有限电阻网络的计算机优化求解

PB20000296 郑滕飞

摘要：基尔霍夫方程组的完备性保证了由此出发可解出任意有限电阻网络，但直接对所有边构造方程会导致复杂度急剧上升，由此需要从不同层面进行优化。本文实现了基于基尔霍夫定律的有限电阻网络求解，并在此基础上进行优化，大幅降低了复杂度。利用此求解法，亦可实现对含源电路的模拟。

一、传统方法的计算机实现及其局限

传统的有限电阻网络求解方法分为三步：读取有限电阻网络并建成图、设出各支路电流后得到方程组（其中又按两定律分为不同方程）、求解得出的线性方程组。此处先分别将三步实现，再进行整合。

1. 电阻网络转化成图结构存储

对有限电阻网络，实质可以看作由有限个点、有限条边构成的图，任意两个点之间可能连有度数非负的边，代表两点之间的阻值。不过，所有的点并非相同，若我们需要求两个点之间的电阻，此两点不能进行任意操作，也即“固定”的结点。不失一般性，我们假设v_0, v_1两点固定，也即计算此两点间的电阻。

读入的文件中，第一行代表结点个数与边的条数，此后的每行有三个数a, b, r，代表点v_a, v_b间连有了阻值为r的电阻。例如，下方的图 1 转化成的输入为（分号表示换行，假设$ABCDEF$分别为$v_0, v_1, v_2, v_3, v_4, v_5$）：

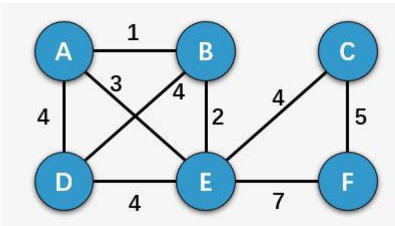

图 1 网络样例

6 9; 1 0 1; 0 3 4; 2 4 4; 1 4 2; 0 4 3; 3 4 4; 4 5 7; 1 3 4; 5 2 5

为之后实现优化方便，选择利用邻接矩阵储存，也即，利用$v \times v$（v为顶点数）矩阵存储两点之间所连电阻。储存效果如图 2。

#	1.0	#	4.0	3.0	#
1.0	#	#	4.0	2.0	#
#	#	#	#	4.0	5.0
4.0	4.0	#	#	4.0	#
3.0	2.0	4.0	4.0	#	7.0
#	#	5.0	#	7.0	#

图 2 储存效果示意图（#表示不相连）

在这种储存方式中，每条边实际被储存了两次（如(v_0, v_4)与(v_4, v_0)实际是同一条边），但好处在于，计算结点的度数（即与其相连的结点数时），可以直接通过本行不为#的个数得到结果。

2. 基尔霍夫第一方程

为求解电阻网络，我们假设在v_0, v_1间加有$1V$的电压，并由此计算各支路的电流。我们先对边进行初始化：假设每条边上都有电流，并且电流始终从编号较小的顶点流向编号较大的顶点。

```
(0,1) (0,3) (0,4) (1,3) (1,4) (2,4) (2,5) (3,4) (4,5)
```

图 3 边集（即为对应需求电流集）示意图

我们采用$n \times (n+1)$的矩阵来存储线性方程组，每一行的前n个分量代表第n个电流的系数，最后一个分量代表常数项。

对除了v_0, v_1外的每个点，都有总流入电流等于总流出电流，一共有$v-2$个这样的方程。而这些方程的常数项均为 0。

```
0A  0B  0C  0D  0E  1F  1G  0H  0I=  0
0A -1B  0C -1D  0E  0F  0G  1H  0I=  0
0A  0B -1C  0D -1E -1F  0G -1H  1I=  0
0A  0B  0C  0D  0E  0F -1G  0H -1I=  0
```

图 4 基尔霍夫第一方程示意图（加号省略，不同字母代表不同电流）

3. 独立支路的选取

为了构造合适的第二方程，需要寻找足够独立的回路。此处，先选出一些边构造电阻网络的一棵生成树，在生成树上每加一条不在其中的边，则形成一个独立回路，由于生成树所需要的边数为$v-1$，这样的独立回路数为$e-v+1$（e为边数）。每个独立回路中，电势的代数和为 0，即电流与电阻乘积（考虑方向）的代数和为 0。

