Lin–Kernighan AN

I. 简单的用数学语言描述 power switch 的连接问题

设二维平面上存在一个无向图 G=(V,E), 其中 V 为所有顶点(vertex)的集合,E 为所有边(edge)的集合, 现要找到一条路径(path) P = {e | e ϵ E} 遍历(visit)每个顶点且只遍历一次(如果将该路径的头尾顶点也相连则形成一个环 cycle)

解的存在性: 构造一个完备图(complete graph),则 P 必定存在

优质的解: 我们的需求是路径 P 上任意连续的两个顶点之间距离不能太长(满足 transition 的要求)

这个问题算法上属于 NP 问题,时间复杂度是指数级的,我们只能得到近似的解.

Ⅱ. 旅行售货员问题---Travelling salesman problem

旅行售货员问题与 power switch 的连接问题相似,只不过它要求 P 总长最短,P 上任意连续两点之间的距离无要求. 但 P 的总长等于 P 上所有边也即两个顶点之间距离之和,所以 P 总长要短,则 P 上任意连续两个顶点之间的距离也要短-------那么这又恰好回到了我们的实际需求上.

旅行售货员问题离散数学上也讲烂了,网上现成的代码也很多,我们就直接选个好的拿来用就行了这里我们选得是 Lin-Kernighan 启发式(heuristic)算法,该算法能够非常快速的找到比较满意的近似解.

Ⅲ. linkern 程序的使用

Base script 在这 /ux/WXSQ88/user/minggpeng/cpu_top/51_apr/work_icc_FNL1p0/scripts/create_power_switch_chain.tcl 只算出Ⅰ中的 Path，方便使用者扩展

下面的例子由简单到复杂,计算结果从粗糙到精细,当然限制太多可能无解,直接运行 linkern 不带参数可显示帮助

1. linkern -d -1 -h -1 -o ***graph.sol graph.tsp***

   解释：这个是原始的 linkern 算法(未经我改过的) –d 表示你对任意连续两点之间的距离的需求 -1(负一)表示没有要求; -h 限定 P 的总长, -1(负一))表示取消限定; -o 结果输出文件;graph.tsp 输入文件，原则上选项顺序无关的，但原代码里要求 graph.tsp 输入文件放在最后

   优点：原始的 linkern 算法不管你的图是什么样子，它都能迅速的得出结果(当然可能有 1k,2k 的这种长线啦，但条数一般很少)，这就非常适合 try run 阶段，不折腾

2. linkern –d ***max_distance*** -o ***graph.sol graph.tsp***

   很简单啦，说出你的需求(任意连续两点之间的最大距离) 限制的太紧可能无解陷入死循环哦,影响你跑 batch job,算不出来你就加大 *max_distance 吧*

3. linkern –d ***max_distance*** -m ***num*** -o ***graph.sol graph.tsp***

   如果你不喜欢 1k 这种长长线，偏爱 500um 这种’短’长线(优点：在 driver 端插一个 buffer 就能搞定 transition 问题), 那你就用-m 选项(允许多少条长线，默认值是 1 条，1 条的话限制太紧可能得不到结果)

   num 的设定依据: 你先用 2 的方式跑一下看看刷屏的时候 Best: 后面的数字趋近多少， 然后再除以 50w(我内部设的 P 默认总长)，向上取整得到 *num*

References:

1. Concorde: http://www.math.uwaterloo.ca/tsp/concorde.html

2015/12/21