entropy

MDPI

Article

An Efficient Algorithm to Count Tree-Like Graphs with a Given Number of Vertices and Self-Loops

Naveed Ahmed Azam *, Aleksandar Shurbevski and Hiroshi Nagamochi

Department of Applied Mathematics and Physics, Kyoto University, Kyoto 606-8502, Japan; shurbevski@amp.i.kyoto-u.ac.jp (A.S.); nag@amp.i.kyoto-u.ac.jp (H.N.)

* Correspondence: azam@amp.i.kyoto-u.ac.jp

Received: 23 July 2020; Accepted: 18 August 2020; Published: 22 August 2020

Abstract: Graph enumeration with given constraints is an interesting problem considered to be one of the fundamental problems in graph theory, with many applications in natural sciences and engineering such as bio-informatics and computational chemistry. For any two integers $n \geq 1$ and $\Delta \geq 0$, we propose a method to count all non-isomorphic trees with n vertices, Δ self-loops, and no multi-edges based on dynamic programming. To achieve this goal, we count the number of non-isomorphic rooted trees with n vertices, Δ self-loops and no multi-edges, in $\mathcal{O}(n^2(n + \Delta(n + \Delta \cdot \min\{n, \Delta\})))$ time and $\mathcal{O}(n^2(\Delta^2 + 1))$ space, since every tree can be uniquely viewed as a rooted tree by either regarding its unicentroid as the root, or in the case of bicentroid, by introducing a virtual vertex on the bicentroid and assuming the virtual vertex to be the root. By this result, we get a lower bound and an upper bound on the number of tree-like polymer topologies of chemical compounds with any "cycle rank".

Keywords: trees; chemical graphs; enumeration; dynamic programming; polymer topology

1. Introduction

Counting and generation of discrete objects are two fundamental problems in combinatorial mathematics and have many applications in the fields of natural science and engineering, such as computational chemistry and bioinformatics. The counting problem asks to count all possible objects under given constraints. On the other hand, the generation problem asks to list all possible objects under given constraints. One of the notable advantages of the counting problem is that we can know the size of the solution space before generating all solutions.

Different kinds of enumeration methods are used to solve counting and generation problems, where branching algorithms and Polya's enumeration theorem are the two most commonly used methods for these problems. In branching algorithms, the computation is performed by following a computation tree, and the required solutions are attained at the leaves of the computation tree. It is important to mention that the branching algorithms can only count all solutions after generating each one of them, and therefore they are inefficient for the problem where we first want to know the size of the solution space before the generation of solutions.

The well-known Polya's enumeration theorem [1,2] is used for counting all distinct objects. The idea of this method is to use the cyclic index of the group of symmetries of the underlying object to develop a generating function, which is then used to count all possible objects. Note that finding the group of symmetries and its cyclic index is a challenging task, which may make the use of Polya's theorem harder for some problems.

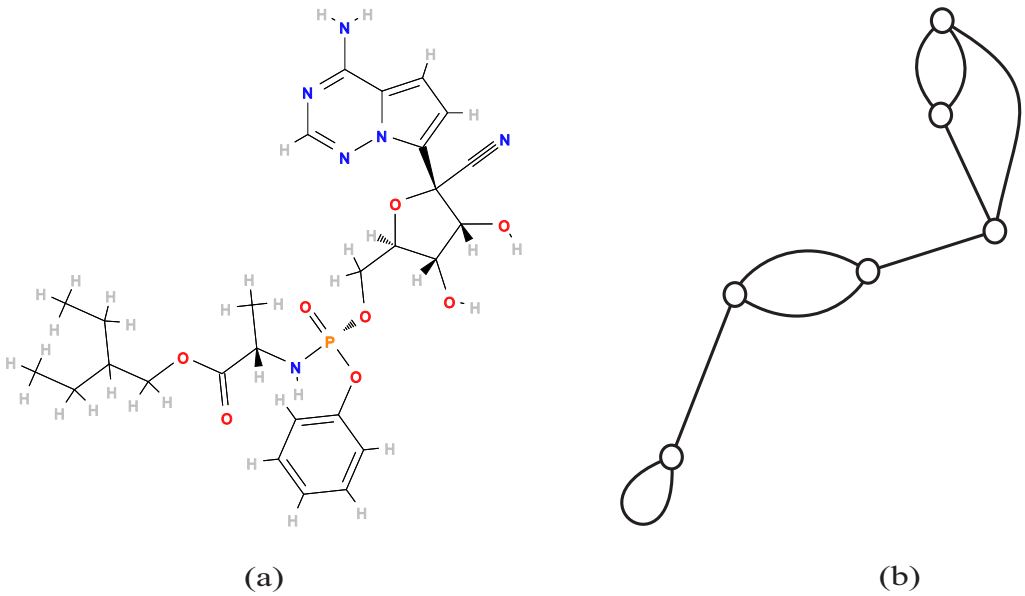
The drawback of branching algorithms discussed above and the difficulty of using Polya's theorem necessitate the exploration of new enumeration methods to solve counting problems efficiently. For an enumeration method, it is necessary to satisfy the following three conditions:

(i) Consider all solutions: The method does not miss any of the required objects;
(ii) Avoid duplication: The method does not count and generate isomorphic objects; and
(iii) Low computational complexity: The method can count and generate all solutions in low time and space complexity.

Designing such a method is not an easy task, because of the underlying symmetries and the computation difficulty for their detection.

Counting and generation of chemical compounds have a long history and numerous applications in designing novel drugs [3–8] and structure elucidation [9]. The problem of counting and generation of chemical compounds can be viewed as the problem of enumerating graphs with given constraints. There are several available chemical compound enumeration tools [10–12]. We can divide these tools into two classes. One class of enumeration tools treats general graph structures [10,12]. In the other class, the tools are focused on enumerating some restricted chemical compounds. One such tool is Enumol2 [11]. Enumeration of restricted chemical compounds with specialized tools is more efficient than with the tools which use general graph structures. This led to a new trend of developing efficient enumeration of restricted chemical compounds in the field of chemoinformatics [13].

A polymer is a large molecule with interesting chemical properties consisting of many sub-molecules. From a graph-theoretic perspective, we represent the structure of a polymer with a graph G called *polymer topology*, possibly with self-loops and multi-edges, such that G is connected and the degree of each vertex in G is at least three [14]. For a chemical graph, we get its polymer topology by repeatedly removing the vertices of degree one and two. For example, the polymer topology of Remdesivir $C_{27}H_{35}N_6O_8P$ Figure 1a, a potential candidate of treatment for COVID-19, is illustrated in Figure 1b.



(a) (b)

Figure 1. The chemical compound Remdesivir $C_{27}H_{35}N_6O_8P$ and its polymer topology: (**a**) chemical structure of Remdesivir $C_{27}H_{35}N_6O_8P$ obtained from the PubChem database; (**b**) the polymer topology of Remdesivir with six vertices, two multi-edges of multiplicity 2, one self-loop and cycle rank 4.

