

轨道分配工具LSAssigner说明文档

工具使用说明

代码结构

```
1  eda_api/
2  |
3  |—— ls_assigner/                # eda求解器相关代码
4  |  |
5  |  |—— NuPBO/                  # pbo求解器
6  |  |  |—— cmdline.h            # 命令行参数解析头文件
7  |  |  |—— heuristic.cpp        # 启发式算法实现文件
8  |  |  |—— heuristic.h          # 启发式算法头文件
9  |  |  |—— init.cpp             # 初始化函数实现文件
10 |  |  |—— init.h                # 初始化函数头文件
11 |  |  |—— MaxSATFormula.cpp     # 最大可满足性问题公式实现文件
12 |  |  |—— MaxSATFormula.h       # 最大可满足性问题公式头文件
13 |  |  |—— my_class.cpp          # 自定义类实现文件
14 |  |  |—— my_class.h            # 自定义类头文件
15 |  |  |—— nupbo.h               # 求解器主头文件
16 |  |  |—— nuPBOFunction.cpp     # 求解器接口实现文件
17 |  |  |—— parse_arguments.cpp   # 解析命令行参数实现文件
18 |  |  |—— parse_arguments.h     # 解析命令行参数头文件
19 |  |  |—— settings.cpp          # 设置参数实现文件
20 |  |  |—— settings.h            # 设置参数头文件
21 |  |  |—— string_util.cpp       # 字符串工具函数实现文件
22 |  |  |—— string_util.h         # 字符串工具函数头文件
23 |  |  |—— struct_from_basis.h   # 从基础结构生成的结构头文件
24 |  |  |—— basis_pms.cpp         # 基础 PMS 实现文件
25 |  |  |—— basis_pms.h          # 基础 PMS 头文件
26 |  |
27 |  |—— buildmodel/              # 建模相关代码
28 |  |  |—— model.cpp             # 模型实现文件
29 |  |  |—— model.h               # 模型头文件
30 |  |  |—— struct.h              # 部分结构体头文件
31 |  |
32 |  |—— LSAssigner.cpp           # LocalSearchAssigner 实现文件
33 |  |  |—— LSAssigner.h          # LocalSearchAssigner 头文件
34 |  |  |—— ids.hpp               # 部分结构体头文件
35 |
36 |—— test.cpp                    # 测试文件
```

```
37 |— test.h          # 测试头文件
38 |— Makefile       # 编译文件
39 |— main.cpp       # 主函数代码
```

API调用说明

- LSAssigner中是LocalSearchAssigner动态库的源代码，Makefile是动态库编译命令。
- test是调用api的示例代码。test_panel函数调用parseInputFile函数读入文件，调用ls_a.GetResult函数进行轨道分配问题的求解并输出结果文件。
- 运行示例

```
1 make      # 编译生成llsassigner动态库
2 g++ -fPIC -g -O3 -std=c++17 -fopenmp -c test.cpp -o test.o
3 g++ -fopenmp -m64 -g -O3 -std=c++17 -o TA main.cpp test.o -L. -llsassigner
  # 编译测试文件并链接到动态库，生成可执行文件
4 ./TA ispd18_test5_metal5.input.def.ta.txt >out.txt      # 运行可执行文件
```


输入输出说明

输入文件

- txt文件
- 格式如下

```
panel [layer_id] [panel_id] [lb_x] [lb_y] [rt_x] [rt_y] [prefer_direction]
{
track_list
[axis] [start] [step_length] [end]
wire_list
[net_id] [lb_x] [lb_y] [rt_x] [rt_y]
soft_shape_list
[net_id] [lb_x] [lb_y] [rt_x] [rt_y]
hard_shape_list
[net_id] [lb_x] [lb_y] [rt_x] [rt_y]
}
```
- 示例

```
panel 0 0 0 0 3200 1840000 V
{
track_list
X 100 200 3100
Y 100 200 1839900
wire_list
3 0 902650 50 905750
13 0 601450 50 602750
58 0 867050 50 872750
73 0 603450 50 604950
88 0 870250 50 871550
131 0 617050 50 617750
171 0 609450 50 611350
173 0 972650 50 972950
224 0 863850 50 866750
235 0 911850 50 912350
249 0 899050 50 900350
302 0 939850 50 940950
305 0 891450 50 892950
344 0 879850 50 880150
374 0 610250 50 611350
383 0 1233050 50 1235350
497 0 926650 50 928150
531 0 937250 50 940950
558 0 1235050 50 1235350
```

 test.txt

输入参数

参数示例	参数含义	是否必须添加	默认值（只有非必须参数有）	备注
./TA	./二进制文件	是		
ispd18_test5_metal5.inp ut.def.ta.txt	要处理的文件的路径	是		
-timelimit 60	单个panel建模时间限制（单位为秒）	否	60s	注意中间的空格
-sol_time 20	单个panel对应的模型求解时间限制（单位为秒）	否	10	
-output_file "home/output.txt"	结果输出文件，输出各panel中wire的坐标	否	和可执行文件同目录下的output_file.txt	
-value_file "home/value.txt"	目标值输出文件，输出模型中各	否	和可执行文件同目录下的value_file.txt	

	panel目标值和总目标值			
-blockcoef 300	blockcoef参数	否	500	
-pincoef 2	pincoef参数	否	2	
-routecoef 1	routecoef参数	否	1	
-distcoef 0.1	distcoef参数	否	0.1	
-nthreads 128	并行的线程数	否	1	
-solve_option 1	求解方式，暂时只能为1	否	1	
-testall 2	建模第几个panel(下标从0开始)	否	-1(表示全部panel)	

输出文件

- txt文件
- 格式如下

```
panel [layer_id] [panel_id] [lb_x] [lb_y] [rt_x] [rt_y] [prefer_direction]
{
wire_list
[net_id] [lb_x] [lb_y] [rt_x] [rt_y]
}
```

加入iEDA的测试说明

- 测试方法：将LSAssigner的API合入iEDA代码中，分别使用LSAssigner和iEDA的轨道分配算法运行并调用同一个测试函数，输出wire与wire重叠次数，wire与pin重叠次数和wire与block重叠次数。
- 测试结果

样例	panels数量	LSAssigner				原有iEDA3
		total_cost_rr	total_cost_rp	total_cost_rb	total cost	total_cost
		(wire与wire重叠次数)	(wire与pin重叠次数)	(wire与block重叠次数)	总重叠次数	(wire与v重叠次数)
ispd18_test1	384	9	68	0	77	
ispd18_test3	1829	189	955	5	1149	

