arXiv:2303.03125v2 [cs.DS] 1 Apr 2024

最大叶数生成树的一个简单 2-近似*

I-Cheng Liao† Hsueh-I Lu‡

2024年4月2日

摘要

对于一个 m 边连通简单图 G，寻找一棵叶子数最多的生成树是 MAXSNP完全问题。即使 G 是平面图且 G 的最大度至多为四，该问题仍然是 NP 完全问题。Lu 和 Ravi 给出了已知的首批多项式时间近似算法，其近似因子分别为 5 和 3。之后，他们获得了一种在近线性时间内运行的 3-近似算法。目前最好的结果是 Solis-Oba、Bonsma 和 Lowski 的 $O(m)$ 时间 2-近似算法。我们提出了一种不同的简单 $O(m)$ 时间 2-近似算法，其分析更为简单。本文谨以此纪念我们敬爱的挚友西关隆夫教授。

1 引言

我们谨以此文，向挚爱的朋友西关隆夫教授致敬，铭记他对图算法领域作出的持久贡献。在卓越的学术生涯中，西关教授为该领域贡献了许多经典成果，其中包括关于近似算法 [6, 14, 22, 23] 和生成树 [15, 20, 21, 30]的有趣研究。我们怀着深切的爱戴与感激，缅怀他卓越的成就。

对于一个有 n 个顶点、 m 条边的连通简单图 G，求一棵叶子数最多的生成树是 NP完全问题；即使 G 是平面图且最大度至多为四 [12]，该问题仍然是 NP完全的。此外，该问题是 MAXSNP完全的 [11], ，这意味着除非 NP $=$ P [1, 2, 24]，否则不存在多项式时间近似方案。该问题在通信网络、电路布局和计算机图形学中有应用 [7, 8, 31]。关于最大叶生成树问题的最新综述，参见 [26] 。

Lu 和 Ravi [18] 基于局部搜索，首次给出了该问题的多项式时间近似算法。他们的 3-近似（分别为 5-近似）算法运行时间为 $O(m^2n^3)$（分别为 $O(mn^2)$）。随后，他们又给出了一种新的 3-近似算法，其运行时间为 $O(m \cdot \alpha(m,\ n))$ [19]。Solis-Oba、Bonsma 和 Lowski [27, 28] 通过给出一种

*有关本工作的初步版本，请参见 [17] ； 本工作是独立于 [27, 28]开展的。

†国立中正大学计算机科学与信息工程系，台湾嘉义。

‡通讯作者。电子邮箱：hil@csie.ntu.edu.tw。国立台湾大学计算机科学与信息工程系，台湾台北。该作者的研究受国家科学及技术委员会（NSTC）110–2221–E–002–075–MY3号资助项目支持；该委员会前身为科学技术部（MOST）。

$O(m)$ 时间的 2-近似算法，改进了他们的结果。他们的算法包含多个循环阶段。其关于近似比的证明也较为复杂。我们给出了一种更简单的替代性 $O(m)$ 时间 2-近似算法：(1) 我们的算法只有一个简单循环，利用数组和链表等基本数据结构即可轻松实现为线性时间运行。(2) 我们关于近似比的证明更短。

定理 1. 树算法(G) 是一个 $O(m)$时间的 2-近似算法，用于求解最大叶节点生成树问题。

我们采用了与 Solis-Oba、Bonsma 和 Lowski [27, 28]提出的方法类似的思路：每一轮中，算法从当前树中选取一个顶点 u，并通过将其所有尚未包含在树中的邻居 u 直接连接到 u，来扩展这棵树。 u的选择取决于其位于当前树之外的邻居数量。我们的算法以贪心方式扩展树，力求在每一轮中尽可能增加叶子数量。如果某一轮无法增加叶子数量（即每个叶节点在树外至多有一个邻点），算法就向前多看一轮，尝试最大化连续两轮所增加的叶子数量。当连续两轮都找不到合适的扩展顶点时，算法便采用“深度优先”扩展策略，以简化近似比的分析。

最大叶数生成树问题在文献中得到了广泛研究。早期的大多数工作都致力于在最小度至少为 d、其中 $d \geq 3$ 的图中寻找多叶生成树。对于这类图，文献 [4, 5, 9, 13, 16, 25, 29] 给出了生成树中可实现叶子数量的良好下界。也有研究探讨了如何在多项式时间内判断给定图是否存在至少有 k 个叶子的生成树，其中 k 为固定值。首个此类算法由 Fellows 和 Langston 提出，运行时间为 $O(n^2)$ [10]。Bodlaender [3] 将运行时间改进至 $O(m)$。

