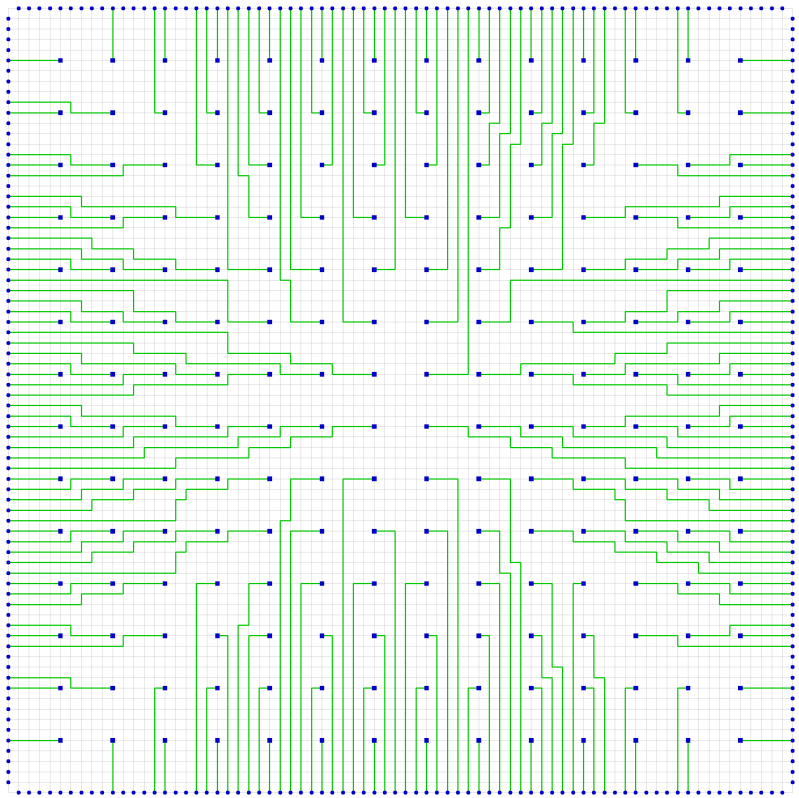
Rule-Based Regular Routing Method

周聿浩　翁家翌

June 6, 2017

The Problem

在一个电路板上，给定一个均匀分布的 $n \times n$ 个内部节点，要求确定电路板的大小，计算各个节点到电路板边界的路径，使得各个路径不相交并且长度之和最短。



The Problem

为了解决这个问题，我们实现了基于费用流的布线方案，该方案能够获得最优解，然而由于不适应与大规模的数据，因此又实现了基于规则的布线方案，该方案可以在可以接受的时间内得到较优的解。

我们支持将计算得到的方案以图片形式输出到文件或窗口，同时也支持以原始路径的形式保存到文件。并且支持从文件中读取原始数据并且显示到窗口。

Some Details

- **main.cpp** 主要实现了和用户交互的逻辑
- **vertex.h** 主要实现了基本的节点类型 Vertex
- **path.h/cpp** 主要实现了基本的路径类型 Path
- **route.h/cpp** 整个路径规划算法的抽象接口 Route，同时定义了一些辅助函数用于输出等。
  - 具体的几个实现。
- **visualization.h/cpp** 用于可视化算法输出

Some Details

- **main.cpp** 主要实现了和用户交互的逻辑
- **vertex.h** 主要实现了基本的节点类型 Vertex
- **path.h/cpp** 主要实现了基本的路径类型 Path
- **route.h/cpp** 整个路径规划算法的抽象接口 Route，同时定义了一些辅助函数用于输出等。
  - 具体的几个实现。
- **visualization.h/cpp** 用于可视化算法输出

Route 的具体实现都不对用户可见，利用一个工厂函数来创建其实例同时转换为基类的指针。为了减少用户手动删除造成的不必要错误，使用 std::shared_ptr 来管理内存。