此段的程序实现较为复杂，大致需要以下几部分：首先，通过从边集中每次选取不会成环的边添加入选择的边中，直到添加了$v-1$个，形成生成树。其次，对每条不在生成树中的边，将其加入生成树，然后每次清理度为 1 的顶点，直到只剩下含此条边的环。最后，从加入环的边开始，依次循环一圈，确定每边的对应分量的正负。

```
1A -4B  0C  4D  0E  0F  0G  0H  0I=  0
1A  0B -3C  0D  2E  0F  0G  0H  0I=  0
0A  4B -3C  0D  0E  0F  0G  4H  0I=  0
0A  0B  0C  0D  0E  4F -5G  0H  7I=  0
```

图 5 基尔霍夫第二方程示意图

4. 方程组的最终构建

至此，我们已经拥有了$e-1$个方程，而由于电流的分量共有e个，还需要最后一个方程，也即找一条从v_0到v_1的通路，通路上的总电压降为 1。

利用类似上一部分的思想（相当于将边(v_0, v_1)加入生成树后构造环，再删去这条边即可得到最终结果），可以找到这条通路，并由此构造最后一个方程。

在得到了所有方程后，我们既可以写出完整的基尔霍夫线性方程组。方程与变量个数为e，代表每条边通过的，假设从负向留至正向的电流。

```
 0A  0B  0C  0D  0E  1F  1G  0H  0I=  0
 0A -1B  0C -1D  0E  0F  0G  1H  0I=  0
 0A  0B -1C  0D -1E -1F  0G -1H  1I=  0
 0A  0B  0C  0D  0E  0F -1G  0H -1I=  0
 1A -4B  0C  4D  0E  0F  0G  0H  0I=  0
 1A  0B -3C  0D  2E  0F  0G  0H  0I=  0
 0A  4B -3C  0D  0E  0F  0G  4H  0I=  0
 0A  0B  0C  0D  0E  4F -5G  0H  7I=  0
 1A  0B  0C  0D  0E  0F  0G  0H  0I=  1
```

图 6 完整的基尔霍夫方程组

为验证程序正确性，笔者还使用了另一组输入，代表非平衡电桥的输入：

```
4 5; 0 2 1; 0 3 2; 1 2 3; 1 3 4; 2 3 5
```

此输入得到了如下的基尔霍夫方程组：

```
-1A  0B -1C  0D  1E=  0
 0A -1B  0C -1D -1E=  0
 1A -2B -3C  4D  0E=  0
 1A -2B  0C  0D  5E=  0
 1A  0B -3C  0D  0E=  1
```

图 7 电桥的基尔霍夫方程组

经过验证，两方程组均独立，由此基本验证了程序的正确性。

5. 方程组的求解与局限性

对线性方程组的求解，有大量现成的程序，如直接利用 Mathematica 软件即可得到解。将解中所有从v_0流出的电流（由于默认方向为自小向大，过v_0的电流必为流出）相加，即可得到总电流，利用$R=\frac{U}{I}$，由于已设电压为$1V$，直接取倒数即可得到最终结果。

```
In[18]:= LinearSolve[{{-1, 0, -1, 0, 1}, {0, -1, 0, -1, -1}, {1, -2, -3, 4, 0}, {1, -2, 0, 0, 5},
         线性求解
            {1, 0, -3, 0, 0}}, {0, 0, 0, 0, 1}];
```

In[19]:= $\left\{\frac{22}{85}, \frac{27}{170}, -\frac{21}{85}, -\frac{29}{170}, \frac{1}{85}\right\}$

图 8 Mathematica 所得的非平衡电桥电路最终结果

此外，笔者还测试了代表平衡电桥的输入：

```
4 5; 0 2 1; 0 3 2; 1 2 3; 1 3 6; 2 3 5
```

得到最终结果为：

```
In[1]:= LinearSolve[{{-1, 0, -1, 0, 1}, {0, -1, 0, -1, -1}, {1, -2, -3, 6, 0}, {1, -2, 0, 0, 5},
        线性求解
           {1, 0, -3, 0, 0}}, {0, 0, 0, 0, 1}];
```

