Term Project GuideLine

EECE667 2022 Fall Term Project in POSTECH

Contents

!! Plagiarism Warning !!

Discussion among the student is allowed only in speech.

However, sharing pseudo code or seeing others' code will be **strictly forbidden**.

We will run the plagiarism detection program, and if it detects your code as positive and TA also determines that, then you will get an F grade for this course.

1. **Don't even see other student's code**
2. **Don't share the pseudo code.** Only discussion in speech is allowed.

Plagiarism will be strictly reacted as **severe punishment**.

Introduction

The parts that you should implement are two things; `Quadratic Placement` and `Your own placement` .

1. Quadratic Placement (30pt)

This is about what you learned in `Placement` Chapter of the class.

You should implement Quadratic placement using a given data structure or using your own data structure.

You need to make a binary file whose name is `quadratic_placer` (This binary file has to be able to rebuild by TA), and that binary file should do a quadratic placement.

If you more information about implementing quadratic placer, then you can refer to the pdf in `doc/ref/QP.pdf` . This material is from the Coursera Course, `VLSI CAD Part II: Layout` , from the `University of Illinois/NCSA` .

2. Your Own Placement (70pt)

In this part, you have to implement your own placement. The algorithm can be from your own idea, any journal, paper, or textbook. If you use other ideas from outside of your own, then annotate where the idea is from on the code.

Code Implementation

1. **You should submit very brief your own reports about your own code.**
   This will not affect on your score, but only be refered for preventing plagiarism.
2. **Write your own comments in the code as much as you can.**

1. Quadratic Placement

Unless you use your own data structure, then you only need to write your own code in `void Circuit::quadraticPlacement()` function of `src/algorithm/place.cpp` file.

When you implement quadratic placement, then you will need to solve the below matrix.

$$Ax = b \tag{1}$$

Here, the size of A matrix will be n x n where n is the cell number in the circuit. In average, the cell number will be about `2e5` $(2 * 10^5)$, therefore we can not get the x in the reasonable memory (about `4e10` components will be needed for only one matrix). Therefore, you should use `sparse matrix` data structure to solve the above problem.

The `sparse matrix` data structure is implemented as `coo_matrix` class. If you see the `solve_example()` function in `src/algorithm/math/matrixSolver.cpp` file, you can see how to use this data structure. The example code shows how to use `coo_matrix` object and how to solve x with the above matrix problem.

In the code, the A matrix is same with below.

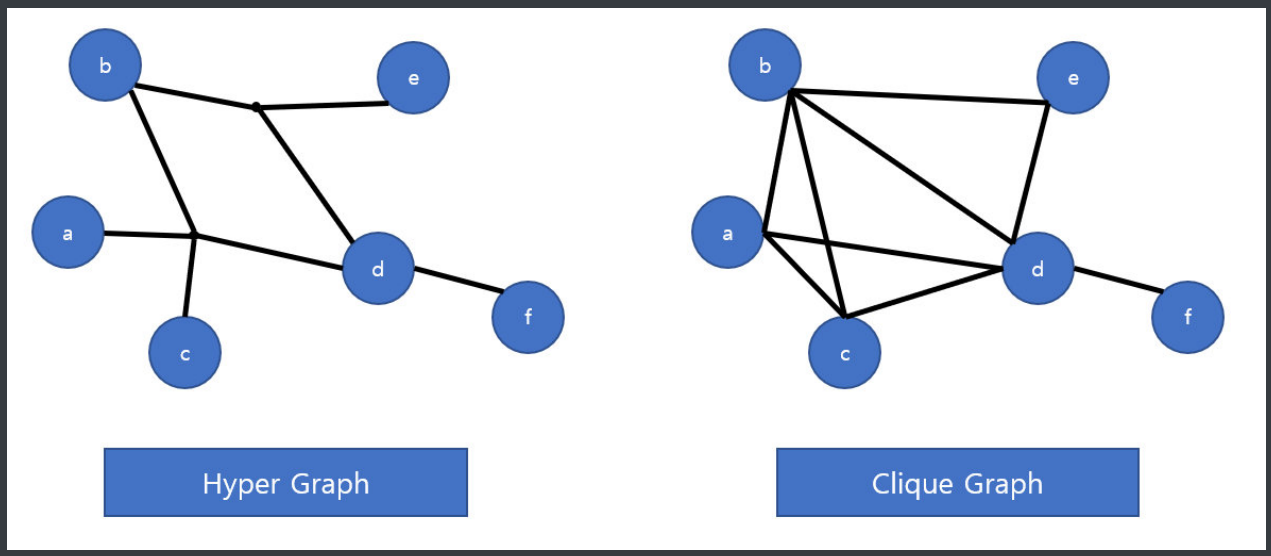
$$A = \begin{pmatrix} 4.0 & -1.0 & 0 \\ -1.0 & 4.0 & -1.0 \\ 0 & -1.0 & 4.0 \end{pmatrix} \tag{2}$$

