多维代价图模型上最优路径查询问题的研究

杨雅君 高 宏 李建中
(哈尔滨工业大学计算机科学与工程学院 哈尔滨 150001)

摘 要 近年来,图数据模型被广泛地用于刻画现实世界中各种各样的实体间的复杂关系.最短路径查询是图研究领域中一类非常重要的查询并有着广泛的应用.然而,目前大多数关于最短路径的查询都是定义在单代价(权重)图模型下的.现实世界中,基于单一代价所选择的最短路径并不明智,比如路程最短的路径需要花费极高的费用.该文中,作者介绍了多维代价图模型的概念,并给出了多维代价图模型下基于函数的最优路径的定义.现有的计算最短路径的方法都利用了最短路径的子路径最优的性质:最短路径上的任意两点间的子路径是这两点的最短路径.因此,在计算最短路径的过程中,对访问过的每个顶点,只需保留起点到该点的最短路径即可.不幸的是,多维代价图模型下,当评分函数是非线性的时候,子路径最优的性质并不成立.因此,目前的方法均不能应用于多维代价图模型下基于函数的最优路径查询问题.该文给出了一个 best-first search 分支界限法并给出3种优化策略.进一步,给出了一个顶点过滤算法,该算法能从图中过滤掉大部分不属于最优路径的顶点.最后,用真实数据集上的实验验证了算法的有效性.

关键词 多维代价图;最短路径;目标函数;路径查询
中图法分类号 TP311 **DOI号**: 10.3724/SP.J.1016.2012.02147

Optimal Path Query Based on Cost Function Over Multi-Cost Graphs

YANG Ya-Jun GAO Hong LI Jian-Zhong
(*School of Computer Science and Engineering*, *Harbin Institute of Technology*, *Harbin* 150001)

Abstract Graphs have been widely used to model complex relationships among various entities in real applications. Shortest path query is an important problem in graphs and has been well-studied. However, most approaches for shortest path are based on single-cost (weight) graphs. However, it is non-sufficient that considering only one cost type. In this paper, we introduce the definition of multi-cost graph and propose a new query: the optimal path query based on function over multi-cost graphs. Most existing methods to compute shortest path utilize the property of optimal sub-path in shortest path: any sub-path of a shortest path is also a shortest path. Thus, they only need to maintain the shortest path from starting node to any visited node (not ending node) when computing shortest path. Unfortunately, the optimal sub-path property doesn't hold in multi-cost graphs. We propose a best-first search branch and bound algorithm and three optimizing strategies. We also propose a vertex-filtering algorithm to filter a large proportion of vertices from graph. We confirmed the effectiveness and efficiency of our algorithms using real-life datasets in experiments.

Keywords multi-cost graph; shortest path; objective function; path query

收稿日期:2012-06-30;最终修改稿收到日期:2012-08-17.本课题得到国家"九七三"重点基础研究发展规划项目基金(2012CB316200)、国家自然科学基金(60903016,61003046,60533110,60773063,61173022)、黑龙江省自然科学基金(F201031)、中国博士后科学基金(20110491064)、黑龙江省博士后基金(LBH-Z09140)及哈尔滨工业大学科研创新基金"中央高校基本科研业务费专项基金"(HIT.NSRIF.2010060)资助.**杨雅君**,男,1983年生,博士研究生,主要研究方向为图数据库和图挖掘、海量数据处理.E-mail: yjyang@hit.edu.cn.**高 宏**,女,1966年生,博士,教授,博士生导师,主要研究领域为图数据库和图挖掘、海量数据处理.**李建中**,男,1950年生,博士,教授,博士生导师,主要研究领域为图数据库和图挖掘、海量数据处理、无线传感器网络.

1 引 言

近年来,图数据模型被广泛地用于刻画现实世界中各种各样实体间的复杂关系,如交通网、XML数据库以及社交网络等等.其中,最短路径查询是图研究领域中一类非常关键的问题并有着重要的应用.例如,在交通网中,顶点代表城市,两个城市之间存在边意味着它们之间存在一条公路,最短路径查询返回两座城市之间最短的公路里程.社交网络中,顶点代表人,边代表人和人之间存在直接的交往关系,边的权值反映了两人直接关系的亲密程度,两点之间的最短路径则反映了社交网络中两人联系的紧密度.

目前,大多数工作都假设图模型中边的代价(权值)是单一的.事实上,不同实体之间的关系可以从多个角度来进行考察,因而描述这些关系的边就需要用多个代价来进行刻画.比如在交通网中,两座城市之间建有一条高速公路,公路的长度、通过该公路的费用以及该公路的拥挤程度分别从不同角度刻画了通行这条公路的代价.因此,在描述该网络的图模型中,边上的代价应该是多维的,进而任意两个顶点间路径的代价也是多维的.现实生活中,用户对最短路径的选择应该是基于多维代价的综合考虑,仅仅依靠某一代价做出的最短路径选择并不明智.比如,公路网中,总路程最短的路径有着相当高的经济代价.在此情况下,用户宁愿选择一条路程稍长的路径,但是其经济代价远远小于路程最短的路径.因此,如何根据用户的喜好,返回给用户一个综合考虑的最优路径成为了一个重要的问题.

本文假设用户给出一个评分函数 $f(\cdot)$,$f(\cdot)$反映了用户对不同类别的代价的重视程度,不同用户给出的函数 $f(\cdot)$可能不同.函数 $f(\cdot)$根据路径在多个维度上的代价计算出一个综合代价得分.给定起点和终点,本文的目的是找到函数 $f(\cdot)$下得分最小的路径,我们称之为基于函数 $f(\cdot)$的最优路径.

Dijkstra 算法是求解图上两点间最短路径的经典算法.Dijkstra 算法的核心思想是利用了最短路径的子路径最优性质:最短路径上的任意一条子路径也是最短路径.因此,在计算最短路径的过程中,算法对访问过的每个顶点,只需要保存起点到该点的最短路径即可.

目前有关最短路径的工作都是基于 Dijkstra 算法的核心思想[1-6].这些工作的主要方法是:在图上建立索引,索引内部顶点之间的最短路径已经被计算并且保存起来.当一个查询到来,算法先检索索引内部需要访问的最短路径,然后与索引外部的最短路径进行优化连接,最终得到查询结果.索引不同,算法效率也就不同.然而,这些方法,无论是在建立索引过程中,还是在利用索引完成最短路径查询时,都利用了子路径最优的性质:即只需维护索引内部顶点间的最短路径,同时,对运算过程中访问过的节点,只需维护起点到它的最短路径即可.不幸的是,当用户提供的评分函数是非线性的时候,子路径最优的性质在多维代价图中并不成立(详见 2.2 节).因此,目前关于最短路径的方法都不能解决本文所要面对的问题.

运筹学领域中的相关研究表明,非线性代价函数在物流网应用中广泛存在.在物流网中,运输大量货物时,在不同时段或不同地段所花费的运输费用随着通行距离的不同而发生变化.如文献[7]所示,该费用是与距离相关的分段线性函数或者分段凹函数.这说明,该费用代价函数是与距离相关的非线性函数.进而,费用与距离以任何形式组合得到的函数也是非线性的.另外,在某些特殊情况下,如自然灾害中急需物资的运输或者战争中军需物资的运输,运输网络可能遭到破坏.如文献[8]所示,此时距离和费用的不同度量标准导致了非线性代价函数的存在(多项式函数或凸函数).本文中所给出的算法,可适用于各种类型的代价函数(线性的和非线性的).

多维代价图上的多标准 Pareto 最优路径计算(MCPP)问题在运筹学领域被广泛研究[9-11].给定起点和终点,MCPP 的目的是找到起点到终点的互不支配的全部路径,即 Pareto 最优路径.显然,一条 Pareto 最优路径也是一条 skyline 路径.大多数解决 MCPP 的方法都基于 Dijkstra 的算法思想,即任意一条 Pareto 最优路径上的子路径也应该为一条 Pareto 最优路径.这些方法需迭代地计算出目标 Pareto 路径所经过全部顶点的 Pareto 最优路径,并扩展出起点到终点的全部 Pareto 最优路径.MCPP 与本文中问题的主要区别在于,MCPP 要找到起点到终点的全部 Pareto 最优路径,而本文中的问题只需找到一条在评分函数 $f(\cdot)$下的最优路径即可.显然,找到全部 Pareto 最优路径之后再计算函数 $f(\cdot)$下的最优路径的效率十分低下,因为绝大部分 Pareto 最优路径并非函数 $f(\cdot)$下的最优路径,所以这些路径无需被计算出来.另一方面,这些方法只是

从理论上给出计算 Pareto 最优路径的方法，并没有从算法实际运行效率的角度设计算法和优化算法，也没有采用数据处理的优化技术.因此，这些方法需要承受较高的时间和空间开销.

