文章编号:1671-4598(2020)05-0233-04 DOI:10.16526/j.cnki.11—4762/tp.2020.05.050 中图分类号:TP391 文献标识码:A

基于 RRT * 的母线布线路径规划算法

刘 冲,周 驰
(华南理工大学 机械与汽车工程学院,广州 510641)

摘要:针对母线布线设计繁杂,低效,耗时成本高的问题;对工程中母线布线设计的约束与优化目标进行了研究总结,提出了一种基于快速扩展随机树算法(RRT*)的母线布线路径规划算法;在传统的 RRT* 算法的基础上,通过引入中间点(corner 点)的方式改变已生成路径到随机点的扩展方式,使生成路径符合母线的走向限制,实现了初始路径的生成;同时在初始路径生成过程中采取贪心的优化策略,获得弯头数量最少且满足约束的路径;仿真结果表明,相较于传统的 RRT* 路径规划算法,文章提出的算法可以很好的满足母线的各项布线要求,为母线的自动布线问题提供了一个新方法。

关键词:母线;自动布线;RRT*;路径规划;布局设计

Busbar Path Planning Based on RRT * Algorithm

Liu Chong, Zhou Chi
(School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou 510641, China)

Abstract: The design of the busbar wiring is complicated, inefficient, and time-consuming. The research summarizes the constraints and optimization objectives of busbar wiring design in engineering. A busbar routing path planning algorithm based on rapidly-exploring random trees algorithm (RRT*) is proposed. To refine the initial path, on the basis of the traditional RRT* algorithm, the extension method of the generated path to the random point is changed by introducing the intermediate point (corner point), the generated path satisfy the tendency of the busbar. At the same time, a greedy optimization strategy is adopted in the path generation process to obtain a path that uses the least amount of elbows and satisfies the constraints. The simulation results show that compared with the traditional RRT* path planning algorithm, the proposed algorithm can meet the busbar wiring requirements well and provide a new method for the busbar automatic routing problem.

Keywords: busbar; automatic routing; RRT*; path planning; layout design

0 引言

近年来,随着电子电力及其相关产品的改进与发展,母线在一些高层建筑、轨道交通、环保、石化、智能型大厦和房地产项目,以及各种城市工业园的输配电系统中得到了快速发展和广泛应用[1]。母线布局设计是母线设计的核心环节,除了满足电气连接的基本需求之外,还需要综合考虑与相邻风火水电系统的干涉情况、制造和安装成本、以及工作使用的要求。母线布局路径的规划,本质上就是在三维空间中设计一条连接起止端,并满足上述约束的路径。目前母线设计主要依靠人工方式,费时费力,且难以寻获最优路径,设计出一种自动化的母线路径规划算法对工程中的母线布局设计具有重大意义。

为了解决路径规划问题,国内外学者在大量研究与不断探索的基础上,提出了很多路径规划的算法。常用的路径规划算法有 A*,蚁群算法,遗传算法,粒子群算法,人工势场法等[2-6],这些算法在解决传统的二维平面上的路径规划问题时,由于解空间较小并且问题简单,可以取得不错的效果。但是到了三维空间中,复杂的环境和高一维的空间容易引发维度灾难,使得这些算法的计算复杂度会剧烈增加,导致问题求解困难,收敛时间变长。近年来,基于抽样的路径规划算法被引入用来解决三维空间的路径规划问题,例如快速探索随机树(RRT)[7],通过随机抽取一些采样点来进行搜索,由于该算法避免构建显式的任务空间,进而大大降低了算法的计算复杂度,同时该算法具有概率完备性,这些特性使得该算法在解决路径规划相关问题中被广泛应用。然而,RRT 算法仍有一些缺点:由于其采用的是随机的采样策略,导致其生成的路径不稳定,规划出的解一般都不是最优的,同时 RRT 算法求解出的路径具有很多的冗余点,不仅增加了路径长度,同时导致路径存在较多不规则的弯折。而在母线的布线的过程中,仅支持 90 度弯折。为了解决 RRT 算法存在非概率最优解问题,2010 年 S. Karaman 和 E. Frazzoli 提出了 RRT* 算法,该算法在迭代过程中对生成的路径进行局部的重新布线,使得路径长度减小,确保算法可以收敛到最优解,并且与传统 RRT 算法计算的时间消耗只在一个常数比例内[8-9]。