The example code is also in below.

```
void solve_example() {
  // matrix solve example
  cout << endl << "** small demonstration **" << endl;
  coo_matrix A;
  int row_idx[] = {0, 0, 1, 1, 1, 2, 2};
  int col_idx[] = {0, 1, 0, 1, 2, 1, 2};
  double data[] = {4.0, -1.0, -1.0, 4.0, -1.0, -1.0, 4.0};
  // component # excepting zero value
  int data_number = sizeof(row_idx) / sizeof(int);

  // initializing coo_matrix object
  A.n = 3;  // 3x3 matrix
  A.nnz = data_number;
  A.row.resize(data_number);
  A.col.resize(data_number);
  A.dat.resize(data_number);

  // value inserting in coo_matrix object
  A.row = valarray<int>(row_idx, A.nnz);
  A.col = valarray<int>(col_idx, A.nnz);
  A.dat = valarray<double>(data, A.nnz);

  // initialize as [1, 1, 1] for golden solution
  valarray<double> x(1.0, A.n);
  valarray<double> b(A.n);
  A.matvec(x, b); // b = Ax
```

```
  cout << "b should equal [3,2,3]" << endl;
  cout << "b = ";
  print_valarray(b);

  // make we don't know the x value
  for (int i = 0; i < A.n; ++i) {
    x[i] = (double) random() / (double) RAND_MAX;
  }

  // solve for x
  cout << endl << "x = ";
  print_valarray(x);
  A.solve(b, x);
  cout << "after solve" << endl;
  cout << "x = ";
  print_valarray(x);
}
```

First, you should make the A matrix. When make A matrix, consider the hyper graph as the clique graph like below for more easy implementation.



If you encounter difficult even after analyzing the example code, then please raise your question in `piazza`.

How to Build

```
mkdir build
cd build
cmake ..
make
```

```
./qPlacer <defname.def>
```

For example,

```
./qPlacer simple01.def
```

2. Your Own Placement

In this part, you have to implement your own placement.

The algorithm can be from any journal or paper, text book, or your own idea.

If you use another idea from outside of your own, then annotate on the code where the idea is from.

You can use external libraries, only for pure math (ex. `fft`, `matrix solver`, etc). If you want to use an external library, then please let TA know what you will use through `piazza`.

How to Build

```
mkdir build
cd build
cmake ..
make
```

```
./placer <defname.def>
```

For example,

```
./placer simple01.def
```

Code Usages

In the `example.cpp` file, you can see how to use the methods in `Circuit`, `Instance`, `Net`, and `Pin`.

```
void Circuit::howToUse() {
  /*!
   * This function is
   * example for how to use variables in Circuit class
   *
   * You should refer this function and know how to use `Net`, `Instance`, `Pin`, etc variables
   * */

  for (int i = 0; i < 5; ++i) cout << endl;

  // Access all Instances(cells) in the circuit
  cout << "Access all Instances(cells) in the circuit" << endl;
  for (Instance *instance : instance_pointers_) {
    string cell_name = instance->getName();
    cout << cell_name << " ";
  }
  cout << endl << endl;

  // Access the 5th instance for demonstrate how to use the data structures
  cout << "Access 5th cell in the circuit." << endl;
  Instance *instance = instance_pointers_.at(10);
  cout << "Cell name: " << instance->getName() << endl;
```

```
        cout << "Cell height: " << instance->getHeight() << endl;
        cout << "Cell width: " << instance->getWidth() << endl;
        cout << "Cell area: " << instance->getArea() << endl;
        cout << "Cell library name: " << instance->getLibName() << endl;
        cout << "Cell coordinate before set coordinate: " << endl << "  "
             << "x[" << instance->getCoordinate().first << "] "
             << "y[" << instance->getCoordinate().second << "]" << endl;
        // set coordinate of the cell as (100, 200)
        instance->setCoordinate(100, 200);
        cout << "Cell coordinate after set coordinate: " << endl << "  "
             << "x[" << instance->getCoordinate().first << "] "
             << "y[" << instance->getCoordinate().second << "]" << endl;

        // Access pins in that instance
        cout << "Access pins in that instance" << endl;
        for (Pin *pin : instance->getPins()) {
          cout << "\tPin coordinate: "
               << "x[" << pin->getCoordinate().first << "] "
               << "y[" << pin->getCoordinate().second << "]";
          cout << "\t\tCorrespond instance name: " << pin->getInstance()->getName();
          cout << "\t\tSignal Type: " << pin->getSignalType();
          cout << "\t\tPin name:" << pin->getPinName() << endl;
        }


        // Grab any pin for demonstrate how to use the data structures
        // here, I'll grab 2nd pin in the above selected cell
        if (instance->getPins().at(2)) {
          Pin *pin = instance->getPins().at(2);

          // You can access the instance correspond to the pin
          cout << "The name of instance which includes the pin: " << pin->getInstance()->getName() << endl << endl;
          // You can access the net correspond to the pin
          Net *net = pin->getNet();
          cout << "Net name: " << net->getName() << endl;
          // You can access the pins which is included in the net
          for (Pin *connected_pin : net->getConnectedPins()) {
            cout << "\tPin coordinate: "
                 << "x[" << connected_pin->getCoordinate().first << "] "
                 << "y[" << connected_pin->getCoordinate().second << "]";
            if (connected_pin->getInstance())
              cout << "\t\tCorrespond instance name: " << connected_pin->getInstance()->getName();
            cout << "\t\tSignal Type: " << connected_pin->getSignalType();
            cout << "\t\tPin name:" << connected_pin->getPinName() << endl;

          }
        }
      }
```

If you call the function `void Circuit::howToUse()` and read the code with the console message, then you will be able to learn how to use the methods (codes).