本文的主要贡献如下：我们首先给出了一个有效的 best-first search 分支界限算法，该算法可以裁剪掉大量不是最优结果的路径.我们同时给出了两个优化剪枝策略以及快速计算和更新阈值的策略，这些策略均有效地提高了算法的效率.进一步地，我们给出一个过滤算法，该算法可以从图中过滤掉大部分不属于最优路径的顶点，从而缩小图的规模.最后，我们通过真实数据集上的实验验证了算法的有效性.

本文第 2 节介绍多维代价图上最优路径查询问题的定义以及面临的挑战性问题；第 3 节给出 best-first search 分支界限算法、剪枝策略以及快速计算和更新阈值的方法；第 4 节给出顶点过滤算法；第 5 节通过真实数据验证算法的有效性；第 6 节讨论相关工作；最后，我们总结本文.

2　问题定义及面临的挑战性问题

在本节中，我们将介绍多维代价图模型以及在该图模型中最优路径查询问题的定义.

2.1　多维代价图的概念

多维代价图是一个有向图，记为 $G=(V,E)$，其中 V 是图上的顶点集合，E 是图上的边集合.E 中每条边 e 记为 $e=(u,v)$，$u,v\in V$，e 被称为 u 的出边或者 v 的入边，v 称作 u 的出边邻居，u 称作 v 的入边邻居.每条边 e 被赋予一组 d 维代价向量 $\boldsymbol{cost}(e)$，$\boldsymbol{cost}(e)=(c_1,c_2,\cdots,c_d)$，表示通过该边所花费的 d 种代价，其中 $c_i(i=1,2,\cdots,d)$ 为通过该边所花费的第 i 种代价的取值.举例说明，在公路网中，城市 A 与城市 B 之间存在一条有向边 e，代表 A 通过 e 直接可达 B.$\boldsymbol{cost}(e)=(c_1,c_2,c_3)$ 为边 e 上的一组三维代价向量，其中，c_1 表示 A 与 B 之间的欧氏距离，c_2 表示从 A 到 B 所花费的时间，c_3 表示从 A 到 B 所要花费的费用.在本文中，我们合理假设 c_i 取值非负，因为现实世界中不存在小于零的代价.我们的方法可以方便地扩展到无向图上，对一条无向边 $e=(u,v)$，可等价地看作两条有向边 $e_1=(u,v)$ 和 $e_2=(v,u)$，且 $\boldsymbol{cost}(e_1)=\boldsymbol{cost}(e_2)=\boldsymbol{cost}(e)$.简单起见，本文只针对有向图进行讨论.

一条路径 p 是一组由图上顶点所构成的序列，$p=(v_0,v_1,\cdots,v_l)$，其中，$v_i\in V(0\leq i\leq l)$ 并且 $e_i=(v_{i-1},v_i)\in E(0<i\leq l)$.路径 p 被称作是简单的，当且仅当 p 上的顶点没有重复出现，即对任意的 $0\leq i,j\leq l$，满足 $v_i\neq v_j$.路径 p 的代价 $\boldsymbol{cost}(p)$ 为 p 所经过全部边的代价的矢量和.令 $\{e_1,e_2,\cdots,e_l\}$ 为路径 p 所经过的全部边的集合，则路径 p 的代价为 $\boldsymbol{cost}(p)=(c_1(p),c_2(p),\cdots,c_d(p))$，$\boldsymbol{cost}(p)$ 是所有 $\boldsymbol{cost}(e_j)$ 的矢量和，即 $\boldsymbol{cost}(p)=\sum_{j=1}^{l}\boldsymbol{cost}(e_j)$.其中，$c_i(p)=\sum_{j=1}^{l}c_i(e_j),1\leq i\leq d$.这里，$c_i(p)$ 和 $c_i(e_j)$ 分别为路径 p 和边 e_j 在代价向量 $\boldsymbol{cost}(p)$ 和 $\boldsymbol{cost}(e_j)$ 中第 i 个维度上的取值.

评分函数 $f(\cdot)$ 是用户在多维数据空间中指定的一个聚集函数，它反映了用户对不同数据对象喜好程度的得分.$f(\cdot)$ 将数据对象在各个维度上的取值聚合成一个值，通过每个数据的得分情况，返回给用户一个最满意的结果.一般地，分值越小，用户满意度越高.本文中，$f(\cdot)$ 根据 $\boldsymbol{cost}(p)$ 计算路径 p 的得分.我们假设评分函数 $f(\cdot)$ 是单调递增的，即对于两条不同的路径 p 和 p'，如果对 $\forall i,1\leq i\leq d$，都可满足 $c_i(p)\leq c_i(p')$，并且 $\exists i,1\leq i\leq d$，可满足 $c_i(p)<c_i(p')$，则有 $f(p)=f(c_1(p),c_2(p),\cdots,c_d(p))<f(c_1(p'),c_2(p'),\cdots,c_d(p'))=f(p')$.这里，$f(p)$ 等同于 $f(\boldsymbol{cost}(p))$，方便起见，我们在下文中将统一记为 $f(p)$.评分函数的单调性是评分函数的一个普遍性质，并具有其合理性[12].其直观意义为：如果在各个维度上，数据对象 p 的代价都不大于另一对象 p'，则对 p 的评价应不差于 p'.下面，我们给出多维代价图上基于函数 $f(\cdot)$ 的最优路径的定义.

定义 1.　基于函数 $f(\cdot)$ 的最优路径.给定多维代价图 $G(V,E)$ 及评分函数 $f(\cdot)$，$s,t\in V$ 是图上任意两点，令 $P_{s,t}$ 表示 s 到 t 的所有路径的集合.s 到 t 之间基于函数 $f(\cdot)$ 的最优路径，记为 $sp(s,t)$，定义为 $P_{s,t}$ 中具有最小函数值 $f(\cdot)$ 的路径，即对 $\forall p\in P_{s,t}$，有 $f(sp(s,t))\leq f(p)$.

多维代价图上最优路径查询问题的定义为：

输入：多维代价图 G，起点 s 和终点 t，函数 $f(\cdot)$；

输出：基于函数 $f(\cdot)$ 的 s 到 t 的最优路径 $sp(s,t)$.

如图 1 所示，在本例中，我们选取评分函数为 $f(x,y)=x+y$.考虑路径 $p:s\rightarrow d\rightarrow t$，其代价向量

为 $\boldsymbol{cost}(p)=(10,4)$，综合代价得分为 $f(p)=14$. 因此，路径 p 在所有 s 到 t 的路径中综合代价最小，则路径 p 为 s 到 t 的最优路径 $sp(s,t)$.

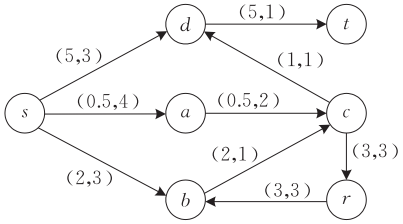


图 1 多维代价图 G

2.2 挑战性问题

几乎目前的所有工作都是基于 Dijkstra 算法的改进，Dijkstra 算法的思想是利用了最短路径的子路径最优的性质. 我们以 Dijkstra 算法为例，说明最短路径的子路径最优的性质在多维代价图中并不成立. 因此，现有的计算最短路径的方法不能直接应用到本文所要解决的问题.

若评分函数 $f(\cdot)$ 是线性的，即对任意边 e_i,e_j，满足 $f(e_i+e_j)=f(e_i)+f(e_j)$，则可将 Dijkstra 算法扩展到本文所要解决的问题. 对图中的每条边 e，先对 e 上的 d 维代价向量计算得分 $f(e)$，并将 $f(e)$ 记为边 e 上的权重. 根据权重 $f(e)$，在图 G 上应用 Dijkstra 算法所找到的最短路径 p 即为本问题的解. 否则，存在另一条路径 p'，使得 $f(p')<f(p)$. 根据函数 $f(\cdot)$ 的线性性质，得到 $f(p')=f(\sum_{i=1}^{l}e'_i)=\sum_{i=1}^{l}f(e'_i)<f(p)=f(\sum_{i=1}^{r}e_i)=\sum_{i=1}^{r}f(e_i)$，该结果与 Dijkstra 算法正确性相矛盾.

若评分函数 $f(\cdot)$ 是非线性的，即函数 $f(\cdot)$ 满足 $f(e_i+e_j)\neq f(e_i)+f(e_j)$，则基于 Dijkstra 算法思想的方法不可应用到本问题中. 在图 1 所示例子中，我们选取 $f(x,y)=x^2+y^2$ 为评分函数. 显然，该函数在 $\{x\geq 0,y\geq 0\}$ 的区域内满足单调递增的性质. 我们的目标为计算出基于函数 $f(x,y)$ 的由 s 到 r 的最优路径. 我们在该图上运行 Dijkstra 算法. 当计算到顶点 c 的代价时，发现路径 $p:s\rightarrow a\rightarrow c$ 的代价为 $f(1,6)=37$，大于路径 $p':s\rightarrow b\rightarrow c$ 的代价 $f(4,4)=32$. 因此，Dijkstra 算法保存 p' 为 s 到 c 的最优路径. 当计算到顶点 r 的最优路径时，因为顶点 r 只连接一条由 c 出发的入边 (c,r)，所以 Dijkstra 算法计算 s 到 r 的最优路径 $sp'(s,r)$ 为 p' 和 (c,r) 的连接，即 $sp'(s,r)=p'+(c,r)=(4,4)+(3,3)=(7,7)$，其得分为 98. 然而，正确结果 $sp(s,r)$ 应为 $s\rightarrow a\rightarrow c\rightarrow r$，得分为 $f(4,9)=97<f(sp'(s,r))$. 原因是多维代价图上并不具有最短路径的子路径最优的性质. 显然，顶点 s 到顶点 r 的最优路径 $s\rightarrow a\rightarrow c\rightarrow r$ 中的子路径 $s\rightarrow a\rightarrow c$ 并不是顶点 s 到顶点 c 的最优路径.