母线布线的路径规划与普通线缆布线有显著区别。首先,母线是由铜排构成的,因此不能像线缆一样自由地改

收稿日期:2019-10-21; **修回日期**:2019-11-05。
基金项目:广东省自然科学基金项目(2016A030313453,2016A030313519)。
作者简介:刘 冲(1995-),男,福建龙岩人,硕士研究生,主要从事 CAD 与智能制造方向的研究。
通讯作者:周 驰(1973-),男,博士,副教授,硕士生导师,主要从事智能 CAD,金属塑性成形理论,板材成形数值模拟方向的研究。

变走向，也就是不能自由的弯曲。母线只能通过 90 度的弯头连接来改变走向，因此要求前后两段母线走向互相垂直。其次，母线的布局设计应该尽可能的少使用弯头，弯头数目的增多不仅增加制造成本，还会削弱母线的稳定性、安全性以及可维护性。最后，受到现场空间和插接箱取电位置的限制，在母线路径规划时还必须考虑母线的朝向，这在普通线缆布线中是不需要考虑的。

由上述分析可知，直接使用标准的 RRT * 算法进行求解无法满足母线布线设计的约束条件。但由于该算法具有良好的适应性，本文在 RRT * 算法的基础上，设计了一个母线的三维空间路径规划算法，该算法抛弃了原始 RRT * 算法扩展过程中直接将采样点和扩展树相连的方式，而是通过选用 corner 点将扩展树和采样点间接相连，在此基础上引入贪心的优化策略进一步满足工程中母线的相关约束。最后通过多组实例计算验证了该算法的有效性和稳定性。

1 RRT * 算法

快速随机搜索树（RRT）算法是一种基于采样的启发式路径搜索算法，其核心步骤开始开始建立以起点为根的扩展树，在全图中随机采样生成一个随机点，之后在从扩展树中找到离这个随机点最近的节点，接着尝试从这个最近的节点向随机点扩展一个步长，如果扩展后没有发生碰撞，则把这个扩展的新节点加入树中，否则重新采样。重复上述过程，直到扩展的节点离终点的距离小于一定值即可认为路径生成成功。最后通过路径回溯的方式在扩展树返回一条连接起点到终点的路径[10-12]。与其他智能搜索算法在路径搜索不同的是：该算法虽然没有实现完整性，但是其具有概率完备性，即该规划算法计算出解的概率会随着采样数量增加快速趋近于 1。因此，其在高维度空间中且复杂约束的条件下具有更高的求解效率。

相较于 RRT 算法，RRT * 算法主要在每次扩展成功后对新增节点的周围的节点尝试进行重新布线来获得渐进收敛到最优路径的特性[13]。RRT * 算法的伪代码如算法 1 所示：其中 V 表示扩展树的节点，E 表示扩展树中的边，T 表示扩展树由 V 和 E 组合而成。RRT * 算法在完成这些数据的初始化过程之后通过随机抽样开始其迭代过程：首先从给定的配置空间中选取一个随机点 x_{rand}，并从扩展树中找到距离 x_{rand} 最近的节点 $x_{nearest}$（3～4 行）。之后从 $x_{nearest}$ 向 x_{rand} 方向扩展出一个给点步长的新节点 x_{new}。再从扩展树中找到一组在以 x_{new} 为中心的球形区域内的点集合 X_{near}（5～6 行）。接着对 X_{near} 中的每个点按照从 x_{init} 到该点的成本和从该点到 x_{new} 的成本的和从小到大排序（7 行），获得列表 Ls。之后在 Ls 中取得可以连接 x_{init} 到 x_{new} 作为 x_{new} 的最佳父节点返回（8 行），如果找到了这个父节点 x_{min}，则将 x_{new} 作为该节点的子节点加入扩展树中，如图 1（a）所示。并进行重新布线（10～11 行），重新布线过程为检查 Ls 中每个节点 x，如果从 x_{init} 到 x_{new} 到 x 的成本小于原来从 x_{init} 到 x 的成本，并且可以将 x_{new} 和 x 相连，那么就断开 x 和原来父节点的连接，将 x_{new} 作为 x 的父节点连接到树中[14-16]。重新布线过程如图 1（b，c，d）所示。RRT * 算法通过引入重新布线的过程保证了渐进最优性。