Below methods will what you will be likely to use. The explanations are in the header files in `include/` directory.

In `include/circuit/Circuit.h`

- `void Circuit::quadraticPlacement()`
- `void Circuit::myPlacement()`

- `void Circuit::placeExample()`
- `ulong Circuit::getHPWL()`

In `include/dataStructures/Instance.h`

- `string Instance::getName()`
- `string Instance::getLibName()`
- `uint Instance::getWidth()`
- `uint Instance::getHeight()`
- `uint Instance::getArea()`
- `std::vector<Pin *> Instance::getPins()`
- `pair<int, int> Instance::getCoordinate()`
- `void Instance::setCoordinate(int x, int y)`
- `bool Instance::isPlaced()`

In `include/dataStructures/Net.h`

- `string Net::getName()`
- `vector<Pin *> Net::getConnectedPins()`
- `int Net::getWeight()`
- `string Net::getSignalType()`
- `ulong Net::getHPWL()`

In `include/dataStructures/Pin.h`

- `bool isInstancePin()`
- `bool isBlockPin()`
- `Instance *getInstance()`
- `Net *getNet()`
- `string getSignalType()`
- `string getPinName()`
- `pair<int, int> getCoordinate()`

You should read the comments in the header files so that completely understand and use these methods.

If you have any questions, then feel free to post questions in the `piazza`.

Grade Criteria

The total score will be 100 points. The grading part will be divided into two parts; **quadratic placement**, **you own placement**.

All submissions should be graded only if the plagiarism check is passed. If the code is positive for plagiarism by a program (and also examined by TA), then you will get an F grade in this course, not only get zero scores.

1. Quadratic Placement (30pt)

Due: 2022/11/22

This part will be graded by HPWL value.

If the HPWL value is the same as the Quadratic placer implemented by TA, then you will be a full score in this part. (each benchmarks has 6 points)

The allowed errors of HPWL value are accepted ±10%.

If the HPWL value is in the target scope, then TA will examine the code whether the code is for the quadratic place algorithm or not.

If the code is not implemented by the quadratic place algorithm, you will get no score.

For evalutation, build like below,

```
mkdir build
cd build
cmake ..
make
```

And execute the binary file with the def file name and evalute version (0).

```
./evaluator <defname.def> 0
```

For example,

```
./evaluator simple01.def 0
```

Then you can the HPWL value and do basic evaluation.

2. Your Own Placement (70pt)

Due: 2022/12/11

The grading considers `HPWL` value and `cell density` for routability.

When calculating cell density, the evaluator will make a 40x40 grid and get the density value by `the total cell area in the grid / the grid area` . If `the max value of the density value` among for all grids overs 1.0, then the penalty will be applied. Otherwise, the density check will be passed.

The grading will follow the below equation.

$$Score = \begin{cases} \frac{1}{1+\log(\frac{HPWL}{HPWL_{1th}})}, & \text{if } maxDensity \leq 1 \\ \frac{1}{1+\log(\frac{HPWL}{HPWL_{1th}})} \times e^{-(maxDensity-1)}, & \text{otherwise} \end{cases} \quad (3)$$

Here, $HPWL_{1th}$ is the smallest `HPWL` value in the students which is passed by the max density check.

You can know the HPWL value and max density value by executing `evaluator` binary file.

After build like below,

```
mkdir build
cd build
cmake ..
make
```

execute the binary file with the def file name and evalute version (1).

```
./evaluator <defname.def> 1
```

For example,

```
./evaluator simple01.def 1
```

Then you can the HPWL value and the max density value.

Benchmarks

Five benchmarks for scoring will exist; `midium01.def` , `midium02.def` , `midium03.def` , `large01.def` , `large02.def` .

`midium01.def` , `midium02.def` , `large01.def` , will be assigned, but the other two things will be hidden.

Simple benchmarks, `simple01.def` , `simple02.def` , `simple03.def` , `simple04.def` will also be given, but this will not be considered for scoring. These benchmarks will be given only for your debugging.

Total running time will be limited 15min for placing above five benchmarks.

Submission

You should submit your whole code except `build` directory for `cmake` build,
through the class server so that your binary file can be rebuilded by TA.

The very brief report needs to be in `doc` directory.

TA Contact

Please get in touch with me through `Piazza` .

License


```
    Developed by:

        The teaching assistant of EECE667, POSTECH in Korea
```