此外，Dijkstra 算法不更新已访问过顶点的代价，这导致无法在图 G 上找到基于非线性函数的最优结果. 同样在图 1 所示例子中，我们的目标为计算 s 到 t 的基于函数 $f(x,y)=x^2+y^2$ 的最优路径. 我们发现 s 到 d 的代价得分小于 s 到 c 的代价得分，因此 Dijkstra 算法优先访问顶点 d. 当访问顶点 c 时，因为顶点 d 已访问过，算法不会更新 s 到 d 的代价，所以 Dijkstra 算法计算出 s 到 t 的最优路径为 $s\rightarrow d\rightarrow t$，其得分为 $f(10,4)=116$. 实际上，s 到 t 的最优路径为 $s\rightarrow a\rightarrow c\rightarrow d\rightarrow t$，其得分 $f(7,8)=113<f(10,4)$.

现有方法都利用了 Dijkstra 算法的核心思想，即最短路径的子路径最优性质. 因此，它们都无法解决本文要面对的问题. 枚举法是计算基于非线性函数 $f(\cdot)$ 最优路径的一种直接方法：给定起点 s 和终点 t，首先找到 s 到 t 的全部路径并分别计算其得分，最后返回给用户得分最小的路径. 设 λ 为图 G 的最大出度，即 $\lambda=\max\{d^+(v)\mid v\in V\}$，其中 $d^+(v)$ 为顶点 v 的出度，则该方法的搜索空间为 $O(\lambda^n)$，n 为图中顶点个数. 显然，该方法不可行. 另外一种方法为预计算：我们先预计算得到图上任意两点之间的最优路径，当一个查询到来，可以在常数时间内将结果返回给用户. 该方法的主要问题是无法应对用户给出不同的评分函数 $f(\cdot)$. 由于用户给出的评分函数千差万别，对某个 $f(\cdot)$ 预计算出的最优路径未必是在另一函数 $f'(\cdot)$ 下的最优路径.

多维代价图上的多标准 Pareto 最优路径计算(MCPP)问题在运筹学领域被广泛研究. 给定起点 s 和终点 t，MCPP 目的是找到 s 到 t 的互不支配的全部路径，即 Pareto 最优路径. 显然，一条 Pareto 最优路径即是一条 skyline 路径. 解决 MCPP 问题的主要方法为标签法. 其主要思想是：用一个优先队列维护图中顶点，并且对图中每个顶点 v 维护一组标签 $list(v)$，$list(v)$ 记录了起点 s 到 v 的 Pareto 最优路径. 当一个顶点 u 从队列弹出后，扩展其邻居顶点 v，并根据 $list(u)$ 和边 (u,v) 更新 $list(v)$. 当 $list(v)$ 支配队列中所有顶点 u 的 $list(u)$ 时，算法结束. 这些方法的主要问题在于：(1) 这些方法基于 Dijkstra 算法的思想，即 Pareto 最优路径上的子路径也一定是 Pareto 最优路径. 因此，若依据路径 p 的函数值

$f(p)$来扩展优先队列中的顶点，会遇到与 Dijkstra 算法计算最优路径相同的问题，即无法找到最优路径.(2)在计算起点 s 到终点 t 所有 Pareto 最优路径的过程中，对所有 Pareto 最优路径经过的所有顶点 v，这些方法都需要计算并维护起点 s 到顶点 v 的 Pareto 最优路径. 显然，这导致了极高的时间开销和空间开销.(3)函数 $f(\cdot)$下的最优路径一定是一条 Pareto 最优路径，但一条 Pareto 最优路径未必是函数 $f(\cdot)$下的最优路径. 因此，找到全部 Pareto 路径之后再计算函数 $f(\cdot)$下的最优路径的效率十分低下，因为绝大部分 Pareto 最优路径并非函数 $f(\cdot)$下的最优路径，所以这些路径无需被计算并且被维护起来.(4)这些方法大多数用于解决 2 维代价图上的 MCPP 问题，在高维度下，这些方法具有极低的效率甚至无法运行. 相反地，本文提出的算法，在高维度下具有十分优秀的效率且随着维度的增高，算法效率不会受到明显的影响. 此外，这些方法只是从理论上给出计算 Pareto 最优路径的方法，并没有从提高算法实际运行效率的角度来设计算法和优化算法，也没有用到数据处理中的优化技术. 因此，这些方法需要承受较高的时间开销和空间开销.

下文中，我们将给出一种有效的 best-first search 分支界限算法，用于搜索非线性评分函数下起点到终点的最优路径. 同时，我们给出 3 种优化策略，这些策略可以显著地提高算法效率. 最后，我们给出一个顶点过滤算法，该算法可以过滤掉图中大部分不属于最优路径的顶点.

3　分支界限算法

在本节中，我们首先提出一个 best-first search 分支界限算法，该算法可以有效地裁剪掉大量不是最优解的路径，从而加速对非线性评分函数下的最优路径搜索. 我们首先给出一个基本的分支界限算法，然后介绍提高算法效率的优化策略.

3.1　基本分支界限算法

给定图 $G(V,E)$，图中所有由起点 s 出发的路径可以组织成一棵搜索树，其根节点是起点集合 $\{s\}$，任意一个非根节点代表一条从点 s 出发的路径(顶点序列). 设 C 和 C' 为搜索树上的两个节点，它们分别代表两条不同的路径. 节点 C 是节点 C' 的父亲节点，当且仅当它们满足以下条件：(i) $C\sqsubset C'$ 且 $|C'|=|C|+1$；(ii) $C'\backslash C$ 中的唯一顶点 v 满足 $v\notin C$ 且 $v\in N^+(u)$，u 为路径 C 的终点，$N^+(u)$ 表示 u 的出边邻居集合. 这里，$C\sqsubset C'$ 表示 C 是 C' 的路径前缀，$|C|$ 表示路径 C 上的顶点个数，$v\notin C$ 保证了将 v 加入 C 不会出现环路，即 C' 是简单的. 因此，起点 s 到终点 t 基于函数 $f(\cdot)$的最优路径查询问题，转化为树搜索问题：即在树上找到一个节点 C，其所代表路径的终点为 t，使得 $f(C)$小于等于任何其他以 t 为终点路径 C'的评分 $f(C')$. 下文中，我们直接用 C 来指代 C 所代表的路径和相应的代价向量. 图 2 展示了图 1 例子中 s 到 t 路径的搜索树.

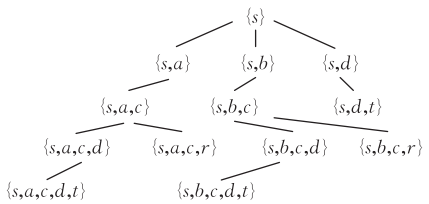


图 2　s 到 t 搜索树示例

我们利用一个最小堆 H 来维护搜索树上待搜索的节点，堆中的节点根据他们在函数 $f(\cdot)$下的得分排序. 我们初始化堆 H 为只包括起点 s 的路径 $\{s\}$. 我们的算法在搜索树上执行 best-first 搜索，即每次弹出堆顶路径 C，$f(C)$取值在堆中最小. 我们用 τ 表示目前已探索到的 s 到 t 最优路径的得分，初始化时，$\tau=\infty$.

若路径 C 的评分 $f(C)>\tau$，则搜索树上以 C 为根的子树可以被安全剪除，即对以 C 为根的子树上的任意一个节点 C'，都有 $f(C')>\tau$. 引理 1 保证了该剪枝策略的正确性.

引理 1.　对于搜索树上的任意两个节点 C 和 C'，如果 C 是 C'的祖先，则有 $f(C')\geq f(C)$.

证明.　若 C 是 C'的祖先，由搜索树的定义，则 C 是 C'的路径前缀. 根据函数 $f(\cdot)$的单调性，可以得到 $f(C')\geq f(C)$.　　证毕.