另外，对于读取已经存在的路径数据的类，利用了适配器模式将读取结果保存在 Route 里。

Algo - MCMF

题目可以抽象为给定一张图，寻找一些点不相交路径，使得总长度最短。

Algo - MCMF

题目可以抽象为给定一张图，寻找一些点不相交路径，使得总长度最短。

每个点只能经过一次？拆点！

Algo - MCMF

题目可以抽象为给定一张图，寻找一些点不相交路径，使得总长度最短。

每个点只能经过一次？拆点！

对于每一个节点，由于可以向其相邻节点前进，就让该节点向四周连边。

这样，计算最小费用最大流之后就可以得到最优方案了。

Algo - MCMF

题目可以抽象为给定一张图，寻找一些点不相交路径，使得总长度最短。

每个点只能经过一次？拆点！

对于每一个节点，由于可以向其相邻节点前进，就让该节点向四周连边。

这样，计算最小费用最大流之后就可以得到最优方案了。

此外还有一个关于最小电路板大小的问题，可以利用二分答案来进行计算。

这个计算过程中只需要判断合法性而无需最优性，我们可以利用最大流算法而忽略费用以提高速度。

Algo - Rule

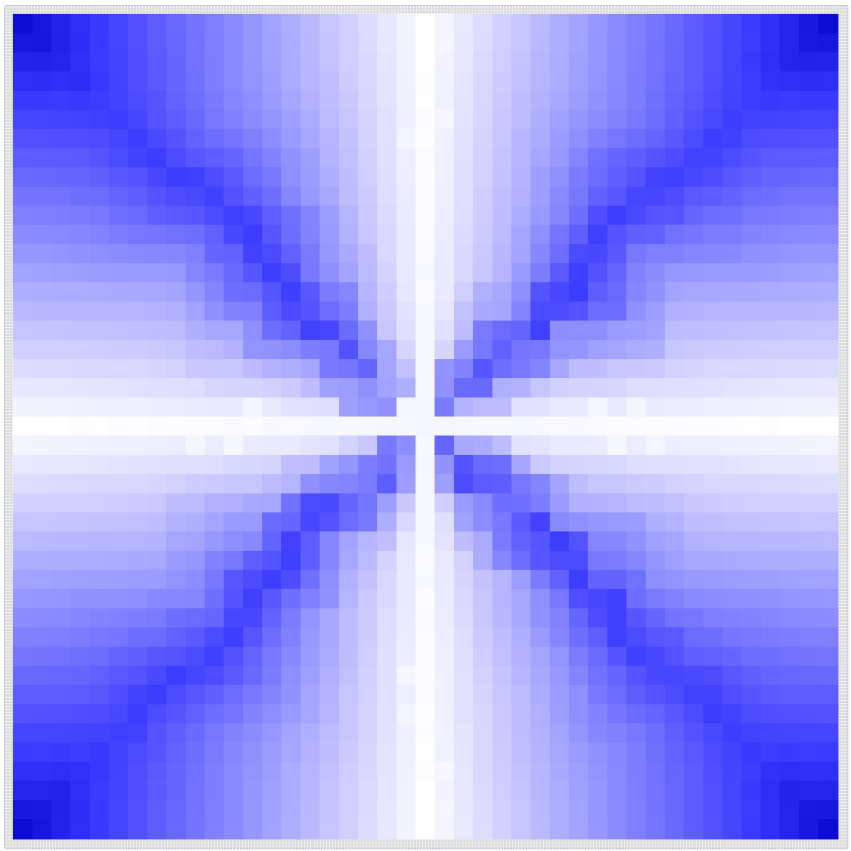


Figure: 将费用流得到的布线方法可视化之后……喵喵喵？

Algo - Rule

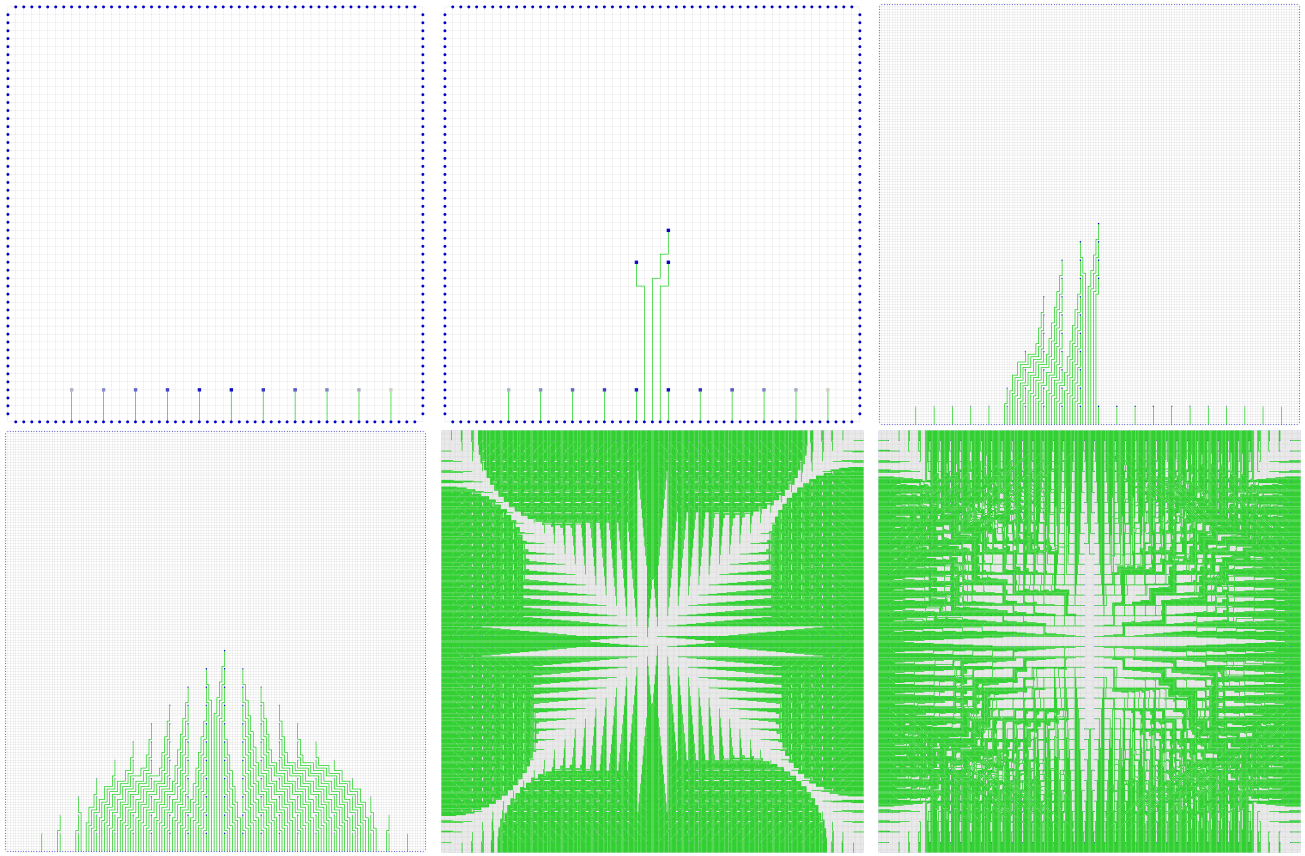


Figure: 图解基于规则的布线方法

Algo - Rule

在实现过程中，考虑到 80×80 的数据，网格图有 1945×1945 的大小，因此寻找路径的算法成为了效率实现的瓶颈。本人使用了 A^* 算法替代传统的 BFS 算法，估价函数设计成尽量贴着已存在的路径进行寻路，从而实现了效率正比于路径长度的高效算法。

经检验，该算法在 $1 \sim 80$ 的数据中只有两个数据需要一些微调，其他数据均快速出解。

Performance

理论上分析，对于 n 个点，m 条边并且容量都为 1 的 *MCMF*，时间复杂度为 $O(nm^2)$，而 *Dinic* 的时间复杂度为 $O(\min(n^{2/3}, m^{1/2}) \cdot m)$。在这个问题中，记 n^2 为需要连接的点的个数，$n \in [1, 80]$，N 为电路板的边长，结果表明 $\frac{N-1}{n+1} \in [\frac{n}{4}, \frac{n}{3}]$，因此可以知道 $N = \Theta(n^2)$。

因此复杂度可估计为 *MCMF*：$O(n^6)$，*Dinic*：$O(n^3)$。

而人工设计的基于规则布线的方法几乎是正比于布线的总长度，因此时间复杂度为 $O(n^2)$。

Performance