第 2 节介绍我们的算法。第 3 节分析近似比。第 4 节展示了我们算法的线性时间实现。第 5 节总结全文。

2 算法

本文中的所有图都是无向简单图，即不含重边和自环。设 H 是一个图。令 $V(H)$（分别为 $E(H)$）表示 H 的顶点（分别为边）集。令 $d_H(u)$ 表示 u 在 H 中的邻点数，其中 $u \in V(H)$。令 $L(H)$（分别为 $V_2(H)$ 和 $V_0(H)$）表示由每个满足 $d_H(u) = 1$（分别满足 $d_H(u) \geq 2$ 和 $d_H(u) = 0$）的顶点 u 组成的集合。若 $u \in L(H)$，则顶点 $u \in V(H)$ 是 H 的叶节点。

令 G 为输入的 n 个顶点、m 条边的连通图。一般性不失地，假设 G 包含一个满足 $d_G(a) \geq 2$ 的顶点 a。令 T 为 G 的一棵树。对于 T 的一个顶点 u，令 $V_T(u)$ 表示满足 $uv \in E(G)$ 的 $v \in V(G)$ 中的顶点，而令 $E_T(u)$ 表示满足 $v \in V_T(u)$ 的 G 中的边 uv。若 $|V_T(u)| = 1$，则令 $v_T(u)$ 表示 $V_T(u)$ 中的顶点。

- 令 $W_2(T)$ 表示 T 中满足 $|V_T(u)| \geq 2$ 的每个顶点 u。
- 令 $W_1(T)$ 表示 T 中满足 $|V_T(u)| = 1$ 的每个顶点 u，以及$|V_{T \cup E_T(u)}(v_T(u))| \geq 2$。
- 令 $W_0(T)$ 由 T 中满足 $|V_T(u)| = 1$ 和$|V_{T \cup E_T(u)}(v_T(u))| \leq 1$ 的每个顶点 u 组成。

树算法(G)：令 T 为 G 的初始树，该树由 G 中满足 $d_G(a) \geq 2$ 的一个顶点 a 组成。重复以下步骤，直到 $V(T) = V(G)$：如果 $W_2(T) \neq \varnothing$，则令 u 为 $W_2(T)$ 中任意一个顶点；否则，如果 $W_1(T) \neq \varnothing$，则令 u 为 $W_1(T)$ 中任意一个顶点；否则，令 u 为 $W_0(T)$ 中最近加入 $V(T)$ 的顶点。在 u 处扩展 T，令 $T = T \cup E_T(u)$。返回 T。

图1：我们的算法在连通图 G上的应用。

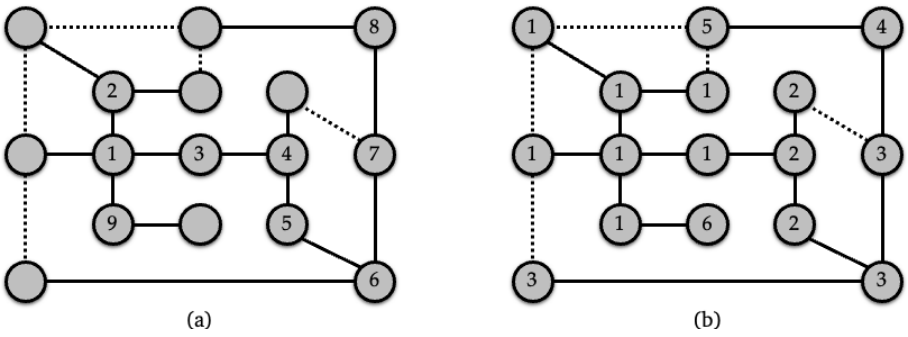


图2：我们的算法返回的生成树 T 示例。 T 的边以实线表示。(a) 顶点 i 是第 i个被扩展的顶点。没有标签的顶点是返回的生成树中的叶子。(b) 每个顶点的标签是其按照(a)中给出的扩展顺序确定的秩。