当路径 C 从最小堆 H 弹出时，我们假设路径 C 的终点为 u. 对 u 的任一以出边相连的邻居 $v\in N^+(u)$，当 $v\neq t$ 时，我们首先判断将 v 添加进路径 C 是否会形成环路，即是否 $v\in C$. 若 $v\notin C$，我们计算新路径 $C'=C\cup\{v\}$在函数 $f(\cdot)$下的得分. 路径 C'的代价向量为 $\boldsymbol{cost}(C')=\boldsymbol{cost}(C)+\boldsymbol{cost}(u,v)$，这里 $\boldsymbol{cost}(u,v)$为边 (u,v)的代价向量. $\boldsymbol{cost}(C)$已知，否则因无法计算 $f(C)$，C 不会被加入堆 H. C'是 C 的孩子，若 $f(C')\geq\tau$，根据引理 1，我们可将以 C'为根的子树剪掉，否则我们将 C'加入堆 H 以待搜索. 当 $v=t$ 时，C'是一条终点为 t 的路径. 我们根据 $f(C')$更新 τ 的取值. 若 $f(C')<\tau$，我们用路径 C'代

替目前找到的终点为 t 的最优路径，并根据 $f(C')$ 更新 τ 的取值. 当 $v=t$ 时，算法不会将 C' 加入堆 H. 当最小堆 $H=\varnothing$，或者堆顶元素 C 的评分 $f(C)\geq\tau$ 时，算法停止. 显然，对堆中任意节点 C'，有 $f(C')\geq f(C)$，而以 C' 为根子树中的任意节点 C''，有 $f(C'')\geq f(C')$. 此时，τ 对应的路径即为起点 s 到终点 t 的最优路径.

3.2 优化剪枝策略

本小节给出两种优化剪枝策略，在给出剪枝策略之前，我们首先介绍 skyline 路径的概念.

定义 2. 路径支配. 给定多维代价图 $G(V,E)$，令 p 和 p' 是 G 上两条不同的路径，且路径 p 和路径 p' 的代价向量分别为 $\boldsymbol{cost}(p)=(c_1(p),c_2(p),\cdots,c_d(p))$ 和 $\boldsymbol{cost}(p')=(c_1(p'),c_2(p'),\cdots,c_d(p'))$. 我们称 p 支配 p'，记作 $p\prec p'$，当且仅当对 $\forall i,1\leq i\leq d$，有 $c_i(p)\leq c_i(p')$，且 $\exists i,1\leq i\leq d$，有 $c_i(p)<c_i(p')$. 这里，$c_i(p)$ 和 $c_i(p')$ 为 $\boldsymbol{cost}(p)$ 和 $\boldsymbol{cost}(p')$ 中第 i 维的代价取值.

定义 3. skyline 路径. 给定多维代价图 $G(V,E)$ 和图上两点 u,v. $P_{u,v}$ 为 u 到 v 所有路径的集合. u 到 v 的 skyline 路径集合，记为 $SKYP_{u,v}$，定义为所有不被 $P_{u,v}$ 中其他路径支配的 u 到 v 路径的集合，即对 $\forall p\in SKYP_{u,v}$，不存在 $p'\in P_{u,v}$，使得 $p'\prec p$. 显然 $SKYP_{u,v}\subseteq P_{u,v}$.

基于 skyline 路径的剪枝规则：在搜索以 $\{s\}$ 为根的搜索树的过程中，对图中每一个顶点 u 维护一个 s 到 u 的已探索到的 skyline 路径集合 $SKYP_{s,u}$. 初始化阶段，$SKYP_{s,u}=\varnothing$. 给定节点 C，C 所表示路径的终点为 u，若 $\exists p\in SKYP_{s,u}$，使得 $p\prec C$，则以 C 为根的子树可以被安全的剪除. 否则，将路径 C 加入集合 $SKYP_{s,u}$. 同时，若 $\exists p\in SKYP_{s,u}$，$C\prec p$，则可将 p 从 $SKYP_{s,u}$ 中删除. 引理 2 保证了该剪枝规则的正确性.

引理 2. 设 C 和 C' 是搜索树上两个不同的节点，且它们所代表路径的终点都为 u. 若 $C'\prec C$，则起点 s 到终点 t 的最优路径不可能出现在以 C 为根的子树中.

证明. 不失一般性，设 $\widetilde{C}$ 为在 C 为根的子树中终点为 t 的路径，我们只需证明存在另外一条 s 到 t 的路径 p，满足 $f(p)<f(\widetilde{C})$. 由搜索树定义，得知 C 为 $\widetilde{C}$ 的路径前缀. 令 $\gamma=\widetilde{C}\backslash C$ 表示 $\widetilde{C}$ 中去掉前缀 C 后的子路径，γ 的起点为 u. 若 $\exists v\in\gamma\backslash\{u\}$，满足 $v\in C'$，则我们可连接 C' 和 γ 得到路径 p（因为 C' 的终点和 γ 的起点都为 u），且 p 中无环路. 因为 $C'\prec C$，所以 $C'+\gamma\prec C+\gamma$，即 $p\prec\widetilde{C}$. 由 $f(\cdot)$ 的单调性，可知 $f(p)<f(\widetilde{C})$.

若 $\exists v\in\gamma\backslash\{u\}$，满足 $v\in C'$，令 v 为路径 γ 中最后一个出现在 C' 的顶点. 我们考虑 C' 的路径前缀 p'，p' 的终点为 v，则 $p'\prec C'$. 又因 $C'\prec C$，有 $p'\prec C$. 令 γ' 为 γ 的路径后缀，γ' 的起点为 v. 我们连接 p' 和 γ' 得到路径 p，p 中无环路. 因为 $p=p'+\gamma'\prec C+\gamma=\widetilde{C}$，所以有 $f(p)<f(\widetilde{C})$. 证毕.

在介绍第 2 个剪枝策略前，我们先介绍最优路径代价下界的定义.

定义 4. 最优路径代价下界. 给定多维代价图 $G(V,E)$，图 G 上的每条边 e 具有 d 维代价向量 $\boldsymbol{cost}(e)$，这里，$\boldsymbol{cost}(e)=(c_1(e),c_2(e),\cdots c_d(e))$. $\mathcal{G}_1$，$\mathcal{G}_2$，$\cdots$，$\mathcal{G}_d$ 为 d 张单代价加权图，$\mathcal{G}_i=(V,E)$ 称为基于图 G 上第 i 维代价的加权图，$\mathcal{G}_i$ 中任意一条边 e 的权值为 $c_i(e)$. 对于图 G 中任意两点 $u,v\in V$，我们称 $\mathcal{P}=\{\mathcal{P}_1,\mathcal{P}_2,\cdots,\mathcal{P}_d\}$ 为 u 到 v 的单代价最短路径集合，其中 $\mathcal{P}_i$ 为 $\mathcal{G}_i$ 上 u 到 v 的加权最短路径，路径 $\mathcal{P}_i$ 的代价为 φ_i. 我们称代价向量 $\boldsymbol{\Phi}_{u,v}=(\varphi_1,\varphi_2,\cdots,\varphi_d)$ 为多维代价图 G 上 u 到 v 的最优路径代价下界.

对图 G 上 u 到 v 的任意一条路径 $p\in P_{u,v}$，令其代价向量为 $\boldsymbol{cost}(p)=(c_1(p),c_2(p),\cdots,c_d(p))$，则我们有 $\boldsymbol{\Phi}_{u,v}\circ p$，即对 $\forall i,1\leq i\leq d$，有 $\varphi_i\leq c_i(p)$.

引理 3 说明 $\boldsymbol{\Phi}_{u,v}$ 是图 G 上 u 到 v 最优路径的一个严格下界.

引理 3. $\boldsymbol{\Phi}_{u,v}$ 是图 G 上 u 到 v 的最优路径的一个严格下界，即不存在另一个下界 $\boldsymbol{\Phi}'_{u,v}$，对 $\forall p\in P_{u,v}$ 和 $\boldsymbol{\Phi}_{u,v}$，有 $\boldsymbol{\Phi}'_{u,v}\circ p\wedge\boldsymbol{\Phi}_{u,v}\prec\boldsymbol{\Phi}'_{u,v}$.

证明. 反证法. 假设存在一个 $\boldsymbol{\Phi}'_{u,v}$，其满足 $\boldsymbol{\Phi}_{u,v}\prec\boldsymbol{\Phi}'_{u,v}$，则 $\exists i(1\leq i\leq d)$，使得 $\varphi'_i>\varphi_i$. 另一方面，因为 $\mathcal{P}_i\in\mathcal{P}$ 和 $\boldsymbol{\Phi}'_{u,v}\circ\mathcal{P}_i$，所以有 $\varphi'_i\leq\varphi_i$. 矛盾. 证毕.

基于最优路径代价下界的剪枝规则：我们预计算出图 G 上任意两点 u,v 之间的最优路径代价下界 $\boldsymbol{\Phi}_{u,v}$. 给定节点 C，C 的终点为 u，我们根据 u 到 t 的最优路径代价下界估计一个下界 $LB(C)$，$LB(C)=f(C+\boldsymbol{\Phi}_{u,t})$，$LB(C)$ 表示在以 C 为根的子树中终点为 t 的路径的得分下界. 若 $LB(C)\geq\tau$，以 C 为根的子树可以被安全地剪除，引理 4 保证了该剪枝规则的正确性.