理论上分析，对于 n 个点，m 条边并且容量都为 1 的 *MCMF*，时间复杂度为 $O(nm^2)$，而 *Dinic* 的时间复杂度为 $O(\min(n^{2/3}, m^{1/2}) \cdot m)$。在这个问题中，记 n^2 为需要连接的点的个数，$n \in [1, 80]$，N 为电路板的边长，结果表明 $\frac{N-1}{n+1} \in [\frac{n}{4}, \frac{n}{3}]$，因此可以知道 $N = \Theta(n^2)$。

因此复杂度可估计为 *MCMF*：$O(n^6)$，*Dinic*：$O(n^3)$。

而人工设计的基于规则布线的方法几乎是正比于布线的总长度，因此时间复杂度为 $O(n^2)$。

演示一下

Performance

n	*MCMF*	*Dinic*	*Rule*
5	0.040	0.040	0.040
15	0.308	0.056	0.040
25	7.504	0.404	0.064
35	3’25”	2.276	0.148
45	24’4”	14.656	0.272
55	数小时	66.584	0.484
65	数小时	7’13”	0.904
75	约 $8 \sim 9h$	约 $1h$	1.712

Table: 性能测试，时间若未注明则以秒计

Performance

n	*MCMF*	*Dinic*	*Rule*
5	0.040	0.040	0.040
15	0.308	0.056	0.040
25	7.504	0.404	0.064
35	3’25”	2.276	0.148
45	24’4”	14.656	0.272
55	数小时	66.584	0.484
65	数小时	7’13”	0.904
75	约 $8 \sim 9h$	约 $1h$	1.712

Table: 性能测试，时间若未注明则以秒计

听说消圈算法巨慢无比？

Performance

n	$MCMF$	$Dinic$	$Rule$	$\frac{Rule-MCMF}{MCMF}\%$	$\frac{Dinic-MCMF}{MCMF}\%$
5	79	79	79	0.000	0.000
15	3862	3910	3879	0.440	1.243
25	27394	27566	27460	0.241	0.628
35	101775	102051	101920	0.142	0.271
45	273183	273811	273433	0.092	0.230
55	602856	603975	603241	0.064	0.186
65	1167116	1168932	1167662	0.047	0.156
75	2057345	2060192	2058082	0.036	0.138

Table: 效果分析

Thank you!