最多叶子生成树问题的核化算法

高文宇
(广东商学院信息学院 广州 510320)

摘 要 对算法领域的最多叶子生成树问题进行了深入研究,提出了对简单连通图 2 度节点的化简规则,并证明了不含 2 度节点的图的生成树的叶子节点数的下限为$(N+6)/4$,给出了构造这样一棵生成树的构造性方法.基于上述化简规则和所证明的结论,给出了最多叶子生成树问题的核化算法,该核化算法可以在$O(n^2)$时间内得到一个$4k-6$大小的线性核.对于这样一个较小的核,将大大提高相关的参数算法和近似算法的性能.

关键词 最多叶子生成树;核化;参数算法
中图法分类号 TP301 **DOI 号**: 10.3724/SP.J.1016.2010.02211

Kernelization Algorithm of Maximum Leaf Spanning Tree

GAO Wen-Yu
(*School of Computer Science, Guangdong University of Business Studies, Guangzhou* 510320)

Abstract This paper proposes 2-degree node reduction rules for Maximum Leaf Spanning Tree problem, and proves that any connected graph without 2-degree node has a spanning tree including at least $(N+6)/4$ leafs. The reduction rules and the low bound are used to construct a kernelization algorithm for Maximum Leaf Spanning Tree problem, which algorithm can output a $4k-6$ linear kernel of Maximum Leaf Spanning Tree problem in $O(n^2)$ time.

Keywords maximum leaf spanning tree; kernelization; parameterized algorithm

1 引 言

最多叶子生成树问题(Maximum Leaf Spanning Tree,MLST)是 NP 完全理论中的一个经典问题.在文献[1]中,将 MLST 归为网络设计领域最重要的 NP 完全问题之一.简单来说,MLST 问题就是在一个给定的图中是否存在一棵生成树,使得该生成树有大于等于 k 个叶子节点(叶子节点指生成树中度为 1 的节点).MLST 在通信网络和电路布局中有着很强的应用背景.然而 MLST 是 NP 难问题,而且是 MAX SNP-hard[2],即 $\exists\varepsilon>0$,不存在对 MLST 问题的 $1+\varepsilon$ 近似,除非 P=NP.

从最优化的角度来说,MLST 可表述为,在一个给定的图中找出一棵生成树,使得该生成树具有最多的叶子节点.从最优化问题的角度来说,MLST 与另一个著名的图问题"最小连通支配集"问题(Minimum Connected Dominating Set,MCDS)是等价的.即若在给定图 G 中找到有最多叶子的生成树,则该生成树中的非叶子节点就构成一个最小连通支配集,反之亦成立.但是从 NP 完全理论的角度,即如果我们考虑该问题的"判定问题"版本,则这两个问题并非是等价的,而是构成一对对偶问题[3],这对对偶问题解决的方法和难度自然也不一样.

2 相关研究

2.1 MLST 问题早期的理论研究

对于 MLST 问题,早期 Lovasz、Payan 和 Storer[4-5] 等人研究了一个变形问题,即节点度为 3 的 n 个节

收稿日期:2009-11-27;最终修改稿收到日期:2010-10-28.本课题得到广东省自然科学基金(8151032001000013)资助.**高文宇**,男,1972 年生,博士,教授,主要研究领域为参数算法理论及应用.E-mail: gwyy@163.com.

点构成的连通图,其生成树的叶子数至少可以达到多少.随后,在文献[6-7]中都独立地证明了节点度至少为3的n个节点构成的连通图,其生成树的叶子数至少可以达到$n/4+2$.

20世纪80年代以来,图子式(Graph Minor)理论[8]的发展对算法研究带来了新的思路.在文献[9]中Fellows等人利用图子式理论对若干组合优化问题进行了研究,其中借助图子式理论对MLST问题也给出了一个极富创见的解.给定一个图集F,F中的图没有哪一个图具有一棵生成树,使得该生成树有大于等于k个叶子节点.显然,图集F中任意图的子式也不会有叶子节点数大于等于k的生成树,这种特性我们称之为图集F关于子式运算封闭.那么根据图子式定理,图集F的障碍集(obstructions)是有限的,而且可以在多项式时间内识别出这些障碍集,因此MLST问题可以在多项式时间内求解.

但遗憾的是,目前很多利用图子式理论进行的研究主要是证明了解决问题的时间复杂度上限,其方法是非构造性的理论证明,即可以证明这个多项式时间的算法是存在的,但我们并不知道这个算法的任何细节,也就是说,我们需要去寻找一个可行的构造性算法,而且子式包含判定算法中包含着极大的常数项,也使得这类算法在实际应用中存在问题.