引理 4. C 是搜索树上的节点且 $LB(C)\geq\tau$，不失一般性，令 $\widetilde{C}$ 为在 C 为根的子树中终点为 t 的路径，则 $f(\widetilde{C})\geq\tau$.

证明. C 所代表路径的终点为 u，C 为 $\widetilde{C}$ 的路径前缀，则 $\gamma=\widetilde{C}\backslash C$ 表示 $\widetilde{C}$ 中去掉前缀 C 后的子路

径且 γ 的起点为 u. γ 是 u 到 t 的一条路径. 根据 u 到 t 的最优路径代价下界的定义，我们有 $\boldsymbol{\Phi}_{u,t}\circ\gamma$. 因此，$C+\boldsymbol{\Phi}_{u,t}\circ C+\gamma$. 根据函数 $f(\cdot)$ 的单调性，$f(\widetilde{C})=f(C+\gamma)\geq f(C+\boldsymbol{\Phi}_{u,t})=LB(C)\geq\tau$. 证毕.

算法 1 给出了 best-first search 分支界限法计算最优路径的流程.

实际上，图中会存在大量不属于最优路径的顶点，探索这些顶点会带来不必要的时间开销. 将这些顶点从图中删除，不会影响查询结果的正确性. 因此，如何判定一个顶点与最优路径无关，成为了一个关键的问题. 在第 4 节，我们介绍一种快速顶点过滤技术，可以有效地从图中过滤掉大部分不属于最优路径的顶点，从而缩减图的规模.

3.3　快速计算和更新阈值

在搜索过程中，阈值 τ 是一个关键的参数. 显然，τ 的取值越小，其剪枝能力越强. 然而，在算法开始阶段，τ 被初始化为 ∞，直到探索到一个终点为 t 的路径 C 时，裁剪规则才开始生效. 因此，裁剪规则在算法开始阶段的裁剪能力非常低. 下面，我们将给出一个预先快速计算 τ 以及搜索过程中快速更新 τ 的方法.

给定多维代价图 $G(V,E)$、起点 s 和终点 t，考虑 s 到 t 的单代价最短路径集合 $\mathcal{P}=\{\mathcal{P}_1,\mathcal{P}_2,\cdots,\mathcal{P}_d\}$. 显然，$\mathcal{P}_i$ 也是多维代价图 $G(V,E)$ 上的一条路径. 此外，我们找到 s 到 t 的跳数最少的路径 $\mathcal{P}_{hop}$，即对 $\forall p\in P_{s,t}$，满足 $|\mathcal{P}_{hop}|\leq|p|$，$|p|$ 为 p 中顶点个数. 我们初始化 $\tau=\min\{f(\mathcal{P}_i),f(\mathcal{P}_{hop})\mid 1\leq i\leq d\}$. 显然，最优路径 $sp(s,t)$ 的评分满足如下关系：$f(\boldsymbol{\Phi}_{s,t})\leq f(sp(s,t))\leq\tau$，$f(\boldsymbol{\Phi}_{s,t})$ 为代价向量 $\boldsymbol{\Phi}_{s,t}$ 的得分. 实验证明，该方法给出的初始阈值 τ 具有很强的剪枝能力.

我们同时给出一种在搜索过程中快速更新 τ 的方法. 当路径 C 从最小堆 H 弹出时，假设其终点为 u，同样考虑 u 到 t 的单代价最短路径集合 $\mathcal{P}=\{\mathcal{P}_1,\mathcal{P}_2,\cdots,\mathcal{P}_d\}$. 我们根据 $\mathcal{P}_1,\mathcal{P}_2,\cdots,\mathcal{P}_d$ 计算 C 的一个得分上界 $UB(C)$，$UB(C)=\min\{f(C+\mathcal{P}_i)\mid 1\leq i\leq d\}$. 若 $UB(C)<\tau$，我们可将 τ 更新为 $UB(C)$. 引理 5 保证了该更新规则的正确性.

算法 1.　Find-$sp(s,t)$-Best-First-Search $(G,s,t,f(\cdot))$.

输入：多维代价图 G，起点 s，终点 t，评分函数 $f(\cdot)$

输出：最优路径 $sp(s,t)$

1. $\tau\leftarrow\infty$，$sp(s,t)\leftarrow\varnothing$，$H\leftarrow\{s\}$，$cost(\{s\})\leftarrow 0$;
2. WHILE $H\neq\varnothing$ DO
3. let C be the path by popping up the top element from H and $end(C)=u$;　//$end(C)$为路径 C 的终点
4. IF $f(C)\geq\tau$ THEN
5. BREAK;
6. FOR EACH vertex v, $v\in N^+(u)\wedge v\notin C$ DO
7. $C'\leftarrow C\cup\{v\}$; $f(C')\leftarrow f(C+(u,v))$;
8. IF $v=t$ THEN
9. IF $f(C')<\tau$ THEN
10. $\tau\leftarrow f(C')$; $sp(s,t)\leftarrow C'$; continue;
11. ELSE
12. $LB(C')\leftarrow f(C'+\Phi_{v,t})$;
13. IF $f(C')\geq\tau$ THEN
14. prune the subtree rooted at C'; continue;
15. IF $\exists p\in SKYP_{s,v}, p\prec C'$ THEN
16. prune the subtree rooted at C'; continue;
17. ELSE
18. $SKYP_{s,v}\leftarrow SKYP_{s,v}\cup\{C'\}$;
19. FOR EACH $p\in SKYP_{s,v}$ DO
20. IF $C'\prec p$ THEN
21. $SKYP_{s,v}\leftarrow SKYP_{s,v}-\{p\}$;
22. IF $LB(C')\geq\tau$ THEN
23. prune the subtree rooted at C'; continue;
24. insert C' into H according to $f(C')$;
25. RETURN $sp(s,t)$, τ

引理 5.　设 C 是搜索树上的一个节点，C 所代表路径的终点为 u，C 的最优路径得分上界为 $UB(C)$，则存在一条 s 到 t 路径的 p，$f(p)\leq UB(C)$.

证明.　设 $\mathcal{P}_{\min}$ 为使得 $f(C+\mathcal{P}_i)(1\leq i\leq d)$ 取值最小的路径 $\mathcal{P}_i$，则 $\mathcal{P}_{\min}$ 是 u 到 t 的一条路径. 若 $\mathcal{P}_{\min}\bigcap C=\{u\}$，则连接 C 和 $\mathcal{P}_{\min}$ 得到一条路径 p，其代价得分即为 $UB(C)$. 若 $\mathcal{P}_{\min}\bigcap C\neq\{u\}$，则 $\{u\}\subset\mathcal{P}_{\min}\bigcap C$. 令 v 是路径 $\mathcal{P}_{\min}$ 中最后一个出现在 C 中的顶点，即对 $\forall v'\in\mathcal{P}_{\min}\bigcap C$，$v'$ 在 $\mathcal{P}_{\min}$ 中位于 v 之前. 令 C' 为 C 中的终点为 v 的路径前缀，$\mathcal{P}'_{\min}$ 为 $\mathcal{P}_{\min}$ 中起点为 v 的路径后缀. 我们连接 C' 和 $\mathcal{P}'_{\min}$ 得到新路径 p，p 中无环路. 因为 $p\prec C+\mathcal{P}_{\min}$，所以 $f(p)\leq f(C+\mathcal{P}_{\min})\leq UB(C)$. 证毕.

4　顶点过滤算法

本节中，我们提出一个顶点过滤算法，该算法可以快速地从多维代价图 $G(V,E)$ 中过滤掉大部分不属于最优路径的顶点，从而有效地缩减图 $G(V,E)$ 的规模. 我们首先介绍过滤索引，然后给出过滤算法.

算法 2.　Vertex-Filtering $(G,s,t,f(\cdot))$.

输入：多维代价图 G，起点 s，终点 t，评分函数 $f(\cdot)$

输出：最优路径 $sp(s,t)$

1. $\tau \leftarrow \min\{f(\mathcal{P}_i), f(\mathcal{P}_{hop}) \mid 1 \le i \le d\}$;
2. FOR EACH $u \in V$ DO
3. IF $\tau < f(\boldsymbol{\Phi}_{s,u} + \boldsymbol{\Phi}_{u,t})$ THEN
4. $V \leftarrow V - \{u\}$;
5. FIND-$sp(s,t)$-Best-First-Search ($G, s, t, f(\cdot)$);
6. RETURN $sp(s,t), \tau$

4.1 过滤索引

过滤索引是一个 $n \times n$ 的矩阵，n 是图 G 上的顶点个数. 矩阵中的行代表起点，列代表终点. 矩阵中每个元素 $A_{u,v}$ 包括两个部分：(1) 顶点 u 到顶点 v 的最优路径代价下界 $\boldsymbol{\Phi}_{u,v}$；(2) u 到 v 的单代价最短路径集合 $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_d\}$ 和 $\mathcal{P}_{hop}$，$\mathcal{P}_{hop}$ 为 u 到 v 的跳数最少的路径. 过滤索引的空间开销为 $O(n^2)$.