我们的算法如图1所示。也就是说，从 G的初始树 $T = \{a\}$ 开始，我们迭代地扩展 T，直到 $V(T) = V(G)$。每次迭代都在 T的一个顶点 u处扩展 T，其中 $|V_T(u)| \neq 0$（除非 $u = a$，否则该顶点必须是 T的一个叶节点），具体做法是令 $T = T \cup E_T(u)$。因此，该算法返回的树是 G的一棵生成树。扩展 T所选取的顶点依次从 $W_2(T)$、 $W_1(T)$和 $W_0(T)$中选择。因此，若要在 $W_i(T)$中满足 $i \in \{0, 1\}$ 的某个顶点处扩展 T，则每个满足 $i < j \leq 2$的$W_j(T)$均为空。此外， $W_0(T)$中的顶点按照它们加入 $V(T)$的逆序选择。结果是，一旦即将在某个顶点 $u \in W_0(T) \cup W_1(T)$处扩展 T，算法便通过反复令 $T = T \cup \{uv_T(u)\}$ 和 $u = v_T(u)$来从 u为 T生长一条路径，直到$|E_T(u)| \neq 1$。若停止条件为 $|E_T(u)| = 0$（分别为 $|E_T(u)| \geq 2$），则所生长的路径在算法返回的最终树中终止于一个叶节点（分别为一个度至少为三的顶点）。在何处扩展 T 的顶点选择顺序，对于下一节中分析近似比至关重要。图2(a)给出了一个示例，其中顶点 i 是扩展 T时所选取的第i个顶点。

3 近似比

在本节中，令 T 表示由我们的算法返回、以初始树的顶点 a 为根的树。对于 T 中除 a以外的每个顶点 v，令 $p(v)$表示 v 在 T中的父节点。对于 T 中每个满足 $d_T(u) \geq 2$的顶点 u，令 T_u 表示当即将在 u 处扩展 T_u 时 T 的子树，并令 $T^u = T_u \cup E_{T_u}(u)$。因此，当且仅当 $v \in V_{T_u}(u)$时，$u = p(v)$ 成立，即 uv 通过在 u处扩展 T_u而加入 T。定义每个 $v \in V(G)$的秩 $r(v)$如下。令 $r(a) = 1$。对于 T 中每条满足$u = p(v)$的边 uv，令

$$r(v) = \begin{cases} r(u) & \text{if } u \in W_2(T_u) \\ 1 + \max\limits_{w \in V(T_u)} r(w) & \text{otherwise.} \end{cases}$$

示例见图 2(b)。如果 uv 是 G 中的一条边，且 $r(u) < r(v)$，那么 $v \notin V(T_u)$。也就是说，当树即将在 u 处扩展时，树中不包含任何秩高于 $r(u)$ 的顶点。因此，对于 G 中的一条边 uv，若 $r(u) < r(v)$，如果 $uv \in E(T)$，那么 $u = p(v)$ 且 $u \in W_0(T_u) \cup W_1(T_u)$；否则，$u$ 是 T 中的一个叶节点，且 $r(p(u)) = r(u)$。令 U 表示由具有唯一秩的顶点组成的集合。图 2(b) 中的 U 由秩为 4、5 和 6 的顶点组成。如果 $u = p(v)$，则根据我们的算法定义可知

- $u \in W_2(T_u)$ 当且仅当 $r(u) = r(v)$，
- $u \in W_1(T_u)$ 当且仅当 $r(u) < r(v)$ 且 $v \notin U$，并且

当且仅当 $r(u) < r(v)$ 且 $v \in U$ 时，• $u \in W_0(T_u)$。

引理 1. 设 uvw 是一条由 G 构成的路径，且具有 $\{u, v\} \subseteq U$ 和 $r(u) < r(v) < r(w)$，则 $d_G(v) = 2$。

证明。反设 G 中存在 v 的一个邻点 x，它既不是 u，也不是 w。由 $\{u, v\} \subseteq U$ 和 $r(u) < r(v)$，可得 $r(u) = r(v) - 1$ 和 $u \in W_0(T_u)$。因此，$W_1(T_u) = W_2(T_u) = \varnothing$，这意味着 $r(x) < r(u)$。由 $\{u, w\} \subseteq V_{T' \cup \{xv\}}(v)$，可得 $x \in W_2(T_{p(u)}) \cup W_1(T_{p(u)})$，这与 $p(u) \in W_0(T_{p(u)})$ 矛盾。 □