2.2 MLST问题的近似算法

对于MLST问题研究的另一个途径是寻找有效的近似算法.其中最重要的进展之一是Lu在文献[10]中提出的一个3-近似算法.在该文中,给出了一个贪婪算法,首先在给定的图G中构造出一个最大的多叶森林(maximally leafy forest)F,然后通过向F中添加一些边来连通F中的子树,从而最终得到图G的一棵生成树.文中证明了通过最大多叶森林构造出来的生成树的叶子节点数至少是图G的任意生成树的叶子节点数的1/3,因此该贪婪算法是一个3-近似算法.在一些特殊图中,近似算法取得了更好的结果,文献[11]中得到了立方体图中MLST问题的3/2-近似算法.

2.3 参数算法在解决MLST相关问题中的进展

对于一些组合优化问题,参数理论是一个很好的解决方案.经过近年来的一些研究,参数算法在很多与MLST问题相关的问题上都取得了很好的成果,如平面图的支配集问题等.参数理论的研究最初来源于观察到很多计算问题都与一个取值范围很小的重要参数相联系,利用参数的性质可以在一定程度上加速计算.

应用参数理论来求解NP难问题时,首先要将该问题转化为参数化问题.参数化问题Q是一个用二元组(x,k)表示的判定问题,其中x代表一个具体的问题,k是一个非负整数,称为参数.然后应用参数理论的算法技术对参数化问题Q加以求解.若我们能够设计一种参数化求解算法,其时间复杂度形如$O(f(k)|x|^c)$,则我们把该问题Q称为固定参数可解的问题(Fixed Parameter Tractable,FPT),简称FPT问题.

FPT算法设计技术的研究一直是参数理论中的研究热点.核化(kernelization)是设计FPT算法的主要技术之一.参数理论中的一个最重要的定理就是:一个参数化问题是FPT问题当且仅当该问题是可核化的.因此,在参数算法设计中,核化技术应用最为广泛.

核化是指如果存在一个多项式时间算法K和一个递归函数g,使得对于参数问题Q的任意一个实例(x,k),应用算法K将Q转化为一个新实例(x',k'),使得$|x'|\leq g(k)$和$k'\leq k$,并满足当且仅当(x',k')是参数问题Q的一个真实例,(x,k)也是Q的一个真实例,则我们说参数问题Q是可核化的,算法K称为核化算法,新实例(x',k')称为核心化后的问题核.核化后问题规模极大地降低.核的大小可能是参数k的线性式(线性核),也可能是参数k的多项式(多项式核),甚至是参数k的非多项式.

事实上,一个规模极大降低的核不仅可以用于设计参数算法,也可用于近似算法的设计.因此核化算法一直是参数理论研究的一个热点领域.

对于MLST问题的参数算法研究,文献[12]提出了一种分支算法能用$O(4^k poly(n))$时间解决该问题,文献[13]则借助于MLST问题的对偶问题即连通支配集问题的一些研究成果,提出了一个时间复杂度为$O(1.8966^n)$的算法.

MLST问题的一个变形就是求有向图中的多叶子生成树(Directed Maximum Leaf Spanning Out-Tree),由于有向图中弧的单向性,使得用于无向图MLST问题的一些方法如图子式理论和文献[4-7]中的方法都无法直接应用于有向图中变形问题的求解.文献[14]给出了有向图多叶子生成树问题的一个时间复杂度为$2^{O(k\log k)}\cdot n^{O(1)}$的参数算法,这也证明了该问题也是属于FPT类的.文献[15]进一步研究了有向图的多叶子生成树问题,得到了一个该问题的平方核,文献[16]则研究了有向无环图中的多叶子生成树问题,其结论是,若限定在有向无

环图中，则该问题存在大小为 $6.6(k+2)$ 的线性核.

对于与 MLST 问题密切相关的支配集问题的核化算法研究，近年来也取得了重要的进展. 如前所述，连通支配集问题和 MLST 问题构成一对对偶问题，但是它们的解决办法和难度都是不一样的. 早期的研究表明，参数化支配集问题是一个 W[2]完全问题[17]，因此它不是 FPT 可解的. 但是将参数化支配集问题限定在平面图这样一个背景下(也就是说要判定在一个平面图中是否存在大小为 k 的支配集)，则平面图支配集问题可以在 $O(c^{\sqrt{k}}\cdot n)$ 时间求解，其中 $c\leq 4^{6\sqrt{34}}$[18]，随后，常数 c 的上界又被优化到 2^{27}[19]以及 $2^{15.13}$[20]. 最近的研究表明，在平面图上求解支配集，通过两个图归约和预处理技术，可以得到一个不超过 $335k$ 大小的核[21]，这个线性核完全独立于原始图的规模. 随后，Chen[3]设计的核化算法将平面图支配集问题的核进一步降低到 $67k$. 由于 k 相对 n 通常是很小的数，问题的规模就从 n 降低到 k 的线性函数，这也就大大降低了问题的规模. 因此参数算法技术在一些问题的求解上是极其有益的.

