

eda 报告

马晋

2024 年 12 月 29 日

摘要

这是大作业报告，我实现了一个 eda 算法，成功的对输入的 def 文件进行了时钟树综合。并且给出综合结果。我没有在比赛之前完成，但是如果没有出现剑走偏锋专门优化 `buffer_count` 或 `total_latency` 或 `skew` 的情况，我认为我的算法大概率是可以进决赛的。就我与此问题本班其他队对比的结果，我的算法的 performance 最好：体现在：时间上，我的算法 skew 可以 100% 保证在 45 以内，并且在 $10(\text{cts_problems}) + 3(\text{pre_submit})$ 中只有 1 个超过 40(42)，其余全都在 32 以下。而且我的 `average_delay` 基本是最大那个距离的 delay(但是由于插入了两到三个 buffer, 所以会有 50 到 75 的偏差)。并且我的聚类算法较佳，以至于在 `iterative_cluster` 的时候，能非常近似的到达只考虑 `max_fanout`(不考虑偏离中心的 bound) 的误差，根据我从另一队 eda 展示的结果，我推测最终评测的 case 应该是在这些 case 中评测规模较大的，并且第一个应该是类似于 `pre_submit` 的 case1, 总结：越是规模大的 case 我越存在优势，但是即使规模小的 case 我的 performance 也很强(此时 `buffer_count` 可能稍微大一点点(约 100 200)), 并且我的程序具有极大的稳定性。

目录

1	引言	2
2	创新点	2
3	方法	2
4	cluster	3
4.1	overlap	3
4.2	floorplan	3
4.3	测试事宜	4
5	实验结果	5

1 引言

通过这个 project, 我认识到, 完善和修改不能一蹴而就, 最好分多步, 循序渐进的构造。

2 创新点

project 的创新点如下

1. 分块操作, 这是一个非常重要的方式, 我对比了没有分块与分块的算法, 时间上加速了 10 到 20 倍
2. 参数可配置化, 全都在 `parameter.cpp` 中, 其中包含 FF 聚类的初始数量, 大小, 以及最后生成逻辑过程中对于时间 skew 的忍受度, 等等。我将前面部分和后面部分解耦合, 采用不同的配置, 目前, 我已经在当前情况下, 将配置调整到了一个比较优的条件, 实际上, 还可以进一步优化, 即根据文件规模与密度进行优化。(而且在我的代码的基础上很容易实现)
3. 关于聚类, 实际上, 我的聚类效果已经接近极限, 我的聚类是实现在偏移中心距离不超过 `max_bound` 下的聚类, 经过我的验证, 这个聚类的约束确实是非常完美的, 可以给出数据, 在 `parameter.cpp` 中, 将 `bound` 和 `max_bound` 改成 2000000(那么此时完全是 `fanout` 约束), 我发现需要 3200 个点的在只有 `fanout` 约束下要 3000 个点, 这说明, 我的聚类算法是非常优化的。
4. 我的 `overlap` 算法是基于优先队列加查找的, 实际上, 这是我考察了题目得出的综合结果, 这一定是近乎最优的方案, 一方面, 它搜索的范围很广, 另一方面, 它能找到偏离中心不会太远的点, 不会因为离群点而倒是成为时间优化的瓶颈。所以实际上, 可视化后可以看见生成的图很漂亮, 并且可以看见重合度非常低。根据最终 `cts_checker` 的结果, 我把所有的 `overlap` 优化到了 0.2% 以下
5. 基于我对题目的观察, 我对二次距离进行了最优的优化, 这是题目中非常重要的性质
6. 实际上, 我在算法中尽量保证了稳当性, 使得尽量在一些极端情况下都能运行, 比如, `highlevelcluster` 中, 我实际上是反复的迭代直到终止, 而按照测试样例, 实际上只用迭代一次。
7. 我完成了可视化, 老师可以通过 `output.svg` 查看
8. 我分析了很多程序中的问题, 用 `gdb -g debug`
9. 我已经为老师和学长写好了自动化脚本, 方便老师和学长测试最终结果

3 方法

方法已经在 ppt 中介绍了, 中心思想是利用可以确定的最远的延迟, 在这个基础上, 调整其他的 buffer 贴近这个最远的延迟。然而事实上, 问题远不止这么简单: 怎样

聚类? 怎么减小 overlap? 怎么保持约束? 怎么 debug? 发现原来的程序不够好, 我要做改动, 我该怎么? (尽量等效)

4 cluster

我的聚类用的是循环的迭代聚类, 它会循环自动的迭代找到最小的聚类数量, 一方面在这个过程中 `fan_out` 不会超出范围. 另一方面这个过程中 `bound` 不会超出范围. 此外, 这个算法的聚类结果与无约束下的聚类结果非常相近, 这是因为我在聚类的时候, 我会根据 `fan_out` 的大小, 选择一个合适的 `bound`, 这个 `bound` 是根据我对题目的观察得出的。

4.1 overlap

对于 overlap 我用的优先队列, 经过我的测试, 我的几乎所有 case 的 overlap 都在 %0.2 左右.