4.2 过滤算法

当查询给定起点 s、终点 t 以及评分函数 $f(\cdot)$ 后，根据过滤索引，可以确定图中哪些顶点能够在本次查询中被安全的删除. 我们首先根据 s 到 t 的单代价最短路径集合和 $\mathcal{P}_{hop}$ 计算 s 到 t 的最优路径得分的一个上界 τ，$\tau = \min\{f(\mathcal{P}_i), f(\mathcal{P}_{hop}) \mid 1 \le i \le d\}$. 若 $\mathcal{P} = \varnothing$，则算法直接返回 s 不可到达 t. 对于一个顶点 u，如果有 $\tau < f(\boldsymbol{\Phi}_{s,u} + \boldsymbol{\Phi}_{u,t})$，则顶点 u 可从图 G 中删去，也就是说，s 到 t 的最优路径一定不会经过顶点 u. 定理 1 保证了该算法的正确性.

定理 1. 给定多维代价图 G、起点 s 和终点 t 以及评分函数 $f(\cdot)$. $\mathcal{P}$ 为过滤索引中 s 到 t 的单代价最短路径集合，$\mathcal{P} \neq \varnothing$. 令 τ 为根据 $\mathcal{P}$ 计算的起点 s 到终点 t 的最优路径的一个得分上界，这里 $\tau = \min\{f(\mathcal{P}_i), f(\mathcal{P}_{hop}) \mid 1 \le i \le d\}$. 对于图 G 中任意顶点 u，如果 $\tau < f(\boldsymbol{\Phi}_{s,u} + \boldsymbol{\Phi}_{u,t})$，$\boldsymbol{\Phi}_{s,u}$ 和 $\boldsymbol{\Phi}_{u,t}$ 分别为 s 到 u 和 u 到 t 的最优路径代价下界，则 s 到 t 的最优路径必定不经过 u.

证明. 只需证明对任何一条经过 u 的路径 p，都存在一条不经过 u 的路径 p'，有 $f(p') < f(p)$. 不失一般性，令 p 是一条经过 u 的路径，p 由两个片段接组成：(1) s 到 u 的片段 $p_{s,u}$；(2) u 到 t 的片段 $p_{u,t}$. 根据最优路径代价下界的定义，我们可知 $\boldsymbol{\Phi}_{s,u} \circ p_{s,u}$ 和 $\boldsymbol{\Phi}_{u,t} \circ p_{u,t}$. 因此，$\boldsymbol{\Phi}_{s,u} + \boldsymbol{\Phi}_{u,t} \circ p$. 由函数 $f(\cdot)$ 的单调性，得到 $f(\boldsymbol{\Phi}_{s,u} + \boldsymbol{\Phi}_{u,t}) \le f(p)$. 令 p' 为 $\mathcal{P} \cup \{\mathcal{P}_{hop}\}$ 中取得最小评分值的路径，即 $f(p') = \tau$. p' 是 s 到 t 的一条路径，p' 不经过 u，否则与 $\tau < f(\boldsymbol{\Phi}_{s,u} + \boldsymbol{\Phi}_{u,t})$ 矛盾. 显然，我们有 $f(p') < f(\boldsymbol{\Phi}_{s,u} + \boldsymbol{\Phi}_{u,t}) \le f(p)$. 证毕.

在图 1 中所示的多维代价图 G 中，我们选取评分函数 $f(x,y) = x^2 + y^2$. 目标为计算出基于函数 $f(x,y)$ 的由 s 到 t 的最优路径. 我们发现路径 $p: s \rightarrow a \rightarrow c \rightarrow d \rightarrow t$ 是 s 到 t 的一条单代价最短路径，且其得分 113 在所有单代价最短路径中最小. 因此，我们选取 $\tau = 113$. 根据定理 1，顶点 b 和顶点 r 以及它们所连接的边可以从图 G 中移除. 过滤后的图 G 只包含顶点 s, a, c, d, t.

算法 2 给出了顶点过滤算法的流程. 顶点过滤算法对图中每一个顶点 u 都要计算 $f(\boldsymbol{\Phi}_{s,u} + \boldsymbol{\Phi}_{u,t})$，该计算过程的时间复杂度为 $O(1)$. 因此，顶点过滤算法的时间复杂性为 $O(n)$. 算法 2 得到新的顶点集合 $\tilde{V} \subseteq V$，$\tilde{V}$ 是 V 中所有不可被过滤掉的顶点集合. 我们计算 $\tilde{V}$ 在 $G(V,E)$ 上的诱导子图 $G(\tilde{V})$，并在 $G(\tilde{V})$ 上调用算法 1. 实验证明，过滤算法具有十分优秀的过滤能力.

5 实验结果和分析

5.1 数据集描述和实验设置

我们在两个真实的图数据集上实现了本文的算法，并与目前计算多标准 Pareto 最优路径的最有效方法进行了比较. 所有实验均在主频为 2.5 GHz 的 Intel Core i5 CPU 和内存为 8 GB 的 PC 机上完成. 我们使用的操作系统为 Windows 7.

我们采用两个真实数据集进行算法性能测试.

California road network：本数据集描述的是美国加利福尼亚州道路交通网，包括 21 047 个顶点和 21 692 条边. 该网络是一个无向图. 本数据来自 http://www.maproom.psu.edu/dcw.

Slashdot dataset：Slashdot 是一个技术新闻网站. 本数据集中，用户代表顶点，顶点 u 到 v 的边代表用户 u 赞同用户 v 的评论. 本数据包括 20 639 个顶点和 87 627 条边. 该网络是一个有向图，本数据来自 http://slashdot.org/.

针对以上数据集中的每一条边，我们随机生成 $d(\in \{2, \cdots, 5\})$ 维代价. 对每一组数据集，我们分别生成规模为 1000 的查询集合，实验中的结果，是每个查询集合的平均值. 我们选取评分函数 $f(x,y) = x^2 + y^2$.

我们在实验中主要考察了以下几方面的内容：(1) 图中代价维度 d 对算法性能的影响；(2) 各个优化策略对算法性能的影响；(3) 过滤算法对算法性能的影响. 算法性能考察的指标为：(1) 算法运行过程中访问的搜索树节点个数；(2) 算法执行时间；(3) 过滤算法过滤的顶点比例和运行时间.

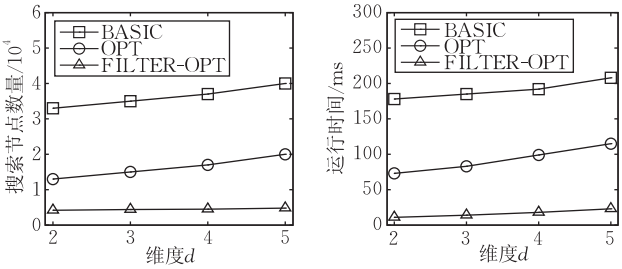
我们与 SHARC 算法进行了比较. SHARC 算

法是目前计算多标准 Pareto 最优路径最有效的方法[10]. 我们先利用 SHARC 计算出起点到终点的全部 Pareto 路径，再分别计算这些 Pareto 最优路径在函数 $f(x,y)$下的评分并找到评分最优的路径.

5.2　实验结果

Exp-1：代价维度对算法性能的影响. 我们考察了搜索树中被算法访问的节点数量和算法运行时间与代价维度 d 的关系. 我们考察的算法有：BASIC、OPT 和 FILTER-OPT. BASIC 是只应用了基本剪枝策略的分支界限法，OPT 是应用了优化剪枝策略以及快速计算和更新阈值策略的分支界限法，而 FILTER-OPT 是应用了顶点过滤算法的 OPT 算法.

图 3 给出了 California road network 上的结果. 图 3(a)中，我们发现搜索节点数量随着 d 的增长而缓慢增长. 其中，OPT 增长较快，这是因为随着维度的增加，基于 skyline 路径的剪枝规则能力变弱. 例如，2 维代价下，路径 p 被 p' 支配，但在 3 维代价下，该支配关系未必成立. FILTER-OPT 增长较小，这是因为过滤算法已经过滤掉大量的顶点. 通过图 3(a)，我们发现 OPT 搜索的节点数量比 BASIC 减少了 50%以上，而 FILTER-OPT 搜索的节点数量比 OPT 减少了 60%以上. 因此，如图 3(b)所示，FILTER-OPT 的运行时间比 OPT 少了 60%以上，而 OPT 的运行时间比 BASIC 少 50%以上.

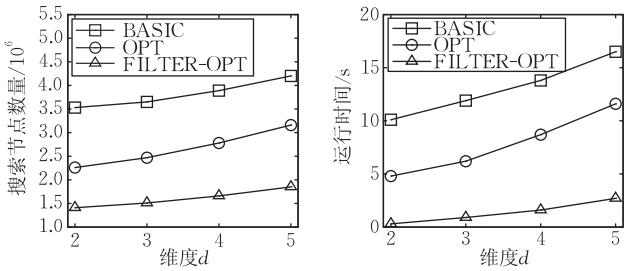


(a) d对搜索节点的影响　　(b) d对运行时间的影响

图 3　代价维度 d 对算法性能的影响(CA)

图 4 给出了 Slashdot dataset 上的结果. 同样地，由于 FILTER-OPT 采取了全部优化技术，它搜索的节点数量最少，运行时间最短，OPT 次之，BASIC 的性能最差.