3　最多叶子生成树问题的核化算法

对于 MLST 问题的核化，我们所做的工作如下：

(1) 首先设计了一些化简规则对给定的图中的 2 度节点进行化简，消去原图中的 2 度节点，并且化简后的图和原图在求解 MLST 问题上是等价的.

(2) 化简后的图中只存在 1 度节点和度大于等于 3 的节点. 我们证明了在这种图中，其生成树的叶子节点数至少可以达到 $(N+6)/4$(其中 N 为化简后的图的节点数)，并给出构造这样的生成树的构造性方法.

(3) 对于 MLST 问题，若 $k\leq(N+6)/4$，则回答“是”，即原图中存在一棵生成树，其叶子节点至少有 k 个；若 $k>(N+6)/4$，则 $N<4k-6$，即化简后的图的节点数小于 $4k-6$，这是一个与原图的节点规模无关的线性核.

下面对相关问题给出详细的描述.

3.1　基本概念

定义 1.　图 $G=(V,E)$ 称为简单连通无向图，需满足以下条件：

(1) G 为无自回路的、连通的无向图.

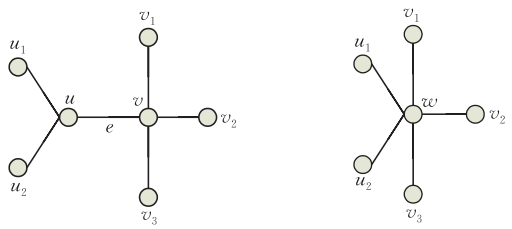
(2) G 中任意两个节点之间至多有一条边.

定义 2.　图 G 中节点 u 和节点 v 之间存在一条边，则称节点 u 与节点 v 相邻.

定义 3(d 度节点).　图 $G=(V,E)$ 中的任意节点 u，称 u 在图中的相邻节点的个数为节点 u 的度，记为 $D(u)$. 假设 $D(u)=d$，则将节点 u 称为 d 度节点.

定义 4(MLST 问题).　给定简单连通无向图 $G=(V,E)$ 以及非负整数 k，问图 G 中是否包含一棵生成树 T，使得生成树 T 中度为 1 的节点(即叶子节点)总数至少为 k.

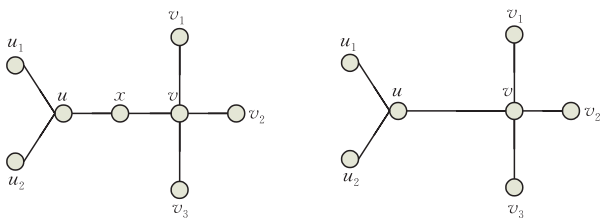
定义 5(边的收缩).　图 G 中，端点为 u 和 v 的边 e 的收缩，就是用一个新顶点 w 代替 u 和 v 并且与新顶点 w 关联的边包括除 e 之外的所有与 u 和 v 关联的边(参见图 1).



(a) 边e收缩前　　(b) 边e收缩后

图 1　边的收缩操作

定义 6(2 度节点的压缩).　图 G 中，若存在一个 2 度节点 x，则对该 2 度节点的压缩就是删除该 2 度节点及其相邻的两条边，然后将该 2 度点原来的两个相邻节点用一条边连接起来(参见图 2).



(a) 节点x压缩前　　(b) 节点x压缩后

图 2　2 度节点的压缩操作

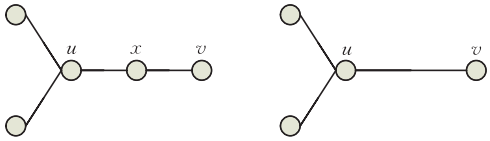
3.2　2 度节点的化简规则

通过对简单连通无向图中 2 度节点的研究，我们可以得到如下的一些化简规则.

(1) 若 2 度节点的其中一个相邻节点的度为 1，则该 2 度节点可以被压缩，该 2 度节点被压缩后的图与原图在求解 MLST 问题上是等价的，即原图中若存在叶子数至少为 k 的生成树，则化简后的图中亦存在；若原图中不存在，则化简后的图中也不存在. 这种情况是显然的，因为图中的 1 度节点必定是最终生成树中的叶子节点，因此与 1 度节点相邻的 2 度节点必定不是最终生成树的叶子节点，因此可

以被压缩.

图 3(a)中的节点 x 就是这种情况,节点 x 可以被压缩,压缩后见图 3(b).



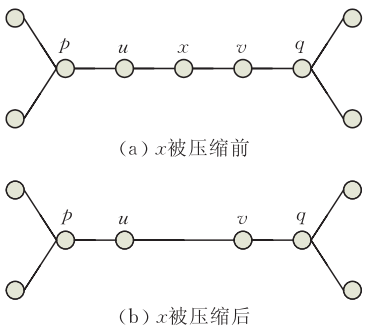
(a) x被压缩前 (b) x被压缩后

图 3 2 度节点的化简,第(1)种情况

(2)若 2 度节点的两个相邻节点都是 2 度节点,则该二度节点也可以被压缩,压缩后的图与原图在求解 MLST 问题上是等价的.