Tezuka and Oike [15] pointed out that a classification of polymer topologies will lay a foundation for the elucidation of structural relationships between different macro-chemical molecules and their synthetic pathways. Different kinds of graph-theoretic approaches have been applied to classify and enumerate polymer topologies [16,17]. For a connected graph G, possibly with self-loops and multi-edges, the *cycle rank* is defined to be the number of edges that must be removed to get a simple spanning tree of G. Recently, Haruna et al. [14] proposed a method to enumerate all polymer topologies with cycle rank up to five.

Notice that trees with no multi-edges but with $\Delta \geq 0$ self-loops have cycle rank Δ and include all polymer topologies with the said structure. Therefore, it is of interest to count and generate all trees with no multi-edges and a given number of vertices and self-loops.

We use dynamic programming (DP) to count all mutually non-isomorphic trees with n vertices, Δ self-loops and no multi-edges. The basic idea of DP is to partition the original problem into subproblems that satisfy some recursive relations, and the union of their solution sets is equal to the solution set of the original problem. Unlike branching algorithms and Polya's theorem, the main advantage of using the DP is that we can count all non-isomorphic structures without their generation and calculation of their group of symmetries. As an application of our results, we get lower and upper bounds on the number of tree-like polymer topologies with self-loops of a given cycle rank.

The rest of the paper is organized as follows: Section 2 reviews some notions and results related to graph theory. Section 3 explains our tree counting method. Section 4 makes some concluding remarks.

2. Preliminaries

Throughout this draft, the term *graph* stands for an undirected graph with no multi-edges and possibly with self-loops unless stated otherwise. Let G be a graph. We denote an edge between two vertices u and v in G by uv $(= vu)$. Let $V(G)$ and $E(G)$ denote the vertex set and edge set of G, respectively. Let $s(G)$ denote the number of self-loops in G. For a vertex $v \in V(G)$, we denote by $s(v)$ the number of self-loops on the vertex v. For a vertex v in G, let $N_G(v)$ denote the set of vertices incident to v except v itself and the *degree* $\deg_G(v)$ of v in G is defined to be $|N_G(v)|$. A graph H with the properties $V(H) \subseteq V(G)$ and $E(G) \subseteq E(G)$ is called a *subgraph* of G. A *simple path* between two distinct vertices $u, v \in V(G)$ is defined to be a subgraph P of G with vertex set $V(P) = \{u = w_1, w_2, \ldots, v = w_k\}$ and edge set $E(P) = \{w_i w_{i+1} \mid 1 \leq i \leq k-1\}$. A graph is called a *connected graph* if there is a path between any two distinct vertices in the graph. A *connected component* of a graph G is defined to be a maximal connected subgraph H of G, i.e., for any vertex $v \in V(G) \setminus V(H)$ it holds that every subgraph with the vertex set $V(H) \cup \{v\}$ is disconnected.

By Jordan [18], any simple tree with $n \geq 1$ vertices has either a unique vertex or edge, the removal of which creates connected components with at most $\lfloor (n-1)/2 \rfloor$ or exactly $n/2$ vertices, respectively. Such a vertex is called the *unicentroid*, the edge is called the *bicentroid*, and collectively they are called the *centroid* of the tree. It is important to note that there exits a bicentroid only for trees with an even number of vertices. A tree with a fixed vertex r is called a *rooted tree* with root r. Note that any tree can be uniquely viewed as a rooted tree by either regarding its unicentroid as the root, or in the case of a bicentroid, by introducing a virtual vertex on the bicentroid and assuming the virtual vertex as the root.

Let H be a rooted tree. Let r_H denote the root of H. For any two distinct vertices $u, v \in V(H)$, let $P_H(u, v)$ denote the unique simple path between them in H. For a vertex $v \in V(H) \setminus \{r_H\}$, we define the *ancestors* of v to be the vertices on the path $P_H(v, r_H)$ other than v. If u is an ancestor of v, then we call v a *descendant* of u. For a vertex $v \in V(H) \setminus \{r_H\}$, the *parent* $p(v)$ of v is defined to be the ancestor u of v such that $u \in N_H(v)$. We call the vertex v a *child* of $p(v)$. Two vertices with the same parent in H are called *siblings*. For a vertex $v \in V(H)$, let H_v denote the subtree of H rooted at v induced by v and its descendants.

Two rooted trees T and H are called *isomorphic* if there exists a bijection $\sigma : V(T) \to V(H)$ such that

(i) $\sigma(r_T) = r_H$;
(ii) for each vertex $v \in V(T)$, it holds that $s(v) = s(\sigma(v))$; and
(iii) for any two vertices $u, v \in V(T)$, it holds that $uv \in E(T)$ if and only if $\sigma(u)\sigma(v) \in E(H)$.

For any two integers $n \geq 1$ and $\Delta \geq 0$, let $\mathcal{H}(n, \Delta)$ denote a maximal set of mutually non-isomorphic rooted trees with n vertices and Δ self-loops, and we define $h(n, \Delta) \triangleq |\mathcal{H}(n, \Delta)|$.

3. Counting Tree-Like Graphs with a Given Number of Vertices and Self-Loops

We develop a method to compute for any two integers $n \geq 1$ and $\Delta \geq 0$, the size $h(n, \Delta)$ of a maximal set $\mathcal{H}(n, \Delta)$ of mutually non-isomorphic rooted trees with n vertices and Δ self-loops; i.e., we are interested in the following problem:

Counting Problem
Input: Two integers $n \geq 1$ and $\Delta \geq 0$.
Output: $h(n, \Delta)$.

We solve this problem by using dynamic programming based on the information of the number of vertices and self-loops in the subtrees rooted at the children of the root of each tree in $\mathcal{H}(n, \Delta)$. We define the following notions.

Let $n \geq 1$ and $\Delta \geq 0$ be any two integers. For each tree $H \in \mathcal{H}(n, \Delta)$, we define

$$\begin{aligned} \mathrm{Max_v}(H) &\triangleq \max\{\{|V(H_v)| \mid v \in N_H(r_H)\} \cup \{0\}\}, \\ \mathrm{Max_s}(H) &\triangleq \max\{\{s(H_v) \mid v \in N_H(r_H), |V(H_v)| = \mathrm{Max_v}(H)\} \cup \{0\}\}. \end{aligned}$$

Note that for any tree $H \in \mathcal{H}(1, \Delta)$, it holds that $\mathrm{Max_v}(H) = 0$ and $\mathrm{Max_s}(H) = 0$.

Let $m, d \geq 0$ be any two integers. We define

$$\mathcal{H}(n, \Delta, m_{\leq}, d_{\leq}) \triangleq \{H \in \mathcal{H}(n, \Delta) \mid \mathrm{Max_v}(H) \leq m, \mathrm{Max_s}(H) \leq d\}.$$

Observe that by the definition of $\mathcal{H}(n, \Delta, m_{\leq}, d_{\leq})$ it holds that

(i) $\mathcal{H}(n, \Delta, m_{\leq}, d_{\leq}) = \mathcal{H}(n, \Delta, n-1_{\leq}, d_{\leq})$ if $m \geq n$;
(ii) $\mathcal{H}(n, \Delta, m_{\leq}, d_{\leq}) = \mathcal{H}(n, \Delta, m_{\leq}, \Delta_{\leq})$ if $d \geq \Delta + 1$; and
(iii) $\mathcal{H}(n, \Delta) = \mathcal{H}(n, \Delta, n-1_{\leq}, \Delta_{\leq})$.

