# 公共交通网络中的可达性查询

## 摘要

   给定空间网络中的查询点，可达性查询检索在关于网络约束的特定时间量内可从查询点到达的所有感兴趣点。可达性查询具有许多有趣的应用，并且对于道路网络已经提出了有效的解决方案。道路网络是时间无关的，即，穿过边缘的成本随时间是恒定的。道路网络的高效算法严重依赖于预先计算最短路径。然而，在公共交通网络中，边缘成本取决于时间表，这使得道路网络的大多数解决方案效率低下。特别是，节点对之间的最短路径不能容易地预先计算，因为它们随时间变化。

   这项工作的目标是为公共交通网络中的可达性查询开发有效的解决方案。核心思想是将网络划分为单元格并计算时间上限和下限以遍历单元格。在查询时，可到达区域逐个单元地扩展（而不是逐个扩展）。使用上限扩展可达到的所有兴趣点都是结果的一部分;可以安全地丢弃在下限扩展中无法到达的所有点;所有其他节点都是候选节点，必须进行验证。本文介绍了扩展算法，并讨论了有关良好网络分区，有效边界和有效候选验证的有趣研究方向。

## 介绍

分析地理位置的可达性有许多有趣的应用，例如，在城市规划中分配医院和学校，在地理营销中定位特许经营店，或在移动应用中探索周围环境。 在空间网络中，位置的可达性不仅取决于欧几里德距离，还必须考虑网络的限制和结构。 例如，汽车只能在道路上行驶，必须遵守交通规则，公交车有时间表，只能在公交车站登车。 允许不同传输模式（例如，行人，汽车和公共交通）的网络被称为多模式网络。

空间网络上的查询对空间和时空数据库至关重要[14,18,27,29]。 将空间网络集成到数据库中可以实现尊重网络约束的查询，并为新的应用领域提供支持。 许多商业和开源系统为空间网络提供扩展，例如，Oracle Spatial，PostGIS for PostgreSQL和ESRI ArcGIS Network Analyst。 在技术层面，空间网络对数据建模，索引和查询处理提出了挑战。 空间网络查询的示例是最短路径查询，空间连接，范围查询和最近邻查询[19,21,23,24]。

可达性查询检索网络中的所有兴趣点（POI），其在特定时间点内在给定时间范围内（成本预算）到达（可以从其到达）给定查询点。 例如，在查询“返回所有可以在步行或通过公共交通工具在10分钟内到达学校的学生”，学生宿舍是POI，查询点是学校，成本预算是10分钟， 时间点是早上8点。

解决可达性查询有两种基本方法：（a）计算到达查询点并与POI集相交的网络子集，（b）计算每个POI与查询点之间的最短路径，并保留这些POI 足够接近。 第一种方法对于具有许多结果点的网络区域是有益的; 遗憾的是，该算法还必须扩展不包含任何结果的网络区域。 第二种方法的效率取决于POI的总数，并且即使对于小的结果大小也可能需要大量的最短路径计算。

对于道路网络，已经提出了许多解决方案，例如，范围网络扩展（RNE），范围欧几里德限制（RER）[24]和增量网络限制[9]。 不幸的是，这些方法不适用于公共交通网络：它们依赖于欧几里德距离作为下限，这在时间表的存在下是无用的。

可达性查询的问题与最短路径查询的计算密切相关。最短路径问题的最快算法严重依赖于预计算。如果预先计算所有节点对之间的最短路径，则将最短路径计算减少为单个查找;然而，这种方法对于大型网络来说是不可行的。收缩层次结构（CH）[12]，自定义路径规划（CRP）[8]和传输节点路由（TNR）[4]等算法存储选定节点集之间的距离，这些节点允许非常快速的查询时间和可行的索引大小在大陆规模的网络。不幸的是，这些方法假设道路网络，其具有遍历边缘的恒定成本。因此，在每对节点之间存在单个最短路径。在公共传输网络中，两个节点之间的最短路径取决于查询时间。边际成本由时间表给出并随时间变化。成本波动可能很大：在几分钟（白天）和几个小时（过夜）之间。