图 4(a)中的节点 x 就是这种情况,节点 x 可以被压缩,压缩后见图 4(b).

如图 4(a),在这种情况中,$p-u-x-v-q$ 这条路径在最终的生成树中可能被断开,断开后符合情况(1),其中一个节点可被压缩;若 $p-u-x-v-q$ 这条路径在最终的生成树中没有被断开,则节点 x 也不是叶子节点;因此 x 被压缩后的图与原图在求解 MLST 问题上是等价的.



(a) x被压缩前

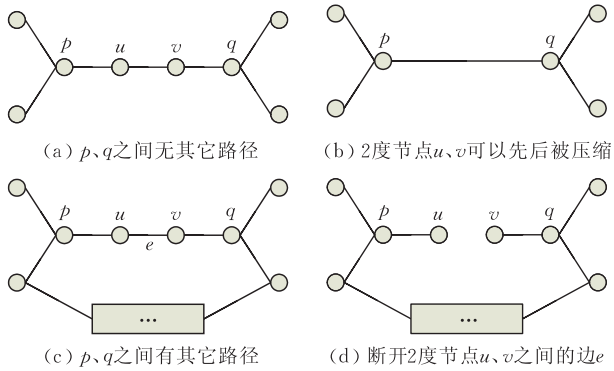
(b) x被压缩后

图 4 2 度节点的化简,第(2)种情况

(3)若 2 度节点的两个相邻节点中,其中一个相邻节点的度为 2,另一个相邻节点的度大于等于 3.事实上这就是两个 2 度节点相邻,然后它们分别再与一个大于等于 3 度的节点相邻.图 5(a)中的节点 u、v 都属于这种情况,因此我们将 u、v 合并在一起考虑.

若这两个相邻的 2 度节点 u、v 各自的另一个相邻节点 p、q 之间除路径 $p-u-v-q$ 之外无别的连通路径,则 2 度节点 u、v 可以被压缩,压缩后见图 5(b).

若这两个相邻的 2 度节点 u、v 各自的另一个相邻节点 p、q 之间除路径 $p-u-v-q$ 之外还有别的连通路径,示意图见图 5(c),则 2 度节点 u、v 之间的边可以断开,因为断开此边可以获得两个叶子节点,断开后的情况见图 5(d).



(a) p、q之间无其它路径 (b) 2度节点u、v可以先后被压缩

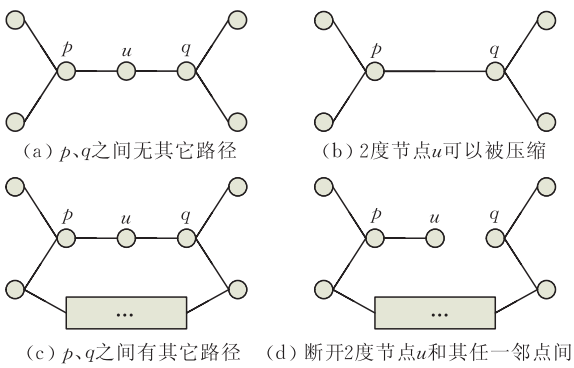
(c) p、q之间有其它路径 (d) 断开2度节点u、v之间的边e

图 5 2 度节点的化简,第(3)种情况

(4)若 2 度节点的两个相邻节点的度都大于等于 3.图 6(a)中的节点 u 就属于这种情况.若这个 2 度节点 u 的两个相邻节点 p、q 之间除路径 $p-u-q$ 之外再无别的连通路径,则 2 度节点 u 可以被压缩,压缩后见图 6(b).

若这个 2 度节点 u 的两个相邻节点 p、q 之间除路径 $p-u-q$ 之外还有别的连通路径,见图 6(c),则可以断开 2 度节点 u 和它的任意一个相邻点之间的边,因为断开此边可以获得一个叶子节点,断开后的情况见图 6(d).

但是,在此情况下,可能会产生新的 2 度节点.即断开边后,原 2 度节点变成了 1 度节点,而原 2 度节点 u 的邻点可能由于度减 1 从而变成一个新的 2 度节点.所以在断开边后应立即检查相关节点,若该节点的度仍然大于等于 3,则继续;若断开边后,度由 3 减为 2,则立即运用前面的规则处理这个新产生的 2 度节点.



