The Number of Connected Components in Graphs and Its Applications

Ryuhei Uehara
uehara@komazawa-u.ac.jp

Natural Science Faculty, Komazawa University

Abstract

For any given graph and an integer k, the number of connected components with k vertices in the graph is investigated. For the vertex set of size n and the maximum degree Δ, the number is bounded above by $\frac{(e\Delta)^k}{(\Delta-1)k}$. The factor Δ^k is essential, since we give the lower bound $n\left(\frac{\Delta}{2}\right)^{k-1}$ for $k < \frac{2n}{\Delta}$. We also show that all connected components with k vertices are enumerable in $O\left(n+m+\frac{(e\Delta)^k}{\Delta-1}k^2\log k\right)$ time with $O(n+m)$ space, where m is the number of edges. Thus, if Δ^k is bounded above by some polynomial of n (e.g. Δ is fixed and $k = O(\log n)$), we can check all connected components with k vertices in polynomial time of n. Under the assumption, several intractable problems (including Topological Containment (Subgraph Isomorphism), Small Minimum Degree 4 Subgraph, Bipartite Graph Embedding, Steiner Tree, and k-Minimum Spanning Tree) are put into tractable.

Key words: Graph theory, parameterized complexity, enumeration algorithm.

1 Introduction

We focus on the following basic problem: How many connected components with k vertices in a given graph? (We call a vertex set *connected component* if its vertex induced graph is connected.) We can easily see the worst case; when a given graph is a complete graph with n vertices, the number of connected components with k vertices is $\binom{n}{k} > \left(\frac{n}{k}\right)^k$, that is not a polynomial of n even for $k = \omega(1)$. Thus, in general, we can not check/enumerate all connected components on a graph in polynomial time, even if the size of the connected components is very small. Hence we turn to consider the problem with two parameters; not only the size of the connected components, but also the maximum degree of the graph. The main results in this paper are the following:

> For any connected graph with n vertices, m edges, and the maximum degree Δ, the number of connected components with k vertices is bounded above by $\frac{(e\Delta)^k}{(\Delta-1)k}$. Moreover, we can enumerate all of them in $O\left(n+m+\frac{(e\Delta)^k}{\Delta-1}k^2\log k\right)$ time with $O(n+m)$ space.

That is, if we can assume that Δ^k is bounded above by a polynomial of n (e.g., Δ is a constant and $k = O(\log n)$), we can check/enumerate all connected components with k vertices in polynomial time. We also show the lower bound stating that the factor Δ^k is essential:

> The lower bound of the number of connected components with k vertices in a graph is $n\left(\frac{\Delta}{2}\right)^{k-1}$ for $k < \frac{2n}{\Delta}$.

Remark that under the assumption that Δ^k is bounded above by a polynomial of n, we can check all leaf-root paths of a search tree of depth k on the graph in a polynomial time. Thus the problems, which can be solved by checking all leaf-root paths, are solvable in polynomial time under the assumption. (For example, Exact Cheap Tour [7, p.461], Short Cheap Tour [7, p.484], k-Clique, and the natural analogue of the Hamilton Circuit [6, p.127] have the property).

The main theorem gets over the limit of the simple search tree. As applications of the main theorem, we put the following problems into tractable under the assumption: Topological Containment (Subgraph Isomorphism), Small Minimum Degree 4 Subgraph, Bipartite Graph Embedding, Steiner Tree, and k-Minimum Spanning Tree.

The Topological Containment problem is, for given graphs $G = (V, E)$ and $H = (V', E')$ with $|V| = n$ and $|V'| = k$, to determine whether G contains a subgraph isomorphic to H [7]. The Topological Containment problem is also known as the Subgraph Isomorphism problem [1]. When $k = n$, the problem is well known Graph Isomorphism problem (see [10] for the detail). While the complexity of the Graph Isomorphism problem is still open, the Topological Containment problem is NP-complete [10, p.24]. On the other hand, there are few results for the parameterized complexity of the Topological Containment problem; it is $W[1]$-hard since it contains some $W[1]$-complete problems as special cases. Recently, using the Color-Coding method, Alon, Yusterm, and Zwick showed that the Topological Containment problem is polynomial time solvable if the input graph H is tree-width bounded, and $k = O(\log n)$ [1]. Our assumption is incomparable to theirs.

The Small Minimum Degree 4 Subgraph problem and the Bipartite Graph Embedding problem are $W[1]$-complete, that is believed that they are parameterized intractable [5, 6]. The Steiner Tree problem is $W[2]$-hard [7], that seems more difficult than $W[1]$.

The k-Minimum Spanning Tree (k-MST) problem is to find a minimum cost tree spanning exactly k vertices. It is well known that n-MST, or ordinary MST, is easy to find even in NC (see e.g. [11]). However, interestingly, the k-MST problem is NP-complete for general k. So far as the approximation algorithms, the sum of the weights of i-MST for $1 \leq i \leq n$ bounds the solution of the Minimum Latency Tour, that is also known in the literature as the deliveryman problem or the traveling repairman problem (see [12, 4]).