Therefore, from now on, we assume that $m \leq n-1$ and $d \leq \Delta$. Further, by the definition of $\mathcal{H}(n, \Delta, m_{\leq}, d_{\leq})$ it holds that $\mathcal{H}(n, \Delta, m_{\leq}, d_{\leq}) \neq \emptyset$ (resp., $\mathcal{H}(n, \Delta, m_{\leq}, d_{\leq}) = \emptyset$) if "$n = 1$" or "$n - 1 \geq m \geq 1$" (resp., otherwise ($n \geq 2$ and $m = 0$)).

We define

$$\mathcal{H}(n, \Delta, m_{=}, d_{\leq}) \triangleq \{H \in \mathcal{H}(n, \Delta, m_{\leq}, d_{\leq}) \mid \mathrm{Max_v}(H) = m\}.$$

It follows from the definition of $\mathcal{H}(n, \Delta, m_{=}, d_{\leq})$ that $\mathcal{H}(n, \Delta, m_{=}, d_{\leq}) \neq \emptyset$ (resp., $\mathcal{H}(n, \Delta, m_{=}, d_{\leq}) = \emptyset$) if "$n = 1$" or "$n - 1 \geq m \geq 1$" (resp., otherwise ($n \geq 2$ and $m = 0$)). Further we have the following relation:

$$\mathcal{H}(n, \Delta, m_{\leq}, d_{\leq}) = \mathcal{H}(n, \Delta, 0_{=}, d_{\leq}) \text{ if } m = 0, \tag{1}$$

$$\mathcal{H}(n, \Delta, m_{\leq}, d_{\leq}) = \mathcal{H}(n, \Delta, m-1_{\leq}, d_{\leq}) \cup \mathcal{H}(n, \Delta, m_{=}, d_{\leq}) \text{ if } m \geq 1, \tag{2}$$

where $\mathcal{H}(n, \Delta, m-1_{\leq}, d_{\leq}) \cap \mathcal{H}(n, \Delta, m_{=}, d_{\leq}) = \emptyset$ for $m \geq 1$.

Next we define

$$\mathcal{H}(n, \Delta, m_{=}, d_{=}) \triangleq \{H \in \mathcal{H}(n, \Delta, m_{=}, d_{\leq}) \mid \mathrm{Max_s}(H) = d\}.$$

Note that if "$n = 1$ and $d = 0$" or "$n - 1 \geq m \geq 1$" (resp., otherwise ("$n = 1$ and $d \geq 1$" or "$n \geq 2$ and $m = 0$")), then by the definition of $\mathcal{H}(n, \Delta, m_=, d_=)$ it holds that $\mathcal{H}(n, \Delta, m_=, d_=) \neq \emptyset$ (resp., $\mathcal{H}(n, \Delta, m_=, d_=) = \emptyset$). Furthermore, we get the following relation for $\mathcal{H}(n, \Delta, m_=, d_\leq)$:

$$\mathcal{H}(n, \Delta, m_=, d_\leq) = \mathcal{H}(n, \Delta, m_=, 0_=) \text{ if } d = 0, \tag{3}$$

$$\mathcal{H}(n, \Delta, m_=, d_\leq) = \mathcal{H}(n, \Delta, m_=, d - 1_\leq) \cup \mathcal{H}(n, \Delta, m_=, d_=) \text{ if } d \geq 1, \tag{4}$$

where $\mathcal{H}(n, \Delta, m_=, d - 1_\leq) \cap \mathcal{H}(n, \Delta, m_=, d_=) = \emptyset$ for $d \geq 1$.

Let $n - 1 \geq m \geq 0$, and $\Delta \geq d \geq 0$ be four integers. Let $h(n, \Delta, m_\leq, d_\leq)$, $h(n, \Delta, m_=, d_\leq)$ and $h(n, \Delta, m_=, d_=)$ denote the number of elements in the families $\mathcal{H}(n, \Delta, m_\leq, d_\leq)$, $\mathcal{H}(n, \Delta, m_=, d_\leq)$ and $\mathcal{H}(n, \Delta, m_=, d_=)$, respectively. We discuss recursive relations for $h(n, \Delta, m_\leq, d_\leq)$ and $h(n, \Delta, m_=, d_\leq)$ in Lemma 1.

Lemma 1. *For any four integers $n - 1 \geq m \geq 0$, and $\Delta \geq d \geq 0$, it holds that*

(i) $h(n, \Delta, m_\leq, d_\leq) = h(n, \Delta, 0_=, d_\leq)$ *if* $m = 0$;
(ii) $h(n, \Delta, m_\leq, d_\leq) = h(n, \Delta, m - 1_\leq, d_\leq) + h(n, \Delta, m_=, d_\leq)$ *if* $m \geq 1$;
(iii) $h(n, \Delta, m_=, d_\leq) = h(n, \Delta, m_=, 0_=)$ *if* $d = 0$; *and*
(iv) $h(n, \Delta, m_=, d_\leq) = h(n, \Delta, m_=, d - 1_\leq) + h(n, \Delta, m_=, d_=)$ *if* $d \geq 1$.

Proof. The case (i) follows by Equation (1). The case (ii) follows by Equation (2) and the fact that for $m \geq 1$ it holds that $\mathcal{H}(n, \Delta, m - 1_\leq, d_\leq) \cap \mathcal{H}(n, \Delta, m_=, d_\leq) = \emptyset$. By Equation (3) the case (iii) follows. The case (iv) follows by Equation (4) and the fact that for $d \geq 1$ it holds that $\mathcal{H}(n, \Delta, m_=, d - 1_\leq) \cap \mathcal{H}(n, \Delta, m_=, d_=) = \emptyset$. □

Next we discuss some boundary conditions for our DP to compute $h(n, \Delta)$.

Lemma 2. *For any four integers $n - 1 \geq m \geq 0$, and $\Delta \geq d \geq 0$, it holds that*

(i) $h(n, \Delta, 0_=, d_=) = 1$ *(resp., $h(n, \Delta, 0_=, d_=) = 0$) if $n = 1$ and $d = 0$ (resp., otherwise ("$n = 1$ and $d \geq 1$" or "$n \geq 2$"));*
(ii) $h(n, \Delta, 0_=, d_\leq) = h(n, \Delta, 0_\leq, d_\leq) = 1$ *(resp., $h(n, \Delta, 0_=, d_\leq) = h(n, \Delta, 0_\leq, d_\leq) = 0$) if $n = 1$ (resp., otherwise ($n \geq 2$));*
(iii) $h(n, \Delta, 1_=, d_=) = 1$ *if "$n = 2$" or "$n \geq 3$ and $d = 0$"; and*
(iv) $h(n, \Delta, 1_=, d_\leq) = h(n, \Delta, 1_\leq, d_\leq) = d + 1$ *if "$n = 2$" or "$n \geq 3$ and $d = 0$".*

Proof. (i) The result follows from the definition of $\mathcal{H}(n, \Delta, 0_=, d_=)$, since a tree H with $\max_{\mathrm{v}}(H) = 0$ exists if and only if $|V(H)| = 1$ and $\max_{\mathrm{s}}(H) = 0$.