(a) p、q之间无其它路径 (b) 2度节点u可以被压缩

(c) p、q之间有其它路径 (d) 断开2度节点u和其任一邻点间的边

图 6 2 度节点的化简,第(4)种情况

3.3 核化算法及复杂性分析

有了前面 4 条化简规则,可以给出核化算法如下(包括步 1~3):

1.依次访问图 G 中的所有节点,对图中的 2 度节点应用第(1)和第(2)条化简规则,得到图 G'.则图 G'中剩余的 2 度节点只可能是规则(3)和规则(4)中所述情况.对于 n 个节

点的图执行此步骤所用时间为$O(n)$.

2. 依次访问图G'中所有节点，对图G'中符合规则(3)所描述的2度节点运用化简规则化简，得到图G''. 则图G''中剩余的2度节点只可能是规则(4)中所述的情况. 对于n个节点的图执行此步骤所用时间为$O(n^2)$. 因为对每对符合规则(3)的2度节点进行化简时需要判断一对节点间是否存在另一条连通路径，使用深度优先算法需要$O(n)$时间，因此总共的时间为$O(n^2)$.

3. 对图G''运用化简规则(4)化简，得到图G'''，则图G'''中不再存在2度节点，只剩下1度节点和度大于等于3的节点. 对于n个节点的图执行此步骤所用时间为$O(n^2)$. 理由同上.

因此，全部执行完步1～3所需时间为$O(n^2)$.

3.4　无2度节点图的核化

对一简单连通无向图G，若采用3.3节中给出的核化算法对2度节点进行化简，在化简后的图中就不再有2度节点.

对于一个不存在2度节点的简单连通无向图，有如下定理成立.

定理1.　N个节点的简单连通无向图G，若图G不含2度节点，则其必有一棵生成树，使得该生成树的叶子节点(度为1的节点)至少为$\frac{N+6}{4}$.

证明.

为了证明上述定理，我们引入一个构造性的算法，在任意的不含2度节点的图G中都能构造出一棵生成树，使得该生成树的叶子节点数至少为$\frac{N+6}{4}$，从而证明该结论.

在构造图G的生成树的过程中，我们用T表示当前已被构造出的包含n个节点，l个叶子节点的树. 若x是T的一个叶子节点，则我们用$d'(x)$表示x在$G-T$中的度，即$d'(x)$等于x在$G-T$中的邻点个数.

我们定义“扩展节点x”操作如下，并简称为“点扩展”.

定义7(扩展节点x).　向树T中增加$d'(x)$条边，也就是增加x与x在$G-T$中的邻居之间的边.

那么通过一系列的点扩展操作就可以将树T从一棵小树扩展成为图G的一棵完整的包含图G全部节点的生成树.

我们将当前的生成树T中的$d'(x)=0$的叶子节点x称为“死叶子”，因为这些“死叶子”不能再进行点扩展操作，从而在构造T的过程中产生的“死叶子”一定是最终的生成树中的叶子节点. 另外用m表示当前树T中的“死叶子”数目. 用Δn，Δl，Δm分别表示当前树T执行一次点扩展操作后所带来的相应的n，l，m的增量.

如果一个点扩展序列对树T的影响满足“累加不等式”$3\Delta l+\Delta m\geq\Delta n$，则称之为“合法”的点扩展序列.

我们从一棵初始树T开始，通过执行一系列“合法”的点扩展序列将T逐步扩展成为图G的生成树.

构造树T的过程：

(1) 若不含2度节点的图G中只有1度节点，则是图7(a)所示情况，定理1显然成立.

(2) 若不含2度节点的图G中有一个度大于等于4的节点，则可以找出一棵如图7(b)所示的初始子树.

(3) 若图G中都是3度节点和1度节点，而且图G中每一条不与1度节点关联的边都是图中某个三角形的一条边，则图G是图7(c)、图7(d)和图7(e)所示情况，这3种情况下，定理1显然都成立.

(4) 若图G中都是3度节点和1度节点，而且图G中至少存在一条不与1度节点关联的边，该条边不是图中任何一个三角形的一条边，则可以找出一棵如图7(f)所示的初始子树.

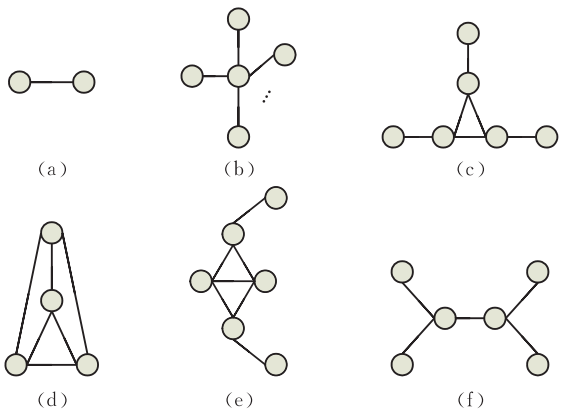


图7　初始情况

若T通过一系列的“合法”的点扩展操作最终变成了图G的一棵包含L个叶子节点的生成树. 那么在树T的生长过程中所产生的叶子节点最终都变成了“死叶子”节点(因为最终的生成树中的叶子节点都是“死叶子”，即都无法再进行点扩展操作). 由于所有的点扩展操作都是“合法”的，即满足前面定义的“累加不等式”. 假设共进行了i次点扩展操作，则每一次的点扩展操作满足的“累加不等式”