Exp-2：各个优化策略对算法性能的影响. 在这部分实验中，我们分析了各个优化策略对算法性能的影响. 其中，OPT 是应用了全部优化策略的算法，SP 是指从 OPT 中移除基于 skyline 路径的剪枝规则的算法，LB 是指从 OPT 中移除基于最优路径代价下界剪枝规则的算法，FT 是指从 OPT 中移除快速计算和更新阈值策略的算法. 我们从搜索树上被

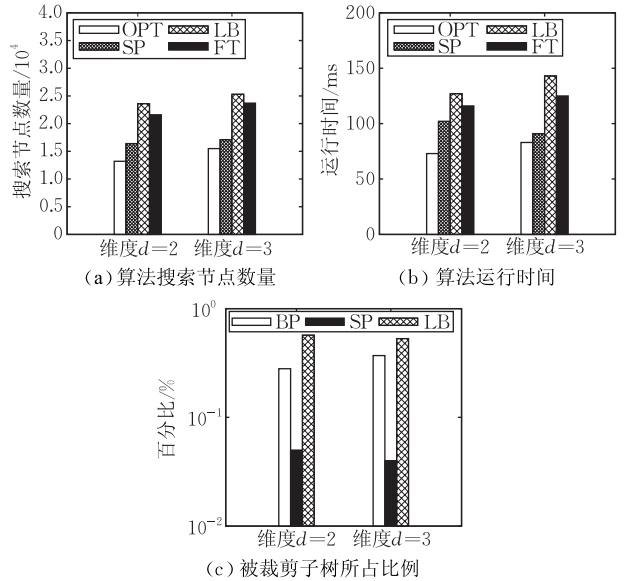


(a) d对搜索节点的影响　　(b) d对运行时间的影响

图 4　代价维度 d 对算法性能的影响(SD)

访问的节点数量和算法运行时间两个方面考察了算法的性能.

图 5 给出了 California road network 上的实验结果. 我们发现 OPT 搜索的节点数量和运行时间比其它算法都要少，这说明各个优化策略都能剪除大量的非最优解的路径. 其中，基于最优路径代价下界的剪枝规则(LB)和快速计算更新阈值的策略(FT)效果最明显. 在图 5(c)中，我们比较了基本剪枝规则(BP)、基于 skyline 路径的剪枝规则(SP)和基于最优路径代价下界剪枝规则(LB)裁剪子树的能力. 我们发现基本剪枝规则(BP)和基于最优路径代价下界(LB)的剪枝规则裁剪子树的能力最强.



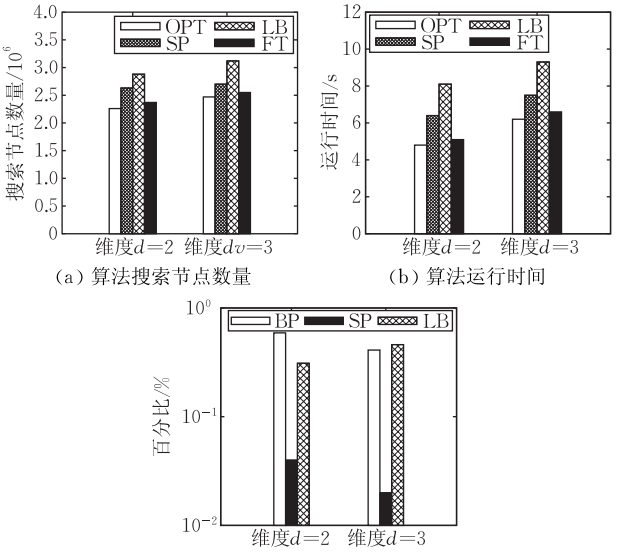
(a) 算法搜索节点数量　　(b) 算法运行时间

(c) 被裁剪子树所占比例

图 5　剪枝策略对算法性能的影响(CA)

图 6 给出了 Slashdot dataset 上的实验结果. 我们同样发现基于最优路径代价下界剪枝规则(LB)的剪枝能力最强. 与 California road network 不同，快速计算和更新阈值的策略(FT)的效果减弱. 这是因为，与 California road network 相比，Slashdot dataset 的密度更高，因此算法可以快速探索到一条起点到终点的路径. 因此，即使在算法开始阶段不计

算出一个初始阈值，也不会对算法性能造成明显的影响. 同样地，在图 6(c)中，我们发现基本剪枝规则(BP)和基于最优路径代价下界(LB)的剪枝规则裁剪子树的能力最强.

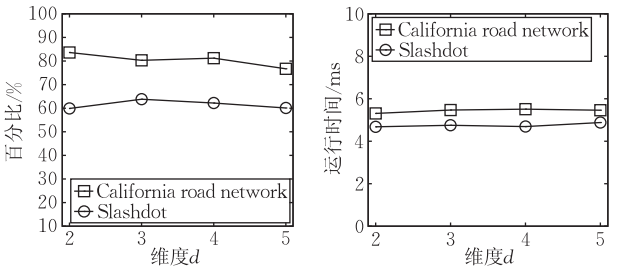


(a) 算法搜索节点数量 (b) 算法运行时间

(c) 被裁剪子树所占比例

图 6 剪枝策略对算法性能的影响(SD)

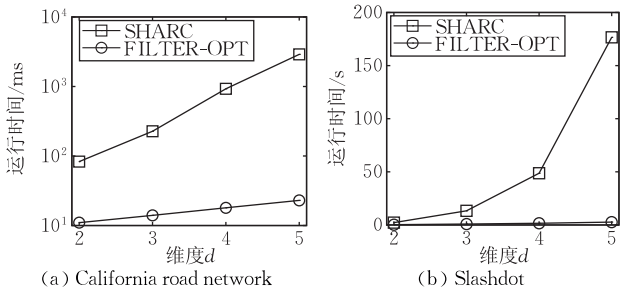
Exp-3：顶点过滤算法的性能. 图 7 给出了顶点过滤算法的效力和效率. 如图 7(a)所示，对 California road network，算法至少可过滤掉 80% 的顶点，对 Slashdot dataset，算法至少可过滤掉 60% 的顶点. 这是因为，Slashdot dataset 的密度较高，所以其过滤效果减低. 我们同时发现，过滤效果与维度 d 无明显关系. 如图 7(b)所示，顶点过滤算法具有十分优秀的时间效率，且其运行时间不受维度 d 影响.



(a) 过滤算法裁剪顶点比例 (b) 过滤算法运行时间

图 7 过滤算法的性能

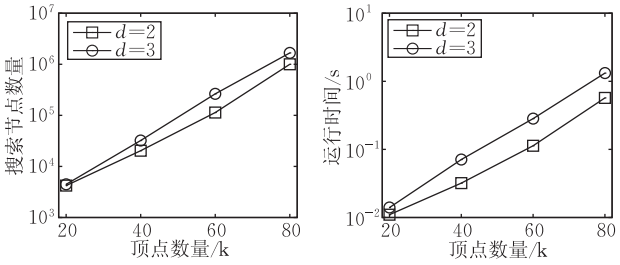
Exp-4：与 SHARC 算法的比较. 我们分别在 California road network 和 Slashdot 上与 SHARC 算法进行了比较. 如图 8(a)所示，在 $d=2$ 时，本文算法比 SHARC 要快 4 倍以上，这是因为 SHARC 要计算出起点到终点的全部 Pareto 最优路径，而本文的算法只需要计算出一条路径. 同时，我们发现 SHARC 的运行时间随着维度 d 的增加而快速增长. 这是因为，随着维度的增加，Pareto 最优路径的数量也急剧增长，因此在高维度下，计算 Pareto 最优路径的方法的效率很低. 相反地，我们的方法不需要找到全部 Pareto 最优路径. 绝大多数 Pareto 路径，在搜索过程中很早地从搜索树中被剪除. 因此，在 $d=5$ 时，我们的算法几乎 100 倍的快于 SHARC 算法. 同样地，如图 8(b)所示，在 Slashdot 数据集上，$d=2$ 时，SHARC 算法与本文算法性能相近，但随着维度增加，SHARC 算法的运行时间急剧增长，而我们算法的运行时间增长十分平缓. 该组实验说明，计算两点间 Pareto 最优路径的方法并不适合于本文所要解决的问题.