引理 2。 u G 中的每个顶点至多有一个邻点 v 属于 G，且满足 $r(u) < r(v)$。

证明。反设 v 和 w 是 u 在 G 中的两个不同邻居，且满足

$$r(u) < r(v) \leq r(w). \tag{1}$$

因此，$u \neq a$。设 $p = p\ (u)$。根据等式（1），有 $V\ (T_p) \cap \{v, w\} = \varnothing$，这意味着 $p \notin W_0\ (T_p)$。此外，我们有 $p \notin W_1\ (T_p)$；否则，$\{v, w\} \subseteq V_{T_u}\ (u)$ 将意味着 $r\ (u) = r\ (v) = r\ (w)$，从而违反等式（1）。因此，$p \in W_2\ (T_p)$ 且 $\{v, w\} \subseteq V_{T^p}\ (u)$，这意味着 $u \in W_2\ (T^p)$。根据等式（1），有$V_{T_u}\ (u) \cap \{v, w\} = \varnothing$。因此，$u \in W_2\ (T_u)$，违反等式（1）。 □

令 F 为从 T 中删除所有满足 $r(u) \neq r(v)$的边 uv 后得到的森林。因此，对于 G中的任意两个顶点 u 和 v，当且仅当 u 和 v 在 F中连通时，$r(u) = r(v)$。

引理3。 若 uv 是 G 的一条边，且满足 $u \in V_2(F)$，则 $r(u) \geq r(v)$。

证明。反设 $r(u) < r(v)$，则推出 $v \notin V(T_u)$。根据 $u \in V_2(F)$，有 $r(u) = r(v)$，矛盾。

□

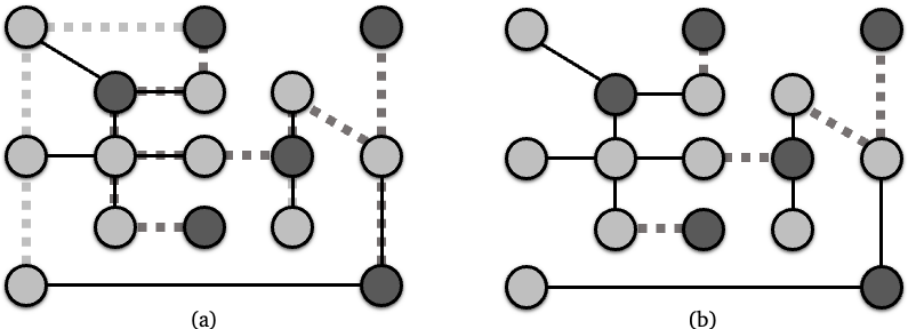


图3：实线边构成一个生成森林 F，其中有 6 个连通分量，且其中三个至少包含三个顶点。任意两个顶点在 F 中连通，当且仅当它们的秩（如图 2(b) 所示）相同。每个连通分量中的深色顶点 F_i 及其$1 \le i \le 6$ 是 f_i F_i中的非叶顶点。（a）15 条点线边构成一个生成树 R。11 条深色点线边构成 R' R的极小子树，该子树覆盖深色顶点 f_1、.....、 f_6。（b）5 条点线边构成一个极小森林 S，其中 $|V（S）| = 9$ 的 R' 满足 $F \cup S$ 是一个生成树。 S有四个连通分量。

引理4。 若 uv 是 G 的一条边，且 $u \in U$ 和 $v \in L(F)$，则 $r(u) > r(v)$。

证明。 反设 $r（u）\le r（v）$。根据 $u \in U$，有 $r（u）< r（v）$，从而得到 $v \notin V（T_u）$。根据 $u \in U$，有 $u = p（v）$，从而得到 $T^u = T_v$。因此， $v \in U \cup V_2（F）$，这违反了 $v \in L（F）$。 □

令 F_i （其中 $i \ge 1$ ）为 F 的第 i 大连通分量。令 k 表示 F 中至少含有三个顶点的连通分量数量。根据 $d_G(a) \ge 2$，我们有 $k \ge 1$。若 uv 是 T 的一条边，且 $r(u) < r(v)$，则 $u = p(v)$ 且 $d_F(v) \ne 1$。根据我们算法的定义，对每个 $i \ge 1$，均有$|V(F_i)| \ne 2$ 成立。