如下：

$$\begin{cases} 3\Delta l_1+\Delta m_1\geq\Delta n_1 \\ 3\Delta l_2+\Delta m_2\geq\Delta n_2 \\ \cdots \\ 3\Delta l_i+\Delta m_i\geq\Delta n_i \end{cases}.$$

将上述 i 个不等式累加起来得

$$3(\Delta l_1+\Delta l_2+\cdots+\Delta l_i)+(\Delta m_1+\Delta m_2+\cdots+\Delta m_i)\geq(\Delta n_1+\Delta n_2+\cdots\Delta n_i) \quad (1)$$

因为：

初始生成树的叶子数$+\Delta l_1+\Delta l_2+\cdots+\Delta l_i=$最终生成树的叶子数$=L$ (2)

$\Delta m_1+\Delta m_2+\cdots+\Delta m_i\leq L$，因为初始生成树可能包含“死叶子” (3)

$\Delta n_1+\Delta n_2+\cdots+\Delta n=N-$初始生成树的节点数 (4)

将式(2)～(4)代入式(1)有

若初始子树是图 7(b)所示情况，则 $3(L-D)+L\geq N-D-1$，其中 D 是一个大于等于 4 的常数，即 D 是图 7(b)所示的初始子树中的叶子节点数目. 化简即得 $4L\geq N+2D-1\geq N+7$，即 $L\geq\frac{N+7}{4}$.

若初始子树是图 7(f)所示情况，则 $3(L-4)+L\geq N-6$，化简即 $L\geq\frac{N+6}{4}$.

下面我们就给出一系列“合法”的点扩展操作，使得仅通过这些操作就可以完成图 G 的生成树的构建. 因此，根据前面的证明，通过这些“合法”的操作得到的生成树的叶子节点数满足 $L\geq\frac{N+6}{4}$.

“合法”的点扩展操作：

O1. 对当前的树 T 中的某个叶子节点 x，若 $d'(x)\geq 2$，则在节点 x 进行的点扩展操作满足 $\Delta l=\Delta n-1$，$\Delta m\geq 0$，显然在此条件下，$3\Delta l+\Delta m\geq\Delta n$ 成立，因此该操作“合法”. 参见图 8(a).

O2. 对当前的树 T 中所有的满足 $d'(x)\leq 1$ 的节点 x，图 G 中一定有某个不属于树 T 的节点，若该节点 x 有至少两个邻点在树 T 中，则对该节点的其中一个邻点进行点扩展操作会增加至少一个“死叶子”(也就是另一个邻点会变成“死叶子”). 因此这样的点扩展操作会导致 $\Delta l=0$，$\Delta m\geq 1$，$\Delta n=1$，显然在此条件下，$3\Delta l+\Delta m\geq\Delta n$ 成立，因此该操作“合法”. 参见图 8(b).

O3. 对当前的树 T 中所有的满足 $d'(x)\leq 1$ 的节点 x，图 G 中一定有某个不属于树 T 的节点，若 y 是 x 在树 T 之外的唯一邻点，且 y 在图 G 中的度不为 1，即至少为 3，则 y 至少有两个不属于树 T 的邻点，则先在 x 进行点扩展，紧接着在 y 进行点扩展会导致 $\Delta l=\Delta n-2\geq 1$，$\Delta m\geq 0$，显然在此条件下，$3\Delta l+\Delta m\geq\Delta n$ 成立，因此该操作“合法”. 参见图 8(c).

O4. 对当前的树 T 中所有的满足 $d'(x)\leq 1$ 的节点 x，图 G 中一定有某个不属于树 T 的节点，若 y 是 x 在树 T 之外的唯一邻点，且 y 在图 G 中的度等于 1，则在 x 进行点扩展，y 就成为一个新的“死叶子”，这种扩展会导致 $\Delta l=0$，$\Delta m=1$，$\Delta n=1$，显然在此条件下，$3\Delta l+\Delta m\geq\Delta n$ 成立，因此该操作“合法”. 参见图 8(d).

因为在图 G 中，节点的度要么为 1，要么大于等于 3. 度为 1 的节点只能是在别的节点执行点扩展时被加入到树 T，而且一旦被加入马上就变成了“死叶子”(规则 O4). 而度大于等于 3 的节点要么有至少两个邻点在 T 中，要么至少有两个邻点在 T 之外，则规则 O1 或 O2 或 O3 必占其一. 所以通过规则 O1～O4 可以从初始生成树构建出图 G 最终的生成树，而每一步都是满足不等式 $3\Delta l+\Delta m\geq\Delta n$ 的“合法”操作. 所以最终的生成树的叶子节点数满足 $L\geq\frac{N+6}{4}$.