Algorithm 1:RRT*(X_{init})

```
1  V ← {x_init}, E ← ∅, T ← (V, E);
2  for i 0 to N do
3  |  x_rand ← Sample();
4  |  x_near ← NearestVertices(x_rand, T);
5  |  x_new ← Steer(x_nearest, x_rand);
6  |  x_near ← NearVertices(x_new, T);
7  |  L_s ← GetSortedList(x_new, x_near);
8  |  x_min ← ChooseBestParent(x_new, L_s);
9  |  if x_min ≠∅ then
10 |  |  T ← InserlVertex(x_new, x_min, T);
11 |  |  T ← RewireVertices(x_new, L_s, T);
12 return T=(V,E);
```

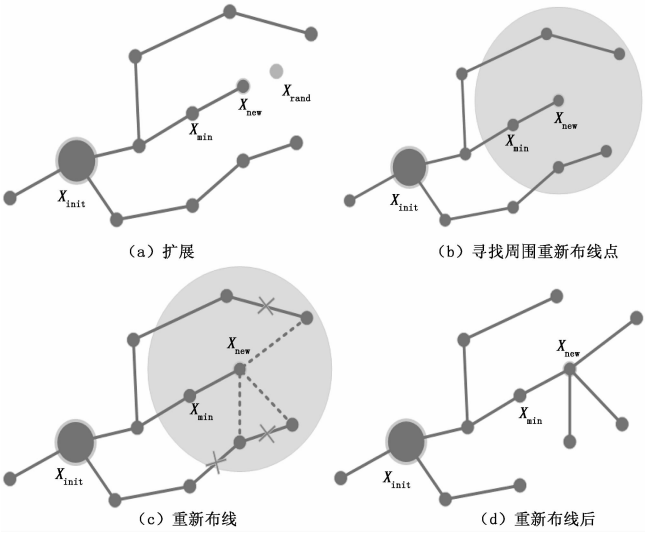

图 1 RRT * 迭代过程

2 母线布线路径规划算法

2.1 母线布线设计的约束与优化目标

在实际的母线布线设计中，建筑的内部结构复杂紧凑，布线空间狭窄且不规则。另外建筑内部除母线外，还存在大量的其他机电设备，如风管，水管，电缆桥架等。这些复杂凌乱的环境使得母线布线的难度大大增加。本文所考虑的工程约束主要有以下三条。

1）空间干涉约束：母线的路径应保证不与建筑中的承重墙、柱以及其他机电设备发生干涉，否则无法完成安装。

2）母线走向约束：不同于传统的线缆布线设计，在母线的布线设计中，相邻的两段母线应保证相互垂直，且每段母线的起点终点坐标只允许有一个分量不同，也就是说每段母线所在的直线必须与所建立的空间坐标轴平行。

3）母线朝向约束：需要保证母线的起止点朝向相对应，同时母线的路径中的特定位置也需要满足朝向的需求。

优化目标主要考虑以下两条：

1）长度最短：一条母线的长度应该尽可能的短，降低

成本，减小空间的占用。

2）弯头数目最少：每个弯头都会影响母线的安全性和稳定性，同时增加维护成本。

原始的 RRT＊迭代过程可以保证空间干涉约束和长度最短的优化目标，本文通过改变 RRT＊算法的的扩展方式来满足母线走向约束，并在迭代过程中引入路径优化过程满足母线朝向约束的弯头数目最少的优化目标。