(a) California road network (b) Slashdot

图 8 与 SHARC 算法的比较

Exp-5：可扩展性测试. 我们在人工合成数据集上测试了我们算法的可扩展性. 我们生成了 4 个人工合成数据集，其顶点规模分别为 20 k，40 k，60 k 和 80 k. 我们分别在 $d=2$ 和 $d=3$ 的情况下，测试了这 4 个人工合成数据集上算法搜索的节点数量和算法的运行时间. 图 9 给出了可扩展性测试的结果. 我们发现，算法搜索的节点数量(图 9(a))和算法的运行时间(图 9(b))都随着顶点规模的增大而增加. 在顶点规模为 80 k 的情况下，本文算法的运行时间仍然小于 1 s，这说明本文算法在大规模数据集上仍然具有十分高效的效率. 该组实验验证了本文算法具有十分优秀的可扩展性.



(a) 算法搜索节点数量 (b) 算法运行时间

图 9 可扩展性测试

6 相关工作

Dijkstra 算法是最短路径查询的经典算法. 当图的规模很大时，Dijkstra 算法有着较大的时间开销. 为了减小时间开销，现有工作都是先建立索引，预计算索引内部顶点间的最短路径. 当查询到来后，利用索引完成查询. 文献[1]提出了一种最短路径 quad-tree 机制，该方法预计算图中所有可能顶点之间的最短路径并用 quad-tree 将这些路径组织起来. 然而，该方法并不适用于本文所要解决的问题. 因为不同用户所给出的评分函数是不同的，针对某一用户评分函数建立的 quad-tree 并不能正确回答另一用户评分函数下的最优路径查询. 文献[2]提出了压缩 BFS-tree 的概念，压缩 BFS-tree 利用图的对称性建立索引. 文献[3]提出了 TEDI 方法，该方法利用树分解理论建立索引并完成最短路径查询. 文献[4]基于顶点覆盖建立一个树结构索引，该索引能够达到较高的 I/O 效率. 文献[5]考虑了标签限制下的最短路径查询问题. 文献[6]提出了一种依赖查询的 Landmark 机制，该机制可以找到与查询点最接近的局部 Landmark，并利用以该 Landmark 为根的最短路径树来完成查询. 不幸的是，所有这些工作都利用了子路径最优的性质，即最短路径上的子路径也是最短路径. 因此，无论是在建立索引过程中，还是在利用索引完成最短路径查询时，这些方法只需保存索引内部顶点间的最短路径即可，再利用这些最短路径进行连接组合得到结果. 然而，在多维代价图中，最短路径子路径最优的性质并不成立，所以，这些方法都不能解决本文所提出的问题.

多维代价图上的多标准 Pareto 最优路径计算(MCPP)问题在运筹学领域被广泛研究[9-11]. 给定起点 s 和终点 t，MCPP 目的是找到 s 到 t 的互不支配的全部路径，即 Pareto 最优路径. 显然，一条 Pareto 最优路径也是一条 skyline 路径. 解决 MCPP 的方法大多数采用的是标签法[9-10]，该类方法也是基于 Dijkstra 算法的思想，即任意一条 Pareto 最优路径上的子路径也应该为一条 Pareto 最优路径. 后续的工作[11]提出 A^* 搜索方法解决 MCPP 问题. 这些方法需迭代地计算出全部 Pareto 最优路径所要经过全部顶点的 Pareto 最优路径，并据此计算出起点到终点的全部 Pareto 最优路径. MCPP 与本文中定义问题的主要区别在于，MCPP 要找到起点 s 到终点 t 的全部 Pareto 最优路径，而本文定义的问题只需找到一条在评分函数 $f(\cdot)$ 下最优的路径即可. 显然，找到全部 Pareto 最优路径之后再计算函数 $f(\cdot)$ 下的最优路径的效率十分低下，因为绝大部分 Pareto 最优路径并非函数 $f(\cdot)$ 下的最优路径. 此外，这些方法只是从理论上给出计算 Pareto 最优路径的方法，并没有从算法实际运行效率的角度设计算法和优化算法，也没有采用数据处理的优化技术. 因此，这些方法需要承受较高的时间和空间开销.

目前许多工作研究了图上基于最短路径的 skyline 查询问题[13-14]. 文献[13]提出了 MSQ 算法解决动态 skyline 查询问题. 然而，该工作考虑的图模型仍然是单代价图模型. 文献[14]解决了多维代价物流网中 skyline 和 Top-k 查询问题，图中每个顶点与查询点在不同维度上的最短路径代价作为该顶点的代价向量，然后根据该代价向量计算出 skyline 和 Top-k 结果. 然而，该工作中的代价向量定义与本文不同，且查询目标不是路径，因此不能用于解决本文提出的问题.

7 结　论

本文中，我们研究了多维代价图模型下的最优路径查询问题. 我们首先给出了多维代价图模型下，基于函数 $f(\cdot)$ 的最优路径查询的定义. 然后，我们提出了一个 best-first search 分支界限法并给出 3 个优化策略. 同时，我们给出一个顶点过滤算法，该算法可以从图中过滤掉大部分不属于最优路径的顶点. 最后，我们通过真实数据集上的实验验证了算法的有效性.

参 考 文 献

[1] Samet H, Sankaranarayanan J, Alborzi H. Scalable network distance browsing in spatial databases//Proceedings of the ACM SIGMOD International Conference on Management of Data. Vancouver, Canada, 2008: 43-54

[2] Xiao Y, Wu W, Pei J, Wang W, He Z. Efficiently indexing shortest paths by exploiting symmetry in graphs//Proceedings of the 12th International Conference on Extending Database Technology. Saint Petersburg, Russia, 2009: 493-504

[3] Wei F. TEDI: Efficient shortest path query answering on graphs//Proceedings of the ACM SIGMOD International Conference on Management of Data. Indianapolis, USA, 2010: 99-110

[4] Cheng J, Ke Y, Chu S, Cheng C. Efficient processing of distance queries in large graphs: A vertex cover approach//Proceedings of the ACM SIGMOD International Conference on

Management of Data. Scottsdale, USA, 2012
[5] Rice M, Tsotras V J. Graph indexing of road networks for shortest path queries with label restrictions//Proceedings of the 37th International Conference on Very Large Database. Seattle, USA, 2011: 69-81
[6] Qiao M, Cheng H, Chang L, Yu J. Approximate shortest distance computing: A query-dependent local landmark scheme//Proceedings of the 28th IEEE International Conference on Data Engineering. Washington, USA, 2012
[7] Bazaraa M S. Nonlinear Programming: Theory and Algorithms. 2nd Edition. John and Wiley and Sons, 1993
[8] Hiller F S, Lieberman G J. Introduction to mathematical programming. 2nd Edition. McGraw-Hill, 1990
[9] Martins E Q V. On a multicriteria shortest path problem. European Journal of Operational Research, 1984, 16(2): 236-245
[10] Delling D, Wagner D. Pareto paths with SHARC* // Proceedings of the 8th International Symposium on Experimental Algorithms. Dortmund, Germany, 2009: 125-136
[11] Mandow L, Cruz J. A new approach to multiobjective a^* search//Proceedings of the 19th International Joint Conference on Artificial Intelligence. Edinburgh, UK, 2005: 218-223
[12] Chaudhuri S, Gravano L. Evaluating top-k selection queries//Proceedings of the 25th International Conference on Very Large Database. Edinburgh, Scotland, 1999: 397-410
[13] Chen L, Lian X. Dynamic skyline queries in metric spaces// Proceedings of the 12th International Conference on Extending Database Technology. Nantes, France, 2008
[14] Mouratidis K, Lin Y, Yiu M L. Preference queries in large multi-cost transportation networks//Proceedings of the 26th IEEE International Conference on Data Engineering. Long Beach, USA, 2010

YANG Ya-Jun, born in 1983, Ph. D. candidate. His main research interests include graph mining, graph database and massive data management.

GAO Hong, born in 1966, Ph. D., professor, Ph. D. supervisor. Her main research interests include graph mining, graph database, massive data management and wireless sensor networks.

LI Jian-Zhong, born in 1950, professor, Ph. D. supervisor. His current research interests include graph mining, graph database, massive data management and wireless sensor networks.

Background

Graphs have been widely used to model complex relationships among various entities in many real-life applications. Shortest path query is an important problem in graphs and has been well-studied. However, most approaches for shortest path are based on single-cost (weight) graphs. In this paper, we introduce the definition of multi-cost graph and propose a new query: the optimal path query based on function over multi-cost graphs. Most existing methods utilize the property of the optimal sub-path in shortest path: any sub-path of a shortest path is also a shortest path. Unfortunately, the optimal sub-path property doesn't hold in multi-cost graphs. To the best of our knowledge, our work is the first paper to address the optimal path query based on function over multi-cost graphs. We propose a novel best-first search branch and bound algorithm with three optimizing strategies. We also propose a vertex filtering algorithm to filter a large proportion of vertices from graph.

This work is supported in part by the State Key Development Program of Basic Research of China, the Key Problem of National Natural Science Foundation of China, the NSF of China and the NSFC-RGC of China. These foundations focus on the research of various areas of data intensive computing. Our group has been working on the research of database for many years, and many high quality papers have been published in worldwide conferences and transactions, such as SIGMOD, VLDB, ICDE, KDD, INFOCOM, TKDE, VLDB Journal et al.