如Bast等人所述。 [2]，“公共交通系统的旅程规划虽然在概念上类似于[道路网络中的旅程规划]，但由于其固有的时间依赖性和多标准性质，因此是一个非常困难的问题”。 我们确定了两个主要问题：（1）时间相关的边缘成本导致距离和最短路径根据查询时间而变化。 （2）网络（例如火车站）中单个顶点的入射边缘的数量远大于道路网络中的入射边缘，其中街道交叉点具有很少的入射边缘[2]。 边缘成本的时间依赖性和较大的顶点邻域使得大多数依赖于预处理的方法对于公共交通网络是不可行的。

这项工作的目标是使用预计算为公共交通网络中的可达性查询找到有效的解决方案。 预计算的时间和空间应扩展到大陆规模网络; 结果索引应支持增量更新以响应本地网络变化（例如，新的总线或更改时间表）。

我们的解决方案基于图分区。 我们将运输网络分成单元格并预先计算上下限以遍历每个单元格。 预先计算的边界用于逐个单元（而不是逐个边缘）有效地扩展可到达区域。 在上限扩展所找到的区域内的兴趣点是可到达的，而在下界区域之外的点是不可到达的。 位于下限但在上限区域之外的所有兴趣点都是候选者，必须进行验证。

## 相关工作

在空间网络中，可达性查询与最短路径（SP）问题密切相关，近年来受到了研究界的广泛关注。 大多数当前的SP算法都基于Dijkstra算法[10]或A \*算法[17]。 这些算法不需要任何预计算。 Dijkstra的算法遵循扩展技术，该技术在所有可能的方向上访问边缘，直到达到目标，这使得算法对于大型网络来说太慢。 A \*在Dijkstra的方法之上使用启发式（例如，欧几里德距离）来指导扩展。 一个好的启发式可以防止不必要的顶点扩展。 已经为Dijkstra算法[25,20,7,30]和A \*算法[15,16]提出了不同的优化技术。

通过使用广泛的预计算，SP查询的在线时间可以大大减少。 高速公路层次结构（HH）[28]将Dijkstra算法的查询时间缩短了三个数量级[28]。 收缩层次算法[12,13]使用HH的思想，该算法在层次结构中组织顶点并应用收缩技术来减少查询处理的图形大小。 通过在图中添加新边来预先计算SP，这些边在查询时使用。 这种CH的预处理技术由SHARC（Shortcuts + ArcFlags）[5]使用。 SHARC被认为是最快的单向算法之一，而CH仅适用于双向查询。

另一种主要基于图分区的SP计算算法是预计算簇距离（PCD）[22]。 它将图分割为不相交的簇，并预先计算所有簇对之间的SP。 公交节点路由（TNR）[3]基于道路网络中的直观观察：到达目的地的所有行程通常共享少量包含称为接入节点的重要道路交叉点的路径。 使用这种思想，TNR首先在图中找到接入节点，并预先计算与接入节点之间的最短路径距离。

定制路线规划（CPR）[8]基于图形分离技术，用于道路网络上的实时查询。 CRP的预处理是与度量无关的，这是优于CH和遵循类似预计算技术的算法的优势：CH提供快速查询时间，但对边缘成本的微小变化敏感。 另一种基于图分离器的算法是GRASP（Graph separator，RAnge，Shortest Path）[11]，它使用多级图分区技术。

大多数上述算法已在道路和公共交通网络中进行了评估，结果在[2]中进行了讨论。 评估表明，这两种网络之间存在巨大的性能差距。 这是由于公共交通网络的时间依赖性边缘成本，这使得许多算法的预计算工作对于大型网络是不可行的。