2.2　母线走向约束的处理

在原始的 RRT＊ 算法中，找到 x_{min} 直接与 x_{new} 相连，此时生成的路径不符合相邻母线段相互垂直的要求，如图 1（a）所示。本文在找到 x_{min} 后，先寻找 x_{min} 与 x_{new} 之间的 $x_{corners}$，通过 $x_{corners}$ 把 x_{min} 与 x_{new} 间接相连，来保证生成的路径符合母线的布线要求。如图 2 所示。重新布线过程也采用这种扩展方式。

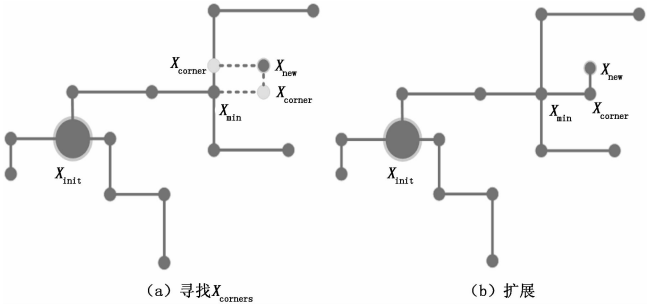

图 2　新的扩展方式

2.3　corner 点的选择

在三维环境中，两个点的坐标关系可以分成三种情况：

1）两个点的三个坐标分量只有一个不同，说明这两个点在平行于坐标轴的同一条直线上，此时不需要 corner 点直接相连即可满足母线的走向需求。

2）两个点的三个坐标分量有两个不同，说明这两个点在平行于坐标平面的同一平面内，需要一个 corner 点来连接 x_{min} 与 x_{new}，此时有两个备用的 corner 点供选择。如图 2（a）所示。

3）两个点的三个坐标分量都不同，需要两个（一对）corner 点来连接 x_{min} 与 x_{new}，此时有六对 corner 点对可供选择。

本文以第二种情况来说明 corner 点选择，其他两种情况类似。如图 3 所示，corner 点的选取时主要考虑下述三种情形：

1）当某个备选的 corner 点位于障碍物中时，此时违反了空间干涉约束，所以不应该选择，如图 3（a）所示。

2）当某个备选的 corner 点位于 x_{min} 到其父节点 $x_{min-parent}$ 的连线上时，并且此时 x_{min} 到 corner 点的方向与 $x_{min-parent}$ 到 x_{min} 方向相反时，即造成了方向冲突，此时生成的路径会产生对折的情况。这种路径不符合现实情况，所以这种 corner 点不应该被选择，如图 3（b）所示。

3）当某个备选的 corner 已经在扩展树中，并且该 corner 点的父节点不是 x_{min}，此时若选用该 corner 点会破坏原来的树结构，从而形成环，后续无法通过回溯取得路径，所以这种 corner 点不应该被选择，如图 3（c）所示。

除开上述三种情形之外，其他情况的 corner 点均可以被选择加入扩展树中，如果有多个或者多对 corner 点可供选择，那么选择构建过程中第一个或者第一对 corner 点。

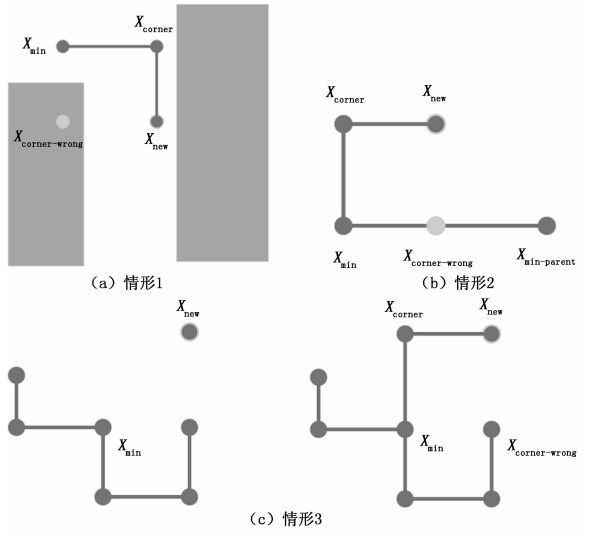

图 3　corner 点选择

2.4　路径优化策略

由 RRT＊算法生成的路径存在很多的冗余点，这些冗余点造成了路径过多的弯折。为使得生成的路径长度进一步减小和弯头的使用数量最少，在搜索过程中每次完成扩展后和每次重新布线后都加入以下优化步骤：