(ii) By Lemma 1(i), (ii) and (iv) it holds that $h(n, \Delta, 0_\leq, d_\leq) = h(n, \Delta, 0_=, d_\leq) = \sum_{p=0}^{d} h(n, \Delta, 0_=, p_=)$. This and Lemma 2(i) imply the required result.

(iii) When $n \geq 2$, then for any tree $H \in \mathcal{H}(n, \Delta, 1_=, d_=)$ it holds that $|N_H(r_H)| = n - 1$. Thus for each $v \in N_H(r_H)$ it holds that $|V(H_v)| = 1$ and $s(H_v) = d$ if "$n = 2$" or "$n \geq 3$ and $d = 0$", i.e., $H_v \in \mathcal{H}(1, d, 0_\leq, d_\leq)$. But by Lemma 2(ii) it holds that $h(1, d, 0_\leq, d_\leq) = 1$. Hence we have the required result.

(iv) Let "$n = 2$" or "$n \geq 3$ and $d = 0$". By Lemma 1(iii) and (iv) it holds that $h(n, \Delta, 1_=, d_\leq) = \sum_{p=0}^{d} h(n, \Delta, 1_=, p_=)$. This and Lemma 2(iii) imply that

$$h(n, \Delta, 1_=, d_\leq) = d + 1. \tag{5}$$

Furthermore, by Lemma 1(iii) it holds that $h(n,\Delta,1_{\leq},d_{\leq}) = h(n,\Delta,0_{=},d_{\leq}) + h(n,\Delta,1_{=},d_{\leq})$. By Lemma 2(ii), we have $h(n,\Delta,1_{\leq},d_{\leq}) = h(n,\Delta,1_{=},d_{\leq})$. Hence the result follows by Equation (5).

□

By Lemma 2, we can get that $h(1,\Delta) = 1$ and $h(2,\Delta) = \Delta + 1$. Furthermore, Lemma 1(i)–(iv) give recursive relations for $h(n,\Delta,m_{\leq},d_{\leq})$ and $h(n,\Delta,m_{=},d_{\leq})$ which depend on $h(n,\Delta,m_{=},d_{=})$. Thus for $n \geq 3$, $m \geq 1$, and $\Delta \geq d \geq 0$, our next goal is to develop a recursive relation for $h(n,\Delta,m_{=},d_{=})$. For any tree $H \in \mathcal{H}(n,\Delta,m_{=},d_{=})$ and any vertex $v \in N_H(r_H)$, the subtree H_v of H satisfies exactly one of the following three conditions:

(C-1) $|V(H_v)| = m$ and $s(H_v) = d$.
(C-2) $|V(H_v)| = m$ and $0 \leq s(H_v) < d$.
(C-3) $|V(H_v)| < m$ and $0 \leq s(H_v) \leq \Delta$.

For any tree $H \in \mathcal{H}(n,\Delta,m_{=},d_{=})$, we define the *residual tree* of H to be the subtree of H rooted at r_H induced by the vertices $V(H) \setminus \bigcup_{\substack{v \in N_H(r_H),\\ H_v \in \mathcal{H}(m,d,m-1_{\leq},d_{\leq})}} V(H_v)$. Note that the residual tree of a tree H has at least one vertex, i.e., the root of H. We give an illustration of a residual tree in Figure 2.

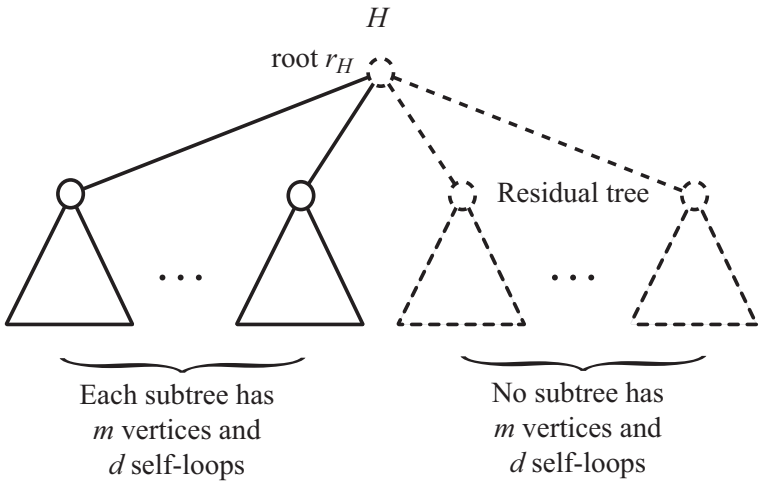


Figure 2. An illustration of a residual tree, where $H \in \mathcal{H}(n,\Delta,m_{=},d_{=})$ and the residual tree of H is shown by dashed lines.

Lemma 3. *For any four integers $n \geq 3$, $m \geq 1$, and $\Delta \geq d \geq 0$, and a tree $H \in \mathcal{H}(n,\Delta,m_{=},d_{=})$, let $q = |\{v \in N_H(r_H) \mid H_v \in \mathcal{H}(m,d,m-1_{\leq},d_{\leq})\}|$. Then it holds that*

(i) $1 \leq q \leq \lfloor (n-1)/m \rfloor$ *with* $q \leq \lfloor \Delta/d \rfloor$ *when* $d \geq 1$.
(ii) *The residual tree of H belongs to exactly one of the families $\mathcal{H}(n-qm,\ \Delta-dq, m_{=}, \min\{\Delta-dq,\ d-1\}_{\leq})$ and $\mathcal{H}(n-qm, \Delta-dq, \min\{n-qm-1, m-1\}_{\leq}, \Delta-dq_{\leq})$.*

Proof. (i) Since $H \in \mathcal{H}(n,\Delta,m_{=},d_{=})$, there exists at least one vertex $v \in N_H(r_H)$ such that $H_v \in \mathcal{H}(m,d,m-1_{\leq},d_{\leq})$. This implies that $q \geq 1$. Also, it holds that $n-1 \geq mq$ and $\Delta \geq dq$. This implies that $q \leq \lfloor (n-1)/m \rfloor$ with $q \leq \lfloor \Delta/d \rfloor$ when $d \geq 1$.