- 若 $|V(F_i)| = 1$，则 F_i 在 T 中至多有两条关联边。因此， $|L(F)| \le |L(T)| + k - 1$。
- 若 $|V(F_i)| \ge 3$，则 F_i 至多有一个顶点 u 满足 $d_{F_i}(u) = 2$。因此， $|V(F_i)| \le 2 \cdot |L(F_i)| - 1$。

我们有 $U = V_0(F)$，并且

$$n - |U| = \sum_{i=1}^{k} |V(F_i)| \le \sum_{i=1}^{k} (2 \cdot |L(F_i)| - 1) = 2 \cdot |L(F)| - k \le 2 \cdot |L(T)| + k - 2.$$

为了证明 T 是 2-近似的，还需对 G的任意生成树 R 证明

$$|L(R)| \le n - |U| - k + 1. \tag{2}$$

令每个带有 $1 \le i \le k + |U|$ 的 f_i 为 $V（F_i）\setminus L（F_i）$ 的任意顶点。令 R' 为 R中包含所有 $k+|U|$ 个顶点 f_i的最小子树。我们有 $L（R'）\cap L（F）= \varnothing$ 以及 $L（R）\cap V（R'）\subseteq L（R'）$，这意味着

$$L(S) \cap L(F) = \varnothing \tag{3}$$

对于任意满足 $S \cup F$ 是 G 的生成树的 R' 的极小森林 S。根据 S 的极小性， S 的每个连通分量中的顶点具有不同的秩。根据 S 的极小性，每个连通

的连通分量 至少有两个顶点，且 S 中一个含有 j 个顶点的连通分量恰好连接 $j-1$ 的连通分量。由于 F 有 $k+|U|$ 个连通分量，因此我们有 $|V(S)| = |U| + k + t - 1$，其中 $t \geq 0$ 是 S 的连通分量数。参见图3中的示例。因此，

$$|L(R) \setminus V(S)| \leq |V(R) \setminus V(S)| = n - |U| - k - t + 1.$$

我们首先断言， S 的每个连通分量至多包含 R的一个叶节点，这意味着 $|L(R) \cap V(S)| \leq t$。因此，公式（2）可由下式推出：

$$|L(R)| = |L(R) \setminus V(S)| + |L(R) \cap V(S)| \leq n - |U| - k + 1.$$

接下来证明该断言。回顾一下， S 的每个连通分量中的所有顶点具有不同的秩。反设 S 的某个连通分量包含 R 的两个不同叶子 u 和 v，且 $r(u) < r(v)$。根据 $S \subseteq R' \subseteq R$，有 $\{u, v\} \subseteq L(S)$。令 P 为 S的 uv-路径。对于任意$\{x, y\} \in V(P)$，令 $P[x, y]$ 表示 P的 xy-路径。根据公式（3），分以下几种情形：

情形1： $u \in V_2(F)$。令 w为 P中 u的邻点。根据引理3，有 $r(w) < r(u) < r(v)$。因此， P中存在一条路径 xyz，满足 $r(y) < r(x) < r(z)$，这与引理 2 矛盾。

情形2： $u \in U$和 $v \in V_2(F)$。根据引理 2 和引理3， P中各顶点的秩从 u到 v单调递增。令 xy为 P 中满足 $V(P[u, x]) \subseteq U$ 和 $y \notin U$ 的边。根据 $x \in U$，有 $p(x) \in W_0(T_{p(x)})$。因此， $T^p(x) = T_x$，这意味着根据 $r(x) < r(y)$，有 $x = p(y)$。根据 $x = p(y)$ 和 $y \notin U$，我们有 $y \in V_2(F)$。我们有 $y = v$；否则，根据引理3， y在 $P[y, v]$ 中的邻点的秩将低于 $r(y)$。引理 1 意味着，对于 P的每个内部顶点 w，都有 $d_R(w) = 2$。因此， a 和 u 在 R中不连通，矛盾。

情形3： $\{u, v\} \subseteq U$。

- 情形 3a：集合 $\{x, z\} = \{u, v\}$ 和 P 中的内部顶点 y 满足 $V(P[y, z] - y) \subseteq U$，其中 $y \in L(F)$。由于 y 在 $P[y, z]$ 中的邻点属于 U，根据引理4，其秩高于 $r(y)$。根据引理 2， $P[x, y]$ 中各顶点的秩从 x 到 y 单调递增。因此，根据引理 3，$V(P[x, y]) \cap V_2(F) = \varnothing$。由 $x \in U$ 和 $y \in L(F)$，存在 $P[x, y]$ 中的顶点 w，它位于 $L(F)$ 中，且它在 $P[x, w]$ 中的邻点属于 U，这与引理4矛盾。