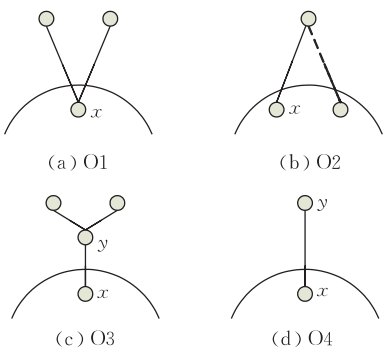
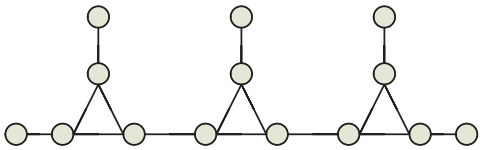


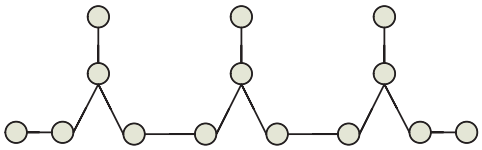
图 8 “点扩展”规则

事实上，定理 1 给出的下限是一个严格的(最好的)下限(tight bound). 也就是说，在不存在 2 度节点的图中，其生成树的最大叶子节点数至少可以达到 $\frac{N+6}{4}$，但是该下限无法再被改进. 事实上，图 9(a)就是一个达到该下限的图. 在该图中仅存在 1 度节点和 3 度节点，该图的具有最多叶子的一棵生成树见图 9(b). 因此最大叶子节点数 $L=5$，图的节点数 $N=14$，所以 $L=5=\frac{N+6}{4}$. 显然也无法从该图中

找出一棵生成树使其叶子节点数大于$\frac{N+6}{4}$. 证毕.



(a) 原图



(b) 图的最多叶子生成树

图 9　满足 $L=(N+6)/4$ 的图

所以，根据定理 1，我们可以得到简化后包含 1 度节点和大于等于 3 度节点的图的叶子节点数的一个下限. 因此，对于 MLST 问题我们有如下的核化定理.

定理 2.　对于在任意简单连通无向图 G 上求解 MLST 问题，应用 3.3 节中的核化算法，可以得到一个 $4k-6$ 大小的核.

证明.

(1) 执行 3.3 节的核化算法，可以将图 G 化简为不含 2 度节点的图 G'，设图 G' 中的节点数为 N 个.

(2) 根据定理 1，N 个节点且不含 2 度节点的图，则其必有一棵生成树，使得该生成树的叶子节点数至少为$\frac{N+6}{4}$. 所以对于定义 4 描述的 MLST 问题，若化简后的图 G' 的节点数 N 满足 $k\leq\frac{N+6}{4}$，则回答“是”，即原图 G 中存在一棵生成树，其叶子节点至少有 k 个.

若 $k>\frac{N+6}{4}$，则 $N<4k-6$，即化简后的图 G' 的节点数小于 $4k-6$，这是一个与原图的节点规模无关的线性核. 因此我们可以在节点数小于 $4k-6$ 的图 G' 中来求解 MLST 问题.　　证毕.

4　结束语

最多叶子生成树问题是 NP 完全理论中的一个经典问题，其在网络设计和电路布线中都有着重要的应用背景. 我们通过深入研究，设计了有效的化简(核化)算法，并证明了化简后的图是一个不超过 $4k-6$ 大小的线性核. 核化算法也是参数理论研究的一个热点领域，有效的核化算法能极大地降低原始问题的规模，从而在一些应用场合能大大降低问题求解的复杂度. 如我们设计的核化算法，在得到 $4k-6$ 大小的线性核之后，利用这个与原问题无关的线性核不仅可以设计参数算法求解，也可以设计近似算法求解，算法的复杂度仅与参数 k 有关，而与原始问题的规模无关.

参　考　文　献

[1] Garey M R, Johnson D S. Computers Andintractability: A Guide to the Theory of NP-Completeness. New York: W. H. Freeman & Co., 1979

[2] Galbiati G, MaNoli F, Morzenti A. A short note on the approximability of the maximum leaves spanning tree problem. Information Processing Letters, 1994, 52(1): 45-49

[3] Chen J, Fernau H, Kanj I, Xia G. Parametric duality and kernelization: Lower bounds and upper bounds on kernel size. SIAM Journal on Computing, 2007, 37(4): 1077-1106

[4] Payan G, Tchuente M, Xuong N H. Arbres avec un nombre maximum de sommets pendants. Discrete Mathematics, 1984, 49(3): 267-273

[5] Storer J A. Constructing full spanning trees for cubic graphs. Information Processing Letters, 1981, 13(1): 8-11