In[2]:= $\left\{\frac{1}{4}, \frac{1}{8}, -\frac{1}{4}, -\frac{1}{8}, 0\right\}$

图 9 Mathematica 所得的平衡电桥电路最终结果

可以发现，电桥支路电流为 0，符合现实，进一步验证程序正确性。

此方法虽然必然能获得最终结果，但有一定的局限性，原因在于，一般线性方程组的求

解需要采取高斯消元法，n阶线性方程组求解是一个复杂度较高的事，结合程序中其他部分的复杂度，当电阻网络增大时复杂程度显著提升。为此，需要寻求优化的策略。

二、化简电阻网络

1. 基本思路

在我们日常进行电阻网络的计算时，对于简单串并联电路，都是可以直接进行计算的。具体的计算方法为：将串联或并联的部分合并，然后重新代入电路计算。这启示我们，如果电阻网络中出现了某种“特定结构”，我们可以优先处理这个结构，再对余下的进行处理。值得注意的是，正如最开始所说，所求两点不能进行任意操作，因此，下文中的“结点”均是指除了起点、终点外的点。

2. 三种情形的处理

首先需要考虑的，是一般的电阻网络不会出现，但在化简过程中可能得到的退化情况。这样的情况分为两种：若某个结点的度数为 1，也即只有一条边连向网络，则可以直接删去这条边，而不影响结果。若某个结点有一条连向自己的边（也即自环），亦可直接删去此边而不影响结果。

其次，需要考虑简单串并联电路。在原始情况中，由于初始数据的限制，不会出现简单并联。而只要某个结点的度为 2，即可知出现了简单串联的情况。此时，可删去此结点，将它所连的两结点替换为两段阻值之和。值得注意的是，若这两个结点原本就连有电阻，则不可直接替换，而是要看作简单并联计算并联电阻。

以上的两种化简虽然能起到一些作用，但现实中大部分的电路是比这两者复杂的。以电桥电路为例，就无法以此进行化简。为此，笔者想到了教材上出现过的习题，利用三角-Y 变换，可以将度为 3 的结点一并化简。

利用变换公式$Y = R_1R_2 + R_2R_3 + R_3R_1, R_{12} = \frac{Y}{R_3}, R_{31} = \frac{Y}{R_2}, R_{23} = \frac{Y}{R_1}$,可将度为三的结点删去，将与之相连的结点间的电阻重构（若原本不连则直接替换，连则视为简单并联），达到更好的化简作用（由于删去的 3 条边 1 个顶点，增添的边数不超过 3，此必为化简）。

3. 化简终止与重构网络

当三种化简都不可进行时，化简即终止。由三种情形的定义可以发现，若化简后并不为最简情况，必然每个结点度大于等于 4，这样的结构在电路中并不多见。事实上，无论是测试样例或电桥电路，均可在化简后直接变成单个电阻，从而直接得到结果：

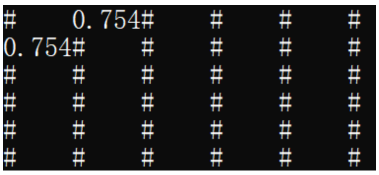

图 10 网络样例的化简结果

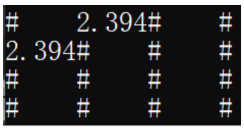

图 11 非平衡电桥化简结果

```
#     2.667#     #
2.667#     #     #
#     #     #     #
#     #     #     #
```

图 12 平衡电桥化简结果

当然，一般的电路未必有如此好的性质，因此需要在化简后重构电路。例如以下输入：

6 12; 0 1 2; 0 2 2; 0 3 2; 0 4 2; 1 2 2; 1 3 2; 1 4 2; 2 3 2; 2 4 2; 3 4 2; 3 5 2; 4 5 2

