reference/html/index.html*, which is generated by *Doxygen*.

Experiment Result

In <configFile> , five parameters are needed. They are starting temperature, ending temperature, probability to do randomMove, probability to do randomSwap , and the number of how many steps. You can use “#” to write some tag in it.

In ./testcase, You can find three files, blocks.in, blocks.conf and blocks.out , which are <inputFile> <configFile> <outputFile> in this experiment.

In this experiment showed int testcase , with starting temperature 0.0000005 and ending temperature 0.000000001 , probability to do randomMove 0.4,probability to do randomSwap , we get the min waste rate 14.9% in 7000000 steps, 597s, with 100 blocks. That's the best result we have got in 10 minutes.



Followed are some other data when we try other parameters.（next page）