- 情形 3b：情形 3a 不成立，且 P 中有一个内部顶点 w 位于 $V_2(F)$ 中。根据引理 2 和引理3，对于每个 $x \in \{u, v\}$， $P[x, w]$ 中各顶点的秩单调递增，这意味着 P 中位于 $V_2(F)$ 的唯一顶点是 w。由于情形 3a 不成立，我们有 $V(P - w) \subseteq U$。设 y 和 z 是 w 在 P 中的邻点，且 $r(y) < r(z)$。因此， $V(T_y) \cap \{w, z\} = \varnothing$。由 $y \in U$ 和 $V_{T^p}(y)(y) \neq \varnothing$，可得 $T^p(y) = T_y$。引理 1 意味着 $d_G(y) = 2$。因此， $y = p(w)$ 且$T^y = T_w$，这意味着 $r(w) = r(z)$，矛盾。
- 情形 3c：情形 3a 和情形 3b 均不成立。因此， $V(P) \subseteq U$。由于 a和 u在 R中连通，且 $\{u, v\} \subseteq L(R)$，因此 P中存在一个内部顶点 w，使得 $d_G(w) \geq 3$。根据引理 1 和引理 2， P中各顶点的秩从 u（分别从 v）到 w单调递增。设 x 和 y是 P中 w 的邻居，且 $r(x) < r(y)$。设 z 是 G 中 w 的一个邻点，且不同于 x 和 y。– 如果 $r(x) < r(z)$，则 $V(T_x) \cap \{y, z, w\} = \varnothing$。根据 $x \in U$和$V_{T^p}(x)(x) \neq \varnothing$，我们有 $T^p(x) = T_x$。引理 1 蕴含 $d_G(x) = 2$。因此， $x = p(w)$且 $T^x = T_w$，从而 $r(w) = r(y)$，矛盾。

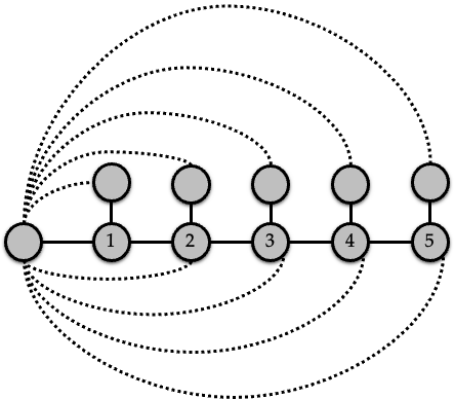


图 4：生成树 T 由实线组成。T中每个内部顶点旁的数字表示其扩展顺序。

– 如果 $r(z) < r(x)$，则 $V(T_z) \cap \{x, y, w\} = \varnothing$。根据 $w \in U$，我们有 $T^z = T_w$，从而 $r(x) = r(w) = r(y)$，矛盾。

我们的分析是紧确的。考虑图 4 中由实线和虚线组成的图 G。如果顶点 i 是第 i 个被扩展的顶点，那么我们的算法计算出的生成树 T 恰好是由实线组成的那棵树，其中 $|L(T)| = 6$。然而， G 有一棵以最高度顶点为根的生成树 R，其中 $|L(R)| = 10$，也就是 $2|L(T)| - 2$。

4 线性时间实现

我们的算法很容易实现，仅使用数组和链表即可在 $O(m)$ 时间内运行。我们维护三个（“等待中”）列表 W_2、 W_1 和 W_0，分别存放 T 的叶子；并为每个 $w \in V(G) \setminus V(T)$ 维护一个列表 U_w，存放 w 在 $V(G) \setminus V(T)$ 中的（“尚未生成树化的”）邻点。要在顶点 $u \in V(T)$ 处扩展 T，我们：(i) 将每个顶点 $v \in V_T(u)$ 插入 W_2 的末尾，并令每个 $vw \in E(G)$ 对应 $U_w = U_w \setminus \{v\}$；(ii) 令 $T = T \cup E_T(u)$。因此，整个算法中扩展 T总共需要 $O(m)$ 时间。算法在 a 处扩展初始的 $T = \{a\}$，然后进入循环，循环的每次迭代都执行以下指令。注意，根据上述实现，在顶点 $u \in V(T)$ 处扩展 T，且满足 $|V_T(u)| = 0$ 时，不会执行任何操作。