取出从起点到刚刚加入扩展树的节点路径，取出该路径最后的 5 个点，记作 a，b，c，d，e。此时考察点 a 和点 e 的之间的 corner 点，如果找到了满足的 corner 点，那么就通过这些 corner 点将点 a 和点 e 相连。由前文的分析可知，如果找到了满足的 corner 点，那么最多的情况只有两个，所以原来的 5 个点至少减少为 4 个，这样就完成了一次优化过程，并且这次优化过后这条被优化路径的长度必定小于等于优化前的长度。之后再取出优化后路径的最后的五个点，重复上述过程。直到某次优化前后路径的节点数目不变，说明路径已经达到了需要最少弯头的情况。

不同于传统的线缆，母线由起点引出时就有一个固定的 N 面，而不同的走向会造成 N 面朝向的不同。相同的母线由于走向的不同导致终点的 N 面朝向不同，而在终点或者路径中安装插接箱的位置也有 N 面朝向的要求。所以在算法迭代的过程中（在引入新节点后和每一次优化路径后）还需要实时计算并记录扩展树中的每一段路径的 N 面朝向，在每次尝试获取连接终点解时需要考虑 N 面朝向是否符合要求。

改进后的算法伪代码如算法 2 所示。主要改进在第 8 行，第 10 行和第 11 行。在第 8 行，我们不仅需要得到 x_{min}，同时还需要获得将 x_{min} 和 x_{new} 相连的 corner 点，这一步如算法 3 所示。在算法 3 中，我们遍历已经排好序的列表 Ls，依次对 Ls 每个点尝试获取到 x_{new} 的 corner 点，一旦获

取到就返回。尝试获取 corner 点这一步如算法 4 所示。算法 2 中第 10 行为将 corner 点和 x_{new} 一并加入树中，第 11 行为按照本文新的扩展方式对 x_{new} 周围节点的重新布线过程，这两步每次扩展成功后均加入 2.3 中的优化策略。

```
Algorithm 2:New-RRT*(X_init)
1 V←{X_init}, E←0, T←(V,E);
2 for i← 0 to N do
3 |  X_rand ←Sample();
4 |  X_nearest ←NearestVertices(X_rand, T);
5 |  X_new ←Steer(X_nearest, X_rand);
6 |  X_near ←NearVertices(X_new, T);
7 |  L_s ←GetSortedList(X_new, X_near);
8 |  X_min, corner ←NewChooseBestParent(X_new, L_s);
9 |  if X_min ≠ Ø then
10|  |  T ← NewInsertVertex(X_new, corner, T);
11|  |  T ← NewRewireVertices(X_new, L_s, T);
12 return T = (V,E);

Algorithm 3: NewChooseBestParent(X_new, L_s)
1 for (x,c)∈ L_s do
2 |  corner← GetCorner(x, X_new);
3 |  if corner≠Ø then
4 |  |  return x, corner;
5 return Ø;

Algorithm 4: GetCorner(x, X_new)
1 for corner ∈ ReservedCorners(x, X_new)do
2 |  if ObstacleFree(x, corner, x_new)then
3 |  |  return corner;
4 return Ø;
```

3 仿真结果及分析

在实验中，规划区域为边长 100 的正方形，起点为 (30，20)，终点为 (60，82)。RRT * 算法采用的代价函数为欧式距离，本文算法采用的是曼哈顿距离。图 4 和图 5 分别为 RRT * 算法和本文算法不同阶段的情况。由图 4 可以看出，原始的 RRT * 算法虽然可以收敛到理论最优解，但是由于扩展方式的原因导致生成的路径不符合母线的走向要求，弯折角度没有规律。由图 5 可以看出，本文改进的扩展方式可以很好的生成满足母线走向要求的路径，同时也保留了原始 RRT * 算法的渐进最优特性，如图 6 所示 (本环境的理论最优值为 202)，随着迭代次数的增加，算法取得的路径长度趋近于理论最优值。但是此时算法迭代过程中并没有加入对路径的优化策略，所形成的的路径折弯过多。加入 2.3 中描述的路径优化策略后，算法的迭代过程如图 7 所示。可以看出，相较于未加入路径策略时，此时扩展树中的节点到起始点的路径都经过优化大大减少了折弯的次数。