4.2 floorplan

我以最新解决的一个问题为例, 可以从中窥出实际上有很多细节问题: 如图所示,

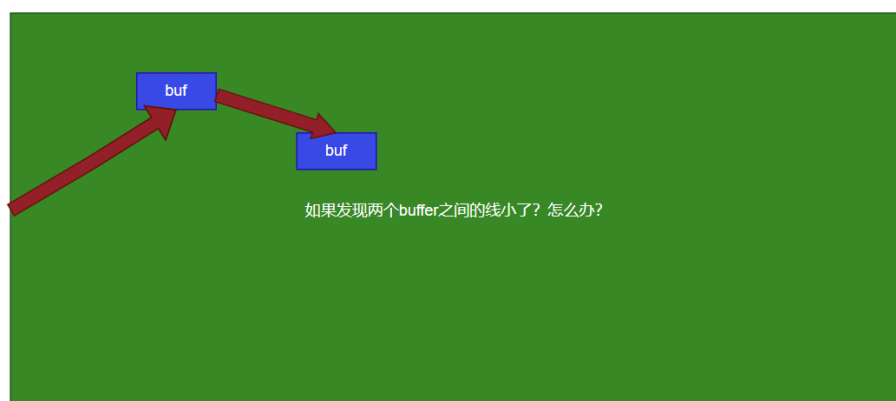


图 1: BUF

这是一个 buffer 的 floorplan, 现在我希望把后面的 buffer 延迟调整到与前面找到的最大值尽量相同从而减少 skew: 如果我沿着两个 buf 的连线, 找到一个 buf 作为中转 (这样算很容易且不用分类讨论), 相连, 则极有可能最后出现这个 buf 的位置超出 floorplan. 但是不这样的话, 为了找到中转 buf (使得最后的 buf 延迟接近于我希望的延迟), 似乎要讨论很多的可能性 (因为是曼哈顿距离, 和绝对值相加, 所以需要有很多的分类讨论), 从而导致问题一下子复杂了很多. 但是, 如果结合实际, 如果 buf 位连线的 latency 小

了, 这个一定是发生在左边, 从而隐式的推断出此时 x 很小的事实, 那么我可以选择找 x 的位置, 此时我预期是不会超过 floorplan 的, 至于 y , 因为我要计算两个 buf 与中转 buf 的曼哈顿距离, 似乎又要分类计算了, 但是发现如果选择 $y = \max(y_{buf_1}, y_{buf_2})$, 那么这个可以回避这个问题, 这样, 通过深层次分析, 避免了分类讨论, 并且在 13 个 case 中都没有出现超出 floorplan 的问题。

4.3 测试事宜

我的算法中有很多可以配置的选项和位置, 这些我已经大致调整到一个比较好的范围了。这些大多是在 parameter.cpp 中。cfg.up 和 cfg.low 不在 parameter.cpp 中确定, 因为它们要根据题目的实际情况分析。cfg.up 和 cfg.low 的算出位置在 generize-topology.cpp 中, 这个是我根据题目的实际情况分析出来的。cfg.low 取为 cfg.up 下的 20ns 的位置目前已知的信息是 clk 的位置是位于 floorplan 的左侧中心, constraint.txt 中 fan_out 和 buf_delay 是会变的, 其它不会变, 但我相信的是, 它们即使变也是在一个合适的范围中, 不会出现太极端的情况。否则我相信大部分算法都失效了。我已经尽力提高了普遍性, 例如, 采用了高层次聚类, 并且在聚类中设置了 level(因为我最终将 FF 的 skew 转化为 highlevelcluster 之后的顶点的 buffer 的 skew, 但是要是聚类顶点的 buffer(对 buffer 进行了聚类, 从而产生的有代表性的 buffer) 到 FF 中经过的 buffer 数量是不同的话, 最终 skew 会在原来基础上 +buffer_delay), 但是我目前尚未观测到 level 不同的情况。应该是因为题目中 buffer 还是相对均匀的。

在加入 level 的过程中, 我用到了等效的思想, 将 level 的作用转化为了 cfg.up 和 cfg.min 的作用, 从而减少的代码所需要做的更改。

似乎我的 max_fanout 会超, 但这个问题不大, 规定是只要超了就扣 1 分, 但是不管怎么超都没事。测试:

./sh 是编译成可执行文件

./test.sh 是逐个测试案例

./check.sh 是根据逐个测试的结果, 用 cts_checker 检测效果

在产生 generated 之后, 可以运行 python resvisualize.py 观察结果。

请务必保证先运行 ./test.sh 再运行 ./check.sh, 否则产生 svg 会出问题。

每次修改 main.cpp 后都需要重新运行 ./sh

注意事项: 我发现单次测试结果可能有随机性, 劳烦助教可以用不同的平台测试几次。当然即使是随机性也是比较小的

5 实验结果

实验的结果分为两部分，都在 generated 文件夹中一个是 svg 文件，这是最终生成的图，FF 是蓝色，BUF 是红色。

另一个是 cts_checker 给出的 report.txt。

此外，由于测量结果具有不确定性，我这边生成了一个 refgenerated 文件夹，里面是我自己的测试结果，可以作为参考。如果老师和助教发现结果有很大偏差，可以联系我 vx:imagine。