(ii) Let K denote the residual tree of H. By the definition of K it holds that $K \in \mathcal{H}(n - mq, \Delta - dq, n - mq - 1_{\leq}, \Delta - dq_{\leq})$. Furthermore, for each vertex $v \in N_H(r_H) \cap V(K)$, the tree H_v satisfies exactly one of the conditions (C-2) and (C-3). Now, if there exists a vertex $v \in N_H(r_H) \cap V(K)$ such that H_v satisfies condition (C-2), then $d - 1 \geq 0$, and hence $K \in \mathcal{H}(n - qm, \Delta - dq, m_{=}, \min\{\Delta - dq, d-1\}_{\leq})$. On the other hand, if condition (C-2) does not hold for any $v \in N_H(r_H) \cap V(K)$; i.e., either $N_H(r_H) \cap V(K) = \emptyset$ or for each $v \in N_H(r_H) \cap V(K)$ it holds that $|V(H_v)| \leq \min\{n - qm - 1, m - 1\}$ and $0 \leq s(H_v) \leq \Delta - dq$, then by the definition of K it holds that $K \in \mathcal{H}(n - qm, \Delta - dq, \min\{n - qm - 1, m - 1\}_{\leq}, \Delta - dq_{\leq})$. This completes the proof.

□

For any five integers $n \geq 3$, $m \geq 1$, $\Delta \geq d \geq 0$, and $t \geq 0$, let $c(m, d; t) \triangleq \binom{h(m,d,m-1_{\leq},d_{\leq})+t-1}{t}$ denote the number of combinations with repetition of t trees from the family $\mathcal{H}(m, d, m - 1_{\leq}, d_{\leq})$. In Lemma 4, we give a recursive relation for $h(n, \Delta, m_{=}, d_{=})$.

Lemma 4. *For any five integers $n \geq 3$, $m \geq 1$, $\Delta \geq d \geq 0$, and q, such that $1 \leq q \leq \lfloor (n-1)/m \rfloor$ with $q \leq \lfloor \Delta/d \rfloor$ when $d \geq 1$, it holds that*

(i) $h(n, \Delta, m_{=}, d_{=}) = \sum_q c(m, d; q) h(n - qm, \Delta, \min\{n - qm - 1, m - 1\}_{\leq}, \Delta_{\leq})$ *if* $d = 0$;

(ii) $h(n, \Delta, m_{=}, d_{=}) = \sum_q c(m, d; q)(h(n - qm, \Delta - dq, m_{=}, \min\{\Delta - dq, d - 1\}_{\leq}) + h(n - qm, \Delta - dq, \min\{n - qm - 1, m - 1\}_{\leq}, \Delta - dq_{\leq}))$ *if* $d \geq 1$;

(iii) $h(n, \Delta, m_{=}, d_{=}) = \sum_q c(m, d; q - 1)((h(m, d, m - 1_{\leq}, d_{\leq}) + q - 1)/q) h(n - qm, \Delta, \min\{n - qm - 1, m - 1\}_{\leq}, \Delta_{\leq})$ *if* $d = 0$; *and*

(iv) $h(n, \Delta, m_{=}, d_{=}) = \sum_q c(m, d; q - 1)((h(m, d, m - 1_{\leq}, d_{\leq}) + q - 1)/q)(h(n - qm, \Delta - dq, m_{=}, \min\{\Delta - dq, d - 1\}_{\leq}) + h(n - qm, \Delta - dq, \min\{n - qm - 1, m - 1\}_{\leq}, \Delta - dq_{\leq}))$ *if* $d \geq 1$.

Proof. Let H be a tree in the family $\mathcal{H}(n, \Delta, m_{=}, d_{=})$. By Lemma 3(i), there exists a unique integer q, $1 \leq q \leq \lfloor (n-1)/m \rfloor$ with $q \leq \lfloor \Delta/d \rfloor$ when $d \geq 1$, such that there are exactly q subtrees H_v with $v \in N_H(r_H)$ and $H_v \in \mathcal{H}(m, d, m - 1_{\leq}, d_{\leq})$. Further, by Lemma 3(ii) the residual tree of H belongs to the family $\mathcal{H}(n - qm, \Delta, \min\{n - qm - 1, m - 1\}_{\leq}, \Delta_{\leq})$ (resp., $\mathcal{H}(n - qm, \Delta - dq, m_{=}, \min\{\Delta - dq, d - 1\}_{\leq}) \cup \mathcal{H}(n - qm, \Delta - dq, \min\{n - qm - 1, m - 1\}_{\leq}, \Delta - dq_{\leq})$) if $d = 0$ (resp., otherwise). Note that $\mathcal{H}(n - qm, \Delta - dq, m_{=}, \min\{\Delta - dq, d - 1\}_{\leq}) \cap \mathcal{H}(n - qm, \Delta - dq, \min\{n - qm - 1, m - 1\}_{\leq}, \Delta - dq_{\leq}) = \emptyset$. This implies that for a fixed integer q in the range given in the lemma, the number of trees K in the family $\mathcal{H}(n, \Delta, m_{=}, d_{=})$ with exactly q subtrees $K_v \in \mathcal{H}(m, d, m - 1_{\leq}, d_{\leq})$, for $v \in N_K(r_K)$, are

(a) $c(m, d; q) h(n - qm, \Delta, \min\{n - qm - 1, m - 1\}_{\leq}, \Delta_{\leq})$ if $d = 0$; and

(b) $c(m, d; q)(h(n - qm, \Delta - dq, m_{=}, \min\{\Delta - dq, d - 1\}_{\leq}) + h(n - qm, \Delta - dq, \min\{n - qm - 1, m - 1\}_{\leq}, \Delta - dq_{\leq}))$ if $d \geq 1$.

Note that, for $m = 1$ and $d = 0$, we have $1 \leq q \leq n - 1$, and by Lemma 2(ii) it holds that $h(n - q, \Delta, 0_{\leq}, \Delta_{\leq}) = 0$ (resp., $h(n - q, \Delta, 0_{\leq}, \Delta_{\leq}) = 1$), if $1 \leq q \leq n - 2$ (resp., otherwise (if $q = n - 1$)). This implies that any tree $H \in \mathcal{H}(n, \Delta, 1_{=}, 0_{=})$ has exactly $q = n - 1$ subtrees $H_v \in \mathcal{H}(1, 0, 0_{\leq}, 0_{\leq})$, for $v \in N_H(r_H)$. However, observe that for each integer $m \geq 2$ or $d \geq 1$, and q satisfying the conditions given in the lemma, there exists at least one tree $H \in \mathcal{H}(n, \Delta, m_{=}, d_{=})$ such that H has exactly q subtrees $H_v \in \mathcal{H}(m, d, m - 1_{\leq}, d_{\leq})$, for $v \in N_H(r_H)$. Hence, this and case (a) (resp., case (b)) imply Lemma 4(i) (resp., Lemma 4(ii)).