为了将公共交通网络建模为图表，已经广泛使用了两种方法：时间扩展和时间依赖方法。 在时间扩展模型中，每个站（例如，公共汽车站或火车站）中的每个到达和离开事件由顶点表示，并且边缘用于表示两个事件之间的链接。 遍历每个边缘的成本是源事件和目标事件之间的时间差。 在时间相关模型中，每个站由单个顶点表示，如果在它们之间存在连接，则在两个站之间使用边。 取决于源顶点处的离开时间的成本函数与每个边缘相关联。 第一种方法的缺点是结果图的大尺寸。 但是，由于每个事件都由顶点表示，因此为道路网络设计的大多数SP算法都可以应用于此方法的结果图中。 有关两种建模技术的详细说明，请参见[26,1]。

范围欧几里德限制（RER）和范围网络扩展（RNE）[24]使用类似于Dijkstra算法的扩展技术来回答可达性查询。 RNE首先确定可到达区域，然后将该道路网络区域与该组POI相交。 RER将相关POI的数量限制为使用欧几里德距离可到达的点，即忽略网络。 对于这些候选点，计算最短路径以便去除误报。 邓等人。 [9]减少RER中候选者的最短路径计算次数：类似于A \*最短路径算法[17]，在最短路径计算期间扩展的每个节点保持欧几里得下界; 首先扩展最接近候选者之一的节点，并且当所有剩余候选者的下限超过成本预算时，扩展停止。

鲍尔等人。 [6]考虑多模式网络而不依赖于预计算。 他们使用Dijkstra [10]扩展查询点，并通过使已经处理的网络节点到期来提高内存使用率。 计算的结果是所谓的等时线，其是在给定时间点网络的可到达部分。 由于必须扩展等时线中的所有边缘，因此这种方法不能扩展到大型网络和等时线。

1. **问题定义**

我们将空间网络建模为具有顶点V和边E的有向图G =（V，E）。顶点嵌入在平面中，即它们具有坐标。 成本函数c（e，t）为每个边e∈E分配正成本：在时间t（顶点x处的开始时间）遍历边e =（x，y）的时间。 两个节点u，v∈V之间的距离d t（u，v）是在时间t从u行进到v所需的最小成本（如果v不能从u到达，则d t（u，v）=∞）。 如果点p位于G的边缘或顶点上，则点G在G中。遍历部分边缘的成本是总成本的线性分数。

**定义1（可达性查询）** 给定空间网络G =（V，E），成本预算Δt，G中的一组关注点P，G中的查询点q和查询时间t。 可达性查询计算所有兴趣点R⊆P，它们可以在开始时间t从q到达，成本为Δt：



反向可达性查询计算在到达时间t可以通过成本Δt到达q的所有兴趣点。

这项工作的主要目的是为公共交通网络中的可达性查询开发一种算法。 用于可达性查询的当前解决方案要么限于道路网络（即，与时间无关的边缘成本），要么不扩展到大型网络。

1. **基于分区的可达性查询方法**

在本节中，我们基于网络分区和每个分区内的本地预计算，绘制了一种可达性查询技术。

我们专注于公共交通网络，这对预计算技术构成了主要挑战。 在公共交通网络图中，节点是站，例如公共汽车站或火车站。 与每个边相关联的与时间相关的成本函数表示两个站之间的连接，并且取决于时间表和等待时间。

为了简化讨论，我们的算法的以下表示假定了无向图。 在本节的后面部分，我们将讨论有向图的可能扩展，它是公共交通网络的现实模型。

## 4.1预计算

### 4.1.1分区

我们将图分割成n个单元格。 单元C是连接的子图，由节点VC⊆V和边E C E组成，其中E C是连接V C中的一对节点的所有边的集合。 单元是不相交的（即，对于任何对C i 6 = C j：V C，i∩V C，j =∅）并覆盖图的节点（即，∪ni = 1 V C，i = V）。 连接来自不同单元的节点的边是边界边，边界边的端点是边界节点。 单元C i的边界节点集用B i表示。