• 情形1： $W_2 \neq \varnothing$。从 W_2顶部删除一个顶点 u。如果 $|V_T(u)| = 1$，则将 u 插入到 W_1底部。否则，在 u 处扩展 T。

• 情形2： $W_2 = \varnothing$和 $W_1 \neq \varnothing$。从 W_1顶部删除一个顶点 u。如果$|V_{T \cup E_T(u)}(v_T(u))| = 1$，则将 u 插入到 W_0 底部。否则，在 u处扩展 T。

• 情形3： $W_2 = W_1 = \varnothing$ 和 $W_0 \neq \varnothing$。从 W_2底部删除一个顶点 u。在 u处扩展 T。

每个涉及 $|V_T(u)|$或$|V_{T \cup E_T(u)}(v_T(u))|$ 的条件都可以利用列表 U_w 在 $O(1)$ 时间内确定。因此，除去展开 T 所需的时间，该循环总共需要 $O(n)$ 时间。三个列表 W_2、 W_1和 W_0 确保按 $W_2(T)$、$W_1(T)$ 和 $W_0(T)$ 的顺序选取 T 的顶点 u。由于每个满足 $i \in \{0, 1, 2\}$ 的 W_i 都会保留其顶点加入 $V(T)$ 的顺序，而 W_0 则按相反顺序处理，因此 $W_0(T)$ 中的顶点会按照它们加入 $V(T)$ 的相反顺序选取。因此，上述算法的实现是正确的，运行时间为 $O(m)$。

定理 1 得证。

5 结论

我们为最大叶数生成树问题提出了一种简单的线性时间 2-近似算法。我们的近似比分析是紧的。然而，正如图 4 所示的例子所提示的，如果从度最高的顶点开始运行算法，近似比是否能够提高，值得探究。该问题是 MAX SNP-完全的，因此，近似比的非平凡下界也值得研究。

参考文献

[1]S. Arora、C. Lund、R. Motwani、M. Sudan 和 M. Szegedy。证明验证与近似问题的困难性。*ACM* 会刊，45(3):501–555，1998。doi:10.1145/278298.278306.[2] S. Arora 和 S. Safra。概率性证明检验：NP 的一种新刻画。*ACM* 会刊，45(1):70–122，1998。doi:10.1145/273865.273901.[3] H. L. Bodlaender。利用深度优先搜索进行线性时间子式测试。《算法杂志》，14(1):1–23，1993。doi:10.1006/jagm.1993.1001.[4] P. S. Bonsma 和 F. Zickfeld。多叶生成树的改进界。离散数学，312(6):1178–1194，2012。doi:10.1016/j.disc.2011.11.043.[5] N. Bousquet、T. Ito、Y. Kobayashi、H. Mizuta、P. Ouvrard、A. Suzuki 和 K. Wasa。多叶或少叶生成树的重构。载于 F. Grandoni、G. Herman 和 P. Sanders 编，第 *28* 届欧洲算法年会论文集，LIPIcs 173，第 24:1–24:15 页，2020。doi:10.4230/LIPIcs.ESA.2020.24.[6] N. Chiba、T. Nishizeki 和 N. Saito。平面图最大独立集问题的近似算法。*SIAM* 计算杂志，11(4):663–675，1982。doi:10.1137/0211055.[7] P. Diaz-Gutierrez、A. Bhushan、M. Gopi 和 R. Pajarola。用于快速交互式渲染的单条带。视觉计算机，22(6):372–386，2006。doi:10.1007/s00371-006-0018-2.[8] E. W. Dijkstra。分布式控制下的自稳定系统。*ACM* 通讯，17(11):643–644，1974。doi:10.1145/361179.361202.[9] G. Ding、T. Johnson 和 P. D. Seymour。多叶生成树。《图论杂志》，37(4):189–197，2001。doi:10.1002/jgt.1013.[10] M. R. Fellows 和 M. A. Langston。良部分序理论及其在超大规模集成电路设计组合问题中的应用。*SIAM* 离散数学杂志，5(1):117–126，1992。doi:10.1137/0405010。