Furthermore, it holds that

$$\begin{aligned} c(m,d;q) &= \frac{(h(m,d,m-1_{\leq},d)+q-1)!}{(h(m,d,m-1_{\leq},d)-1)!q!} \\ &= \frac{(h(m,d,m-1_{\leq},d)+q-2)!}{(h(m,d,m-1_{\leq},d)-1)!(q-1)!} \times \frac{(h(m,d,m-1_{\leq},d)+q-1)}{q} \\ &= c(m,d;q-1) \times \frac{(h(m,d,m-1_{\leq},d_{\leq})+q-1)}{q}. \end{aligned}$$

Hence, Lemma 4(iii) and (iv) follow from Lemma 4(i) and (ii), respectively. □

We design a DP algorithm to compute $h(n,\Delta)$ based on the recursive structures of $h(n,\Delta,m_{\leq},d_{\leq})$, $h(n,\Delta,m_{=},d_{\leq})$ and $h(n,\Delta,m_{=},d_{=})$, $0 \leq m \leq n-1$ and $0 \leq d \leq \Delta$, as given in Lemmas 1 and 4, where $h(n,\Delta) = h(n,\Delta,n-1_{\leq},\Delta_{\leq})$ for $n \geq 1$ and $\Delta \geq 0$.

Lemma 5. *For any four integers $n-1 \geq m \geq 0$, and $\Delta \geq d \geq 0$, $h(n,\Delta,m_{\leq},d_{\leq})$ can be obtained in $\mathcal{O}(nm(n+\Delta(n+d\cdot\min\{n,\Delta\})))$ time and $\mathcal{O}(nm(\Delta(d+1)+1))$ space.*

The proof of Lemma 5 follows from Algorithm 1 and Lemma 6.

Corollary 1. *For any two integers $n \geq 1$ and $\Delta \geq 0$, $h(n,\Delta,n-1_{\leq},\Delta_{\leq})$ can be obtained in $\mathcal{O}(n^2(n+\Delta(n+\Delta\cdot\min\{n,\Delta\})))$ time and $\mathcal{O}(n^2(\Delta^2+1))$ space.*

Next, for any four integers $n-1 \geq m \geq 0$, and $\Delta \geq d \geq 0$, we present Algorithm 1 for solving the problem of calculating $h(n,\Delta,m_{\leq},d_{\leq})$. In this algorithm, for each integers $1 \leq i \leq n, 0 \leq j \leq \Delta, 0 \leq k \leq \min\{i,m\}$, and $0 \leq p \leq \min\{j,d\}$, the variables $h\,[i,j,k_{\leq},p_{\leq}]$, $h\,[i,j,k_{=},p_{\leq}]$, and $h\,[i,j,k_{=},p_{=}]$ store the values of $h(i,j,k_{\leq},p_{\leq})$, $h(i,j,k_{=},p_{\leq})$, and $h(i,j,k_{=},p_{=})$, respectively.

Lemma 6. *For any four integers $n-1 \geq m \geq 0$, and $\Delta \geq d \geq 0$, Algorithm 1 outputs $h(n,\Delta,m_{\leq},d_{\leq})$ in $\mathcal{O}(nm(n+\Delta(n+d\cdot\min\{n,\Delta\})))$ time and $\mathcal{O}(nm(\Delta(d+1)+1))$ space.*

Proof. Correctness: For each integer $1 \leq i \leq n, 0 \leq j \leq \Delta, 0 \leq k \leq \min\{i,m\}$, and $0 \leq p \leq \min\{j,d\}$, all the substitutions and if-conditions in Algorithm 1 follow from Lemmas 1, 2, 3 and 4. Furthermore, the values $h[i,j,k_{\leq},p_{\leq}]$, $h[i,j,k_{=},p_{\leq}]$, and $h[i,j,k_{=},p_{=}]$ are computed by the recursive relations given in Lemmas 1 and 4. This implies that Algorithm 1 correctly computes the required value $h[n,\Delta,m_{\leq},d_{\leq}]$.

Complexity analysis: There are three nested loops over the variables i, j, and p at line 4, which take $\mathcal{O}(n(\Delta(d+1)+1))$ time. Following there are five nested loops: over variables i,j,k,p, and q at lines 5, 6, 7, 8, and 31, respectively. The loop at line 5 is of size $\mathcal{O}(n)$, while the loop at line 6 is of size $\mathcal{O}(\Delta)$. Similarly, the loops at lines 7 and 8 are of size $\mathcal{O}(m)$ and $\mathcal{O}(d)$, respectively. The fifth nested loop at line 18 is of size $\mathcal{O}(n)$ (resp., $\mathcal{O}(\min\{n,\Delta\})$ if $p = 0$ (resp., otherwise). Thus from line 5–36, Algorithm 1 takes $\mathcal{O}(n^2m)$ (resp., $\mathcal{O}(nm\Delta(n+d\cdot\min\{n,\Delta\}))$) time if $\Delta = 0$ (resp., otherwise). Therefore, Algorithm 1 takes $\mathcal{O}(nm(n+\Delta(n+d\cdot\min\{n,\Delta\})))$ time.

The algorithm stores three four-dimensional arrays. When $\Delta = 0$, for each integer $1 \leq i \leq n$, and $1 \leq k \leq \min\{i,m\}$ we store $h[i,0,k_{\leq},0_{\leq}]$, $h[i,0,k_{=},0_{\leq}]$ and $h[i,0,k_{=},0_{=}]$, taking $\mathcal{O}(nm)$ space. When $\Delta \geq 1$, then for each integer $1 \leq i \leq n, 0 \leq j \leq \Delta, 1 \leq k \leq \min\{i,m\}$ and $0 \leq p \leq \min\{j,d\}$ we store $h[i,j,k_{\leq},p_{\leq}]$, $h[i,j,k_{=},p_{\leq}]$ and $h[i,j,k_{=},p_{=}]$, taking $\mathcal{O}(nm\Delta(d+1))$ space. Hence, Algorithm 1 takes $\mathcal{O}(nm(\Delta(d+1)+1))$ space. □

Algorithm 1 DP based counting algorithm for $h(n, \Delta, m_{\leq}, d_{\leq})$

Input: Integers $n-1 \geq m \geq 0$ and $\Delta \geq d \geq 0$.
Output: $h(n, \Delta, m_{\leq}, d_{\leq})$.

$h[1, j, 0_{=}, 0_{=}] := h[1, j, 0_{=}, p_{\leq}] := h[1, j, 0_{\leq}, p_{\leq}] := 1$;
$h[i, j, 0_{=}, p_{\leq}] := h[i, j, 0_{\leq}, p_{\leq}] := 0$;
$h[2, j, 1_{=}, p_{=}] := 1$; $h[2, j, 1_{=}, p_{\leq}] := h[2, j, 1_{\leq}, p_{\leq}] := p + 1$
for each $2 \leq i \leq n, 0 \leq j \leq \Delta, 0 \leq p \leq \min\{j, d\}$;