### 4.1.2图形扩展和边界计算

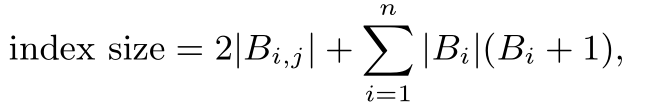
我们向图中添加新节点和边。 （1）在每个单元C i中，我们连接所有边界节点对（b，b 0），b，b0∈Bi。 （2）我们添加一个虚拟节点？ i到每个单元C i并将虚拟节点连接到所有边界节点。 虚拟节点不是原始图的一部分。 边界节点b∈Bi与虚拟节点之间的成本？ i是b与单元C i中距离b最远的节点之间的成本。

我们计算特定节点对之间的时间上限和下限。 上限是最慢连接时两个节点之间的最短路径，即所有时间点上的最大最短路径。 下限是最快连接时的最短路径。 我们预先计算以下边缘的上限和下限：

•边界节点：每个单元内所有边界节点对之间的边;

•虚拟节点：小区C i和小区的虚拟节点CI中的每个边界节点B∈B I之间的边缘;

•所有边界边缘。

索引大小（即预先计算的边界数）是

其中B i，j是G中所有边界边缘的集合，B i是单元C i的边界节点集合。 注意，B i，j是原始图G中的节点E的（希望是小的）子集。该求和将每个单元内计算的所有上限和下限相加。 加数仅取决于每个单元中的边界节点的数量，并且总和随着单元的数量线性增加。 因此，我们希望预计算能够扩展到非常大的图形。

可以响应于调度更改，边缘插入和边缘移除来递增地更新索引。 只有更新的边和包含更新边的单元格才会受到影响。 所有其他单元格和边框边缘的上限和下限不受更新的影响。

### 4.2 候选生成

我们的可达性算法的核心思想是逐个单元地扩展而不是逐个扩展。具有上限（最慢连接）的扩展定义了在给定成本预算内始终可以达到的区域Ru⊆V，与开始时间无关。上限区域中的所有POI都是结果集的一部分。下限扩展（最快连接）定义了在假设每个边缘使用最快连接遍历的情况下可达到的区域Rl⊆V，与遍历时间无关。可以安全地拒绝下限区域R l之外的所有POI，因为它们不是查询结果的一部分。上限和下限区域之间的POI，R l \ R u（Ru⊆Rl）是我们必须验证的候选者。与针对所有POI计算SP的方法相比，在我们的方法中，SP计算仅限于候选者。如下面将讨论的，在一些情况下，可能没有必要在候选者和查询点之间计算SP以便验证候选者。

接下来我们讨论上限和下限的扩展。 对于扩展算法，我们只需要考虑边界节点，边界边和虚拟节点，即我们已预先计算边界的所有边。 扩展区域以单元为单位而不是节点或边缘。

### 4.2.1上限膨胀

假设查询点q位于具有边界节点Bq⊆Vq和虚拟节点C q的单元C q =（V q，E q）中。 扩展必须首先处理小区C q，然后进入相邻小区，直到预算耗尽。

**初始化**： 我们首先扩展细胞C q。 结果是相邻小区的一组（可能是空的）边界节点B，即B6⊆Vq; 每个边界节点bi∈B具有时间预算Δti。 扩展过程如下：

1. 展开单元C q：如果C q中的所有节点都可以从成本预算Δt的q到达，则用C q，R u = V q的节点初始化上界区域; 否则R u =∅。 对于任何节点bq∈Bq，如果Δt≥2ub（b q，cq），则q q可从q到达，其中ub（x，y）是节点x，y之间的上界。 直觉是我们从C q中的任何节点（即也来自q）到达b q，成本至多为ub（bq，cq）; 从b q到达C q中的任何其他节点，具有相同的成本上限。
2. 扩展到相邻的单元格。 计算B，可通过边界边界从C q的边界节点到达的所有节点的集合，并具有非负的成本预算。 可通过边界节点bq∈Bq到达的节点bi∈B的成本预算是Δti=Δt-ub（b q，c q）-ub（b q，b i）。