代表如下图所示的带有星形的网络：

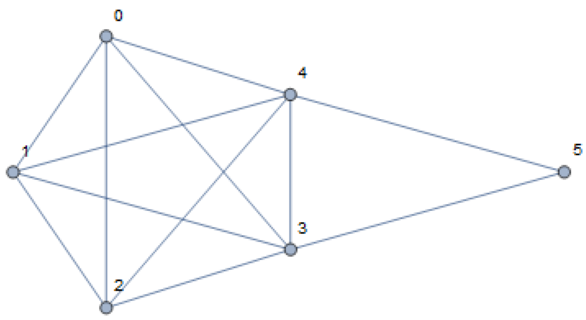

图 13 星形网络示意图

容易看出，此图经过化简，v_5将被删去，但由于其他点还存在，需要重构网络。通过找出所有度不为 0 的顶点即可重构网络，下图为重构后的基尔霍夫方程组，可发现只剩下 10 条边，比起原边数有减少。

```
0.00A -1.00B 0.00C 0.00D -1.00E 0.00F 0.00G 1.00H 1.00I 0.00J=0.00
0.00A 0.00B -1.00C 0.00D 0.00E -1.00F 0.00G -1.00H 0.00I 1.00J=0.00
0.00A 0.00B 0.00C -1.00D 0.00E 0.00F -1.00G 0.00H -1.00I -1.00J=0.00
2.00A -2.00B 0.00C 0.00D 2.00E 0.00F 0.00G 0.00H 0.00I 0.00J=0.00
2.00A 0.00B -2.00C 0.00D 0.00E 2.00F 0.00G 0.00H 0.00I 0.00J=0.00
2.00A 0.00B 0.00C -2.00D 0.00E 0.00F 2.00G 0.00H 0.00I 0.00J=0.00
0.00A 2.00B -2.00C 0.00D 0.00E 0.00F 0.00G 2.00H 0.00I 0.00J=0.00
0.00A 2.00B 0.00C -2.00D 0.00E 0.00F 0.00G 0.00H 2.00I 0.00J=0.00
0.00A 0.00B 2.00C -2.00D 0.00E 0.00F 0.00G 0.00H 0.00I 1.33J=0.00
2.00A 0.00B 0.00C 0.00D 0.00E 0.00F 0.00G 0.00H 0.00I 0.00J=1.00
```

图 14 重构后的星形网络基尔霍夫方程组

三、结果的拓展方向

1. 算法优化

在构建基尔霍夫方程组时，有些算法是较为复杂的，例如此处构造生成树采取了方便从边集直接出发的克鲁斯卡尔方法，若利用深度/广度遍历进行构造，时间复杂度会更小。对其他算法，或许亦有更优化的算法可以得出结果。

2. 更多化简方式的可能

所用的化简方式对度为 3 以内的结点已经基本完备了，但对度数更大的结点仍然难以简化。笔者有想过推广三角-Y 变换至度更多的情形，但此时结点减少反而会导致边数增加（利用三角-Y 变换类似的思路可解出，对于度为n的结点，若想保证去除，需要新增$\frac{n(n-1)}{2}$条边），因此此思路无法进行。不过，这也引申出两种可能——第一，虽然三角-Y 变换去除

了多余的结点，但反向的 Y-三角变换或许也存在意义。在边不变的情况下增加结点数，会导致“好列”的第一方程增加，“难列”的第二方程减少。由此，在结点数以无法再减少的情况下，反向变换可能带来转机。第二，对于度更多的情况，可能也可以找到其中的基本结构而简化电路。

虽然如此，在普通电路网络中，往往会出现大量度数不高的结点，因此针对低度数结点的化简已经具有了重要意义。

3. 模拟含源电路

若网络的每条边上不止存在电阻，还存在电源，则后两类方程组需要进行一定的修正，在计算电势降低时将电源加入环路，即可得到模拟的电路情况。但是，此时又将面临由于方程组过大导致的复杂度提升。

注意到，若只有部分网络中存在电源，则不与含源边连线的结点依然可以做化简操作，由此引申出新的化简方式：

若要计算通过某个电源的电流，利用独立性原理，只需考虑每个电源单独存在时通过此支路的电流。由此，每次计算时最多只有四个结点（支路的两端点、电源的两端点）被固定，当结点数较大时，能起到的化简作用就不小了。

参考文献：

[1] 严蔚敏、吴伟民. 数据结构(C 语言版). 清华大学出版社, 2012.
[2] 叶邦角. 电磁学(第 2 版). 中国科学技术大学出版社, 2018.

辅助材料列表：

Resistance.cpp 电阻网络求解全程序（通过改变主函数中文件名可测试不同样例）
sample.txt 网络样例
bridge.txt 非平衡电桥
bridge2.txt 平衡电桥
star.txt 星形网络