for $i := 3, 4, \ldots, n$ **do**
  for $j := 0, 1, \ldots, \Delta$ **do**
    for $k := 1, 2, \ldots, \min\{i, m\}$ **do**
      for $p := 0, 1, \ldots, \min\{j, d\}$ **do**
        if $p = 0$ and $k = 1$ **then**
          $h[i, j, 1_{=}, 0_{=}] := h[i, j, 1_{=}, 0_{\leq}] := h[i, j, 1_{\leq}, 0_{\leq}] := 1$
        else /* $p \geq 1$ or $k \geq 2$ */
          $c := 1$; $h[i, j, k_{=}, p_{=}] := 0$; /* Initialization */
          if $p = 0$ **then**
            $\ell := \lfloor (i-1)/k \rfloor$
          else /* $p \geq 1$ */
            $\ell := \min\{\lfloor (i-1)/k \rfloor, \lfloor j/p \rfloor\}$
          end if;
          for $q := 1, 2, \ldots, \ell$ **do**
            $c := c \cdot (h[k, p, k-1_{\leq}, p_{\leq}] + q - 1)/q$;
            if $p = 0$ **then**
              $h[i, j, k_{=}, p_{=}] := h[i, j, k_{=}, p_{=}] + c \cdot h[i - qk, j, \min\{i - kq - 1, k-1\}_{\leq}, j_{\leq}]$
            else /* $p \geq 1$ */
              $h[i, j, k_{=}, p_{=}] := h[i, j, k_{=}, p_{=}] + c \cdot h[i - kq, j - pq, k_{=}, \min\{j - pq, p - 1\}_{\leq}] + h[i - kq, j - pq, \min\{i - kq - 1, k-1\}_{\leq}, j - pq_{\leq}]$
            end if
          end for;
          if $p = 0$ **then** /* $k \geq 2$ */
            $h[i, j, k_{=}, 0_{\leq}] := h[i, j, k_{=}, 0_{=}]$
          else /* $p \geq 1$ */
            $h[i, j, k_{=}, p_{\leq}] := h[i, j, k_{=}, p-1_{\leq}] + h[i, j, k_{=}, p_{=}]$
          end if;
          $h[i, j, k_{\leq}, p_{\leq}] := h[i, j, k-1_{\leq}, p_{\leq}] + h[i, j, k_{=}, p_{\leq}]$
        end if
      end for
    end for
  end for
end for;
output $h[n, \Delta, m_{\leq}, d_{\leq}]$ as $h(n, \Delta, m_{\leq}, d_{\leq})$.

Theorem 1. *For any two integers $n \geq 1$ and $\Delta \geq 0$, the number of non-isomorphic trees with n vertices and Δ self-loops can be obtained in $\mathcal{O}(n^2(n + \Delta(n + \Delta \cdot \min\{n, \Delta\})))$ time and $\mathcal{O}(n^2(\Delta^2 + 1))$ space.*

Proof. By Jordan [18], we can uniquely consider any tree as a rooted tree by either regarding its unicentroid as the root, or in the case of a bicentroid, by introducing a virtual vertex on the bicentroid and assuming the virtual vertex as the root of the tree. By the definition of a unicentroid, the number of mutually non-isomorphic trees with n vertices, Δ self-loops and a unicentroid is $h(n, \Delta, \lfloor (n -$

$1)/2\rfloor_{\leq}, \Delta_{\leq})$. Further, if n is even, then there exist trees with n vertices and a bicentroid. This implies that the number of mutually non-isomorphic trees with n vertices and Δ self-loops is $h(n, \Delta, \lfloor (n-1)/2\rfloor_{\leq}, \Delta_{\leq})$ when n is odd. Let n be an even integer. Then any tree H with n vertices, Δ self-loops and a bicentroid has two connected components, A and B obtained by the removal of the bicentroid such that $A \in \mathcal{H}(n/2, i, n/2-1_{\leq}, i_{\leq})$ and $B \in \mathcal{H}(n/2, \Delta - i, n/2-1_{\leq}, \Delta - i_{\leq})$ for some $0 \leq i \leq \lfloor \Delta/2 \rfloor$, where if Δ is even then for $i = \Delta/2$, both of the components A and B belong to $\mathcal{H}(n/2, \Delta/2, n/2-1_{\leq}, \Delta/2_{\leq})$.

Note that for any $0 \leq i \leq \lfloor (\Delta-1)/2 \rfloor$, it holds that

$$\mathcal{H}(n/2, i, n/2-1_{\leq}, i_{\leq}) \cap \mathcal{H}(n/2, \Delta - i, n/2-1_{\leq}, \Delta - i_{\leq}) = \emptyset.$$

Therefore, when Δ is odd (resp., even), the number of mutually non-isomorphic trees with n vertices, Δ self-loops, and a bicentroid is

$$\sum_{i=0}^{\lfloor (\Delta-1)/2 \rfloor} h(n/2, i, n/2-1_{\leq}, i_{\leq})\, h(n/2, \Delta - i, n/2-1_{\leq}, \Delta - i_{\leq}) + \alpha \binom{h(n/2, \Delta/2, n/2-1_{\leq}, \Delta/2_{\leq}) + 1}{2},$$

such that $\alpha = 0$ (resp., $\alpha = 1$). Thus, the number of mutually non-isomorphic trees with n vertices and Δ self-loops is

$$h(n, \Delta, \lfloor (n-1)/2\rfloor_{\leq}, \Delta_{\leq}) + \sum_{i=0}^{\lfloor (\Delta-1)/2 \rfloor} h(n/2, i, n/2-1_{\leq}, i_{\leq})\, h(n/2, \Delta - i, n/2-1_{\leq}, \Delta - i_{\leq}) + \alpha \binom{h(n/2, \Delta/2, n/2-1_{\leq}, \Delta/2_{\leq}) + 1}{2} \tag{6}$$

such that $\alpha = 0$ (resp., $\alpha = 1$) when Δ is odd (resp., even). Moreover, for each $0 \leq i \leq \Delta$, Algorithm 1 also computes and stores $h(n/2, i, n/2-1_{\leq}, i_{\leq})$ during the calculation of $h(n, \Delta, \lfloor (n-1)/2\rfloor_{\leq}, \Delta_{\leq})$, and therefore the required result follows from Lemma 6. □

We implemented the proposed DP algorithm and counting trees with a given number of vertices and self-loops. The experimental results in Table 1 show that the proposed method efficiently counts trees with n vertices and Δ self-loops.

Table 1. Experimental result of the counting method.

(n, Δ)	Number of Trees	Time [s]
$(10, 0)$	106	0.000173
$(20, 0)$	823,065	0.00048
$(10, 5)$	91,037	0.001193
$(10, 30)$	6,629,790,712	0.00881
$(20, 10)$	5,143,681,226,004	0.006869
$(30, 10)$	2,547,562,522,909,694,331	0.015901

We next give a lower bound and an upper bound on the number of tree-like polymer topologies with self-loops of a given rank. For this we prove the following results.

Lemma 7. *For an integer $n \geq 2$, there exists at least one tree-like polymer with n vertices and Δ self-loops if $\Delta \geq \lceil \frac{n}{2} \rceil + 1$.*

Proof. Consider a tree T of n vertices of diameter $\lfloor \frac{n}{2} \rfloor$ such that T contains a path of length $\lfloor \frac{n}{2} \rfloor$, in which each non-end vertex has degree at least 3. Observe that when n is even, the tree T has exactly $\lceil \frac{n}{2} \rceil - 1$ vertices of degree 3, and hence $n - \lceil \frac{n}{2} \rceil + 1 = \lceil \frac{n}{2} \rceil + 1$ vertices of degree less than 3.