**扩展步骤：** 给定Ru外部的一组边界节点B和每个bi∈B的时间预算Δti，扩展如下进行。 对于每个节点bi∈B：

1. 展开当前单元格：令C i为包含边界节点b i的单元格。 如果C i中的所有节点都可以在b i处以剩余成本预算到达，则将该单元添加到上限区域，即，如果Δti≥ub（b i，ωi）则R u = R u∪V i。 否则只添加边界节点b i：R u = R u∪{b i}。
2. 扩展到相邻的单元格。 跟随b i和B i中所有其他边界节点之间的边缘。 节点bj∈Bi\ {b i}的预算是Δtj=Δti-ub（b i，b j）。 对于所有边界节点bj∈Bi：（a）如果tj≥0，则R u = Ru∪{b j}; （b）对于从b j经边界边缘到达的所有节点d：计算节点d的成本预算t d = t j -ub（b j，d）; 如果td≥0，则B =B∪{d}。
3. 从B中删除b i。

重复扩展步骤，直到要扩展的边界节点集为空，B =∅。

**4.2.2下限扩展**

具有下限的扩展类似于具有上限的扩展。 主要区别是：

1. 扩展使用下限而不是上限来计算B中节点的预算。
2. 如果（a）C i包含查询节点q（并且Δt> 0），或者（b）C i包含具有预算Δ的bi∈B的节点，则将单元C i添加到下边界区域R l。 ti> 0。

我们需要通过非零预算扩展可通过下限到达的所有单元。 即使我们知道单元中的某些节点是不可达的（即，虚拟节点的下限超出预算），我们必须确保可到达的节点子集在候选集中。

### 4.3扩展算法

在剩下的部分中，我们将讨论基本可达性算法的扩展。 特别是，我们讨论了有向图的扩展，假设将图分割成细胞的标准，引入时间依赖边界的概念，并指出候选验证的机会。

**4.3.1有向图**

公共交通网络通常将被建模为有向图。 在有向图中，一对节点之间的界限可以取决于方向。 我们扩展预计算算法以计算每个相关节点对的两个上限和下限（每个方向一个）。 例如，我们将边界节点b i的上限与虚拟节点区分开来？ 我从反方向的上限。 总的来说，预计算的大小将增加两倍。 扩展算法必须适应有向边界，例如，当扩展单元格C q时，条件将是Δt≥ub（？q，bq）+ ub（bq，？q）（而不是Δt≥2ub（bq） ，？q），参见第4.2.1节）。

在无向图中，单元中的所有节点都是成对可达的。 通常，这可能不适用于有向图，即，一对节点之间的界限可以是无限的。 虽然仍然正确，但扩展算法对受影响的细胞效果较差。 由于我们为公共交通网络建模，因此这种情况不太可能发生 如果某个节点只有一个方向的路径，我们可以到达该站但不会离开它（反之亦然）。

**4.3.2有效的分区策略**

到目前为止，我们假设将图分割成单元格。 我们的算法对于任何分区都是正确的，但显然分区策略会对方法的效率产生影响。

我们确定了一组有效分区应满足的属性：

•每个小区的边界节点数应该很小;

•边界边缘的总数应该很小;

•细胞上限和下限之间的差异应该很小;

•单元中的所有节点应成对可达;

•不同的细胞应具有相似的大小（即，就上限遍历时间而言是均质的）。

对于单元尺寸，存在平衡：大单元提高了扩展速度，因为单元的处理时间与其大小无关。 小细胞允许更细粒度的扩展并导致更小的候选集。

我们期望用于计算最佳分区的算法将具有不可行的运行时间。 因此，我们将针对最优算法的启发式或近似。

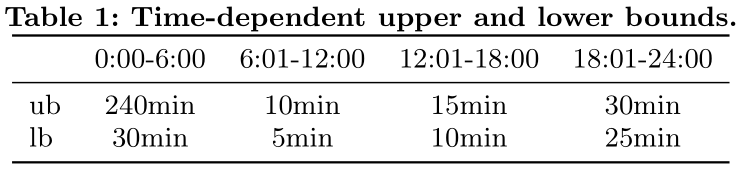
**4.3.3与时间有关的界限**

在4.1节中，我们将上（下）边界定义为在所有时间点上最慢（最快）连接时两个节点之间的最短路径。 对于公共交通时间表而言，这些界限可能非常宽松，其中连接的频率通常根据一天中的时间而变化。 例如，在高峰时段，公共汽车以最高频率行驶，而夜间没有公共汽车。 这导致分钟的下限和小时的上限。 对于白天的可达性查询，上限太松，导致上限区域非常小。 同样地，下限对于晚上的离开而言太小，导致大的候选集。