[6] Griggs J R, Kleitman D J, Shastri A. Spanning trees with many leaves in cubic graphs. Journal of Graph Theory, 1989, 13(6): 669-695

[7] Kleitman D J, West D B. Spanning trees with many leaves. SIAM Journal on Discrete Mathematics, 1991, 4(1): 99-106

[8] Robertson N, Seymour P D. Graph Minors. II. Algorithmic aspects of tree-width. Journal of Algorithms, 1986, 7(3): 309-322

[9] Fellows M R, Langston M A. On well-partial-order theory and its application to combinatorial problems of VLSI design. SIAM Journal on Discrete Mathematics, 1992, 5(1): 117-126

[10] Lu H, Ravi R. Approximating maximum leaf spanning trees in almost linear time. Journal of Algorithms, 1998, 29(1): 132-141

[11] Bonsma P, Zickfeld F. A 3/2-approximation algorithm for finding spanning trees with many leaves in cubic graph//Proceedings of the WG2008. Durham, UK, LNCS 5344. Springer-Verlag, 2008: 66-77

[12] Kneis J, Langer A. Rossmanith P. A new algorithm for finding trees with many leaves//Proceedings of the AC 2008. Sophia Antipolis, France, LNCS 5369. Springer-Verlag, 2008: 270-281

[13] Fernau H, Kneis J, Kratsch D, Langer A et al. An exact algorithm for the maximum leaf spanning tree problem//Proceedings of the IWPEC2009. LNCS 5917. Copenhagen, Denmark. Springer-Verlag, 2009: 161-172

[14] Bonsma P, Dorn F. Tight bounds and a fast FPT algorithm for directed max-leaf spanning tree//Proceedings of the ESA 2008. Karlsruhe, Germany, LNCS 5193. Springer-Verlag, 2008: 222-233

[15] Daligault J, Thomasse S. On finding directed trees with many leaves//Proceedings of the IWPEC2009. Copenhagen, Denmark, LNCS 5917. Springer-Verlag, 2009: 86-97

[16] Daligault J, Gutin G, Kim E J, Yeo A. FPT algorithms and kernels for the directed k-leaf problem. Journal of Computer and System Sciences, 2010, 76(2): 144-152

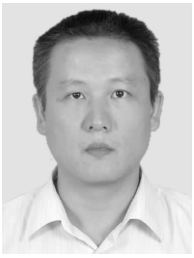
[17] Downey R G, Fellows M R. Fixed-parameter tractability and completeness I: Basic results. SIAM Journal on Computing, 1995, 24(4): 873-921

[18] Alber J, Bodlaender H L, Fernau H, Kloks T. Niedermeier R. Fixed parameter algorithms for dominating set and related problems on planar graphs. Algorithmica, 2002, 33(4): 461-493

[19] Kanj I A, Perkovic L. Improved parameterized algorithms for planar dominating set//Proceedings of the MFCS2002. Warsaw, Poland, LNCS 2420. Springer-Verlag, 2002: 399-410

[20] Fomin F V, Thilikos D T. Dominating sets in planar graphs: Branch-width and exponential speed-up//Proceedings of the 14th ACM-SIAM SODA. New York, 2003: 168-177

[21] Alber J, Fellows M R, Niedermeier R. Polynomial-time data reduction for dominating set. Journal of the ACM, 2004, 51(3): 363-384



GAO Wen-Yu, born in 1972, Ph.D., professor. His research interests include computer algorithm, network and programming.

Background

This work is supported by Guangdong Natural Science Foundation under grant (8151032001000013), which focuses on the application of parameterized algorithm in network. The theory of parameterized computation and complexity is a recently developed subarea in theoretical computer science. The theory is aimed at practically solving a large number of computational problems that are theoretically intractable. The theory is based on the observation that many intractable computational problems in practice are associated with a parameter that varies within a small or moderate range. Therefore, by taking the advantages of the small parameters, many theoretically intractable problems can be solved effectively and practically. The theory of parameterized computation and complexity has found wide applications in area such as computer network, database system, programming languages, parallel and distributed computing, etc.

According to parameterized complexity theory, a decidable parameterized problem is fixed-parameter tractable if and only if it can be kernelized. Kernelization is the most widely applied and effective technique in parameterized algorithm design. Maximum leaf spanning tree is a classical NP-hard problem. There are lots of applications of the maximum leaf spanning tree in network. Many different algorithmic approaches were tried on this problem, including approximation algorithms, linear programming, etc. In this paper, some reduction rules for maximum leaf spanning tree problem are given, which could reduce the 2-degree node in connected graph. Moreover, it is also proven that any connected graph without 2-degree node has a spanning tree including at least $(N+6)/4$ leafs. The reduction rules and the low bound are used to construct a kernelization algorithm for maximum leaf spanning tree problem, which algorithm can output a $4k-6$ linear kernel of Maximum Leaf Spanning Tree problem in $O(n^2)$ time.