When n is odd, the tree T has $\lceil \frac{n}{2} \rceil - 3$ vertices of degree 3 and one vertex of degree 4. Thus, in this case, the number of vertices of degree less than 3 is $n - \lceil \frac{n}{2} \rceil + 2 = \left(2 \lceil \frac{n}{2} \rceil - 1\right) - \lceil \frac{n}{2} \rceil + 2 = \lceil \frac{n}{2} \rceil + 1$. This implies that T can be transformed into a polymer with $\lceil \frac{n}{2} \rceil + 1$ self-loops by assigning a self-loop to each vertex of degree less than 3. Hence, $\lceil \frac{n}{2} \rceil + 1$ self-loops are sufficient to get a tree-like polymer with n vertices. □

For two integers $n \geq 1$ and $\Delta \geq 0$, let $t(n, \Delta)$ denote the number of trees with n vertices and Δ self-loops. For $r \geq 1$, let $p(r)$ denote the number of tree-like polymers with self-loops and no multi-edges of rank r. Observe that a tree with n vertices and k self-loops at each vertex is a polymer with n vertices of cycle rank kn. From this fact and Lemma 7 it holds that

$$\sum_{n,k \in \mathbb{Z}^+ : nk = r} t(n, 0) \leq p(r) \leq \sum_{n \in \mathbb{Z}^+ : \lceil \frac{n}{2} \rceil + 1 \leq r} t(n, r).$$

4. Conclusions

This paper presented an efficient method to count the number of all mutually non-isomorphic trees with a given number of vertices and self-loops. The proposed method is based on dynamic programming where we count the number of all mutually non-isomorphic rooted trees with a given number n of vertices and Δ self-loops in $\mathcal{O}(n^2(n + \Delta(n + \Delta \cdot \min\{n, \Delta\})))$ time and $\mathcal{O}(n^2(\Delta^2 + 1))$ space. As an application of our results, we gave lower and upper bounds on the number of tree-like polymer topologies with a given cycle rank. This is an interesting application of DP to objects such as trees, and offers the advantage of getting the size of the entire solution space at low computational complexity without explicitly generating each object.

An interesting direction for future research is to efficiently generate all mutually non-isomorphic trees with a given number of vertices and self-loops by using the result from the developed counting method. Further, another possible extension of this research is to count and generate all mutually non-isomorphic tree-like polymer topologies with a given number of vertices and self-loops.

Author Contributions: Conceptualization, N.A.A. and H.N.; funding acquisition, N.A.A.; methodology, N.A.A. and H.N.; software, N.A.A.; supervision, H.N.; validation, N.A.A., A.S. and H.N.; writing—original draft, N.A.A.; writing—review and editing, N.A.A. and A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research is partially funded by JSPS KAKENHI Grant Number 18J23484.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pólya, G. Kombinatorische anzahlbestimmungen für gruppen, graphen und chemische verbindungen. *Acta Math.* **1937**, *68*, 145–254. [CrossRef]
2. Polya, G.; Read, R.C. *Combinatorial Enumeration of Groups, Graphs, and Chemical Compounds*; Springer Science & Business Media: New York, NY, USA, 2012.
3. Blum, L.C.; Reymond, J.L. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *J. Am. Chem. Soc.* **2009**, *131*, 8732–8733. [CrossRef] [PubMed]
4. Azam, N.A.; Chiewvanichakorn, R.; Zhang, F.; Shurbevski, A.; Nagamochi, H.; Akutsu, T. A method for the inverse QSAR/QSPR based on artificial neural networks and mixed integer linear programming. In Proceedings of the 13th International Joint Conference on Biomedical Engineering Systems and Technologies—Volume 3: Bioinformatics, Valletta, Malta, 24–26 February 2020.
5. Ito, R.; Azam, N.A.; Wang, C.; Shurbevski, A.; Nagamochi, H.; Akutsu, T. A novel method for the inverse QSAR/QSPR to monocyclic chemical compounds based on artificial neural networks and integer programming. In *Advances in Computer Vision and Computational Biology*; Springer Nature-Research Book Series; Springer: Berlin/Heidelberg, Germany, 2020.

6. Zhu, J.; Wang, C.; Shurbevski, A.; Nagamochi, H.; Akutsu, T. A novel method for inference of chemical compounds of cycle index two with desired properties based on artificial neural networks and integer programming. *Algorithms* **2020**, *13*, 124. [CrossRef]
7. Méndez-Lucio, O.; Baillif, B.; Clevert, D.A.; Rouquié, D.; Wichard, J. De novo generation of hit-like molecules from gene expression signatures using artificial intelligence. *Nat. Commun.* **2020**, *11*, 10. [CrossRef] [PubMed]
8. Lim, J.; Hwang, S.Y.; Moon, S.; Kim, S.; Kim, W.Y. Scaffold-based molecular design with a graph generative model. *Chem. Sci.* **2020**, *11*, 1153–1164. [CrossRef]
9. Meringer, M.; Schymanski, E.L. Small molecule identification with MOLGEN and mass spectrometry. *Metabolites* **2013**, *3*, 440–462. [CrossRef] [PubMed]
10. Benecke, C.; Grund, R.; Hohberger, R.; Kerber, A.; Laue, R.; Wieland, T. MOLGEN+, a generator of connectivity isomers and stereoisomers for molecular structure elucidation. *Anal. Chim. Acta* **1995**, *314*, 141–147. [CrossRef]
11. Available online: http://sunflower.kuicr.kyoto-u.ac.jp/tools/enumol2/ (accessed on 4 July 2020).
12. Peironcely, J.E.; Rojas-Chertó, M.; Fichera, D.; Reijmers, T.; Coulier, L.; Faulon, J.L.; Hankemeier, T. OMG: Open molecule generator. *J. Cheminf.* **2012**, *4*, 21. [CrossRef] [PubMed]
13. Vogt, M.; Bajorath, J. Chemoinformatics: A view of the field and current trends in method development. *Bioorg. Med. Chem.* **2012**, *20*, 5317–5323.
14. Haruna, T.; Horiyama, T.; Shimokawa, K. On the enumeration of polymer topologies. *IPSJ SIG Tech. Rep.* **2017**, *2017-Al-162*, 1–5.
15. Tezuka, Y.; Oike, H. Topological polymer chemistry. *Prog. Polym. Sci.* **2002**, *27*, 1069–1122. [CrossRef]
16. Galina, H.; Sysło, M.M. Some applications of graph theory to the study of polymer configuration. *Discret. Appl. Math.* **1988**, *19*, 167–176. [CrossRef]
17. Zimm, B.H.; Stockmayer, W.H. The dimensions of chain molecules containing branches and rings. *J. Chem. Phys.* **1949**, *17*, 1301–1314. [CrossRef]
18. Jordan, C. Sur les assemblages de lignes. *J. Reine Angew. Math.* **1869**, *70*, 81.