我们的解决方案是时间限制的。 我们将计划分成间隔，并为每个间隔计算不同的界限。 在扩展期间，我们检查特定边界节点b处的最早（t 1）和最晚（t 2）到达时间。 上限是区间[t 1，t 2]中的最大上限; 下限是区间[t 1，t 2]中的最小下限。 计算到达时间的间隔，同时用上限和下限进行扩展。 设b是在上限扩展期间的下限扩展和时间预算Δtu期间具有时间预算Δt1的边界节点。 利用查询时间t（即，查询节点q处的开始时间），节点b处的最早到达时间是t 1 = t +Δt-Δt11; 最近到达时间t 2 = t +Δt-Δtui。

我们有不同的选项将时间表分成间隔：我们要么使用固定长度的间隔（例如，6小时），要么使用可变长度的间隔。 可变长度间隔将导致更严格的界限，因为我们可以选择间隔使得边界最接近真实的时间表。 例如，我们可以创建覆盖高峰时段，日程安排和夜班时间表的间隔。 可以为每对节点选择单独的间隔。 对于给定节点对，间隔甚至可以在上限和下限之间不同。 限制因素是预计算大小，其随着每个附加间隔而增加。

表1示出了边界节点b和虚拟节点α之间的时间相关边界的示例。 24小时的时间表分为4个时间间隔。 该表显示了在不同时间间隔内从b到b单元中任何节点的最小和最大行程时间。 例如，在6:01和12:00之间，上限和下限分别为10分钟和5分钟，而界限分别为240分钟。 晚上30分钟。 当我们在时间间隔[11：17,13：03]到达节点b时，上限为15分钟，下限为5分钟。



**4.3.4高效的候选验证**

验证候选节点的直接方法是计算查询点和候选者之间的最短路径。 由于我们有预先计算的下界，我们不需要求助于Dijkstra算法，但可以使用有效的A \*算法并进行定向搜索。

在某些情况下，我们可以在不计算最短路径的情况下拒绝或接受候选人。 例如，可以使用处于部分可达单元中的上边界区域的边界节点。 这些节点具有剩余时间预算。 在给定时间预算内可以从边界节点到达的所有候选者（考虑边界节点和候选者之间的最短路径）是结果的一部分。 类似地，下界区域的未到达其小区中的所有节点的边界节点可用于拒绝该小区中的候选者。

1. **结论**

我们提出了一种通过将图划分为单元并使用基于上限和下限的新型扩展技术来计算公共交通网络中的可达性查询的技术。 扩展算法生成一组可到达的点，一组可以在没有验证的情况下被拒绝的无法到达的点，以及一组必须被验证的候选者。 为了获得更紧密的界限（更接近的上限和下限），考虑时间依赖的上限和下限。 这反映了公共交通时刻表中的频率模式，其根据一天中的时间和日期（例如，夜间公共汽车，周末时间表，假期时间表）而不同。 更好的界限导致更小的候选集。

与相关工作中使用的类似Dijkstra的扩展技术不同，我们使用预计算并期望显着减少可达性查询的最短路径计算的数量。 这将带来更好的在线查询性能。 预计算与网络大小线性地缩放，并且本地改变（例如，调度改变，新的传输路由）仅需要预先计算的索引的本地更新。

作为未来的工作，我们计划开发一种有效的分区算法，以满足本文中概述的属性。 我们将评估实际公共交通网络上提出的技术，并从相关工作中将相关竞争对手的绩效进行实证比较。 此外，我们计划将我们的公共交通网络解决方案推广到多式联运网络，其中还包括道路和行人边缘。