[11] G. Galbiati、F. Maffioli 和 A. Morzenti。关于最大叶生成树问题可近似性的一则简短说明。《信息处理快报》，52(1):45–49，1994。doi:10.1016/0020-0190(94)90139-2.[12] M. R. Garey 和 D. S. Johnson。《计算机与难解性：*NP* 完全性理论指南》。W. H. Freeman，旧金山，1979年.[13] J. Griggs、D. J. Kleitman 和 A. Shastri。三次图中的多叶生成树。《图论杂志》，13(6):669–695，1989年。doi:10.1002/jgt.3190130604.[14]T. Ito、E. D. Demaine、X. Zhou 和 T. Nishizeki。供需图划分的可近似性。《离散算法杂志》，6(4):627–650，2008年。doi:10.1016/j.jda.2008.03.002.[15] M. Kawabata 和 T. Nishizeki。图的生成分布树。《电子信息通信学会信息与系统汇刊》，97-D(3):406–412，2014年。doi:10.1587/transinf.E97.D.406.[16] D. J. Kleitman 和 D. B. West。多叶生成树。*SIAM* 离散数学杂志，4(1):99–106，1991年。doi:10.1137/0404010.[17] I.-C. Liao。最大叶数生成树的改进近似算法。硕士论文，国立中正大学计算机科学与信息工程系，2001年。https://hdl.handle.net/11296/z535d3.[18] H.-I. Lu 和 R. Ravi。局部优化的威力：最大叶数生成树的近似算法。载于第 *30* 届 *Allerton* 通信、控制与计算年会论文集，第 533–542 页，1992年.[19] H.-I. Lu 和 R. Ravi。近线性时间近似最大叶数生成树。《算法杂志》，29(1):132–141，1998。doi:10.1006/jagm.1998.0944.[20] K. Miura、M. Azuma 和 T. Nishizeki。平面图的典范分解、实现器、Schnyder 标记和有序生成树。《计算机科学基础国际期刊》，16(1):117–141，2005年。doi:10.1142/S0129054105002905.[21] K. Miura、D. Takahashi、S. Nakano 和 T. Nishizeki。寻找四连通平面图中四棵独立生成树的线性时间算法。《计算机科学基础国际期刊》，10(2):195–210，1999年。doi:10.1142/S0129054199000149.[22] T. Nishizeki、T. Asano 和 T. Watanabe。极大平面图哈密顿游走问题的近似算法。《离散应用数学》，5(2):211–222，1983年。doi:10.1016/0166-218X(83)90042-2.[23]Y. Obata 和 T. Nishizeki。图带宽连续多重着色的高效近似算法。《理论计算机科学》，607:208–220，2015年。doi:10.1016/j.tcs.2015.07.052.[24] C. H. Papadimitriou 和 M. Yannakakis。优化、近似与复杂性类。载于第*20*届 *ACM* 计算理论年会论文集，第 229–234 页，1988年。doi:10.1145/62212.62233。

[25] C. Payan、M. Tchuente 和 N. H. Xuong。具有最多悬挂顶点的树。《离散数学》，49(3):267–273，1984。doi:10.1016/0012-365X(84)90163-8.[26] F. Rosamond。最大叶生成树。载于 M.-Y. Kao（编），《算法百科全书》，第 1211–1215 页。施普林格纽约，2016。doi:10.1007/978-1-4939-2864-4 228。[27] R. Solis-Oba。寻找叶子数最多的生成树的 2-近似算法。载于 G. Bilardi、G. F. Italiano、A. Pietracaprina 和 G. Pucci（编），《第六届欧洲算法年会论文集》，《计算机科学讲义》第 1461 卷，第 441–452 页，1998。doi:10.1007/3-540-68530-8 37。[28]R. Solis-Oba、P. S. Bonsma 和 S. Lowski。寻找叶子数最多的生成树的 2-近似算法。《算法学》，77(2):374–388，2017。doi:10.1007/s00453-015-0080-0。[29] J. A. Storer。构造三次图的完全生成树。《信息处理快报》，13(1):8–11，1981。doi:10.1016/0020-0190(81)90141-1。[30] K. Takamizawa、T. Nishizeki 和 N. Saito。寻找图中短闭合生成游走的算法。《网络》，10(3):249–263，1980。doi:10.1002/net.3230100306。[31] M. Tchuente。计算机网络中的自稳定。《RAIRO. 理论计算机科学》´，15(1):47–66，1981。http://www.numdam.org/item/ITA 1981 15 1 47 0/。