最后，为验证本文算法在三维环境下的效果，根据某

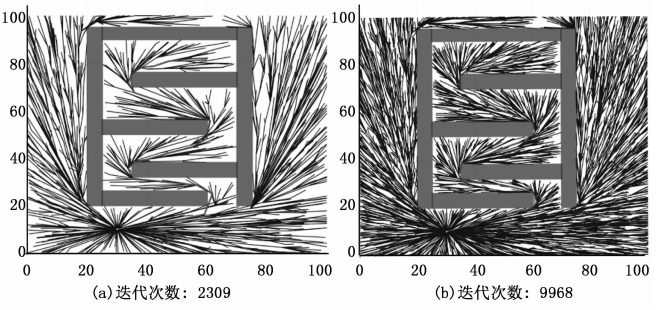

图 4 RRT * 算法

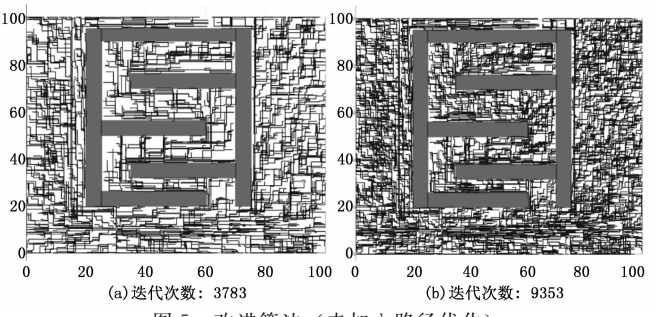

图 5 改进算法（未加入路径优化）

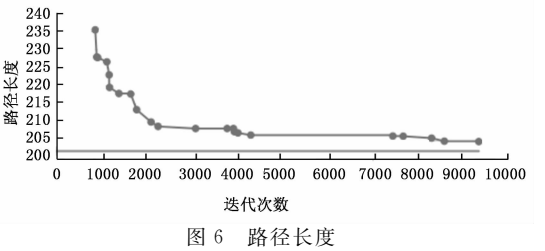

图 6 路径长度

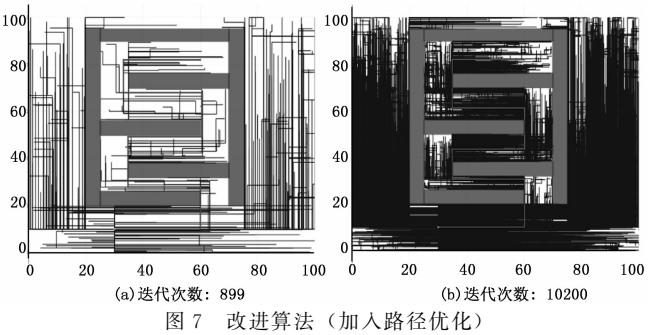

图 7 改进算法（加入路径优化）

公司配电站项目建立简化三维环境的仿真环境如图 8 所示。该配电站一共有三层，其中一条母线需要从一楼的配电柜引出，经过二楼，三楼楼板预留的墙洞引入到三楼配电柜。本文算法在两种不同需求情况下分别给出了两条路径。其中左边的路径在一楼到三楼的上升段没有朝向的要求，而右边的路径在该段需要挂接插接箱造成对该段的朝向有要求。从图中可以看出，两条路径长度相同且均满足在各自需求下的约束。比较左右两条路径发现，左边路径弯折了 8 次，右边路径弯折了 10 次，为了满足上升段的朝向要求，右边的路径比左边多出了两个折弯，整条母线需要多引入两个弯头。

(下转第 257 页)