The latter half of the main theorem states that in the applications above, all solutions or all candidates for solutions are enumerable in a polynomial time under the assumption. The enumeration problems have been widely investigated; for example, the enumeration of all connected components is discussed by Avis and Fukuda [2], and an optimal enumeration algorithm of all (ordinary) spanning trees is given by Shioura, Tamura and Uno [13]. However, the polynomial time computability of the enumeration problems has not been discussed in the literature.

2 Preliminaries

We will deal only with graphs without loops or multiple edges. Throughout the paper, unless stated otherwise, $G = (V, E)$ always denotes the input (undirected) graph with $V = \{0, 1, \cdots, n-1\}$ and $|E| = m$. Without loss of generality, we assume that G is connected. The *neighborhood* of a vertex v in G, denoted by $N_G(v)$, is the set of vertices in G adjacent to v. The *degree* of a vertex v in G is $|N_G(v)|$, and denoted by $d_G(v)$. The *maximum degree* of a graph G is the upper bound of the degree of each vertex in G. For $U \subseteq V$, $N_G(U)$ is $\cup_{u \in U} N_G(u)$, and $G[U]$ is the graph (U, F), where $F = \{\{u, v\} \mid u, v \in U \text{ and } \{u, v\} \in E\}$. For a graph G, $|G|$ denotes the number of vertices in G. For $U \subseteq V$, we call U *k-connected component* of G if $G[U]$ is connected and $|U| = k$.

An acyclic connected graph, one not containing any cycles, is called a *tree*. If T is a tree and $r \in T$ is any fixed vertex, we call the pair (T, r) a *rooted tree*, and r its *root*. Throughout the paper, unless stated otherwise, we will deal only with rooted tree. Hence we often simply write T for (T, r). For each vertex v in T, let $P(v)$ be the unique path from r to v. The *parent* of v is the neighbor of v on $P(v)$. The *children* of v are its other neighbors. The *depth* of v is the length of $P(v)$. (The depth of the root is zero.) In a tree, the *leaves* are the vertices of degree 1, and the *internal vertices* are the vertices which are not leaves. For each leaf v, we sometimes call $P(v)$ the *leaf-root path*. A *binary* tree is a rooted tree in which each vertex has at most two children. Two children of an internal vertex v is designated as its *left child* and *right child* of v. The subtrees rooted at the children of v are the *left subtree* and the *right subtree* of v. We remark that, even if v has only one child, we distinguish with right child and left child of v. (Although such trees are often called "labeled tree", we simply say "tree" in the paper.) A *d-ary* tree is a rooted tree in which each vertex has at most d children, which are also distinguished with each other.

For a given graph G, an edge set F is called *k-spanning tree* of G if the graph (U, F), where $U = \{v | \{v, u\} \in F\}$, is a tree with $|U| = k$. An n-spanning tree is simply called a *spanning tree*.

3 The Number of Small Connected Components

In this section, we show the upper bound of the number of the k-connected components in a graph. To show the upper bound, the number of d-ary trees will be considered. We also show the lower bounds of the number of the k-connected components in some cases.

3.1 The upper bounds

We denote by $t_d(n)$ the number of different d-ary trees with n vertices. (For example, $t_2(1) = 1$, $t_2(2) = 2$, and $t_2(3) = 5$.) The following theorem will be used later (we mention that $t(0)$ is defined by 1 according to the theorem).

Theorem 1 The number $t_d(n)$ is known as the nth Fuss-Catalan number equal to $\frac{\binom{dn}{n}}{(d-1)n+1}$.

Proof. Any tree has its root, say r. Then looking at each subtree, we have the following:

$$t_d(n) = \sum_{\substack{i_0+i_1+\cdots+i_{d-1}=n-1, \\ 0 \le i_0, i_1, \cdots, i_{d-1}}} t_d(i_0) t_d(i_1) \cdots t_d(i_{d-1}),$$

of which the nth Fuss-Catalan number is the solution as noticed in [9, 7.5]. ∎

We now have the following:

Theorem 2 Let $G = (V, E)$ be a given graph and Δ be the maximum degree of G. Then the number of k-connected components in G is bounded by $\frac{n}{k} t_\Delta(k) \le \frac{(e\Delta)^k}{(\Delta-1)k}$.

Proof. We first fix a vertex v in G and consider any k-connected component including v. Any connected component has at least one spanning tree. We thus fix any spanning tree. Distinct connected components have different labeled trees with each other. Thus, the number of k-connected components including v is bounded above by $t_\Delta(k)$. When we add them up for every vertex in G, each component is counted exactly k times. Thus, we have the upper bound $\frac{n}{k} t_\Delta(k)$. The fact $\binom{n}{k} \ge \left(\frac{n}{k}\right)^k$ completes the proof. ∎

Corollary 3 For a given graph with the maximum degree Δ, the number of k-spanning tree in G is bounded by $\frac{n}{k} t_\Delta(k) \le \frac{(e\Delta)^k}{(\Delta-1)k}$.

Proof. In the proof of Theorem 2, we showed that for each k-connected component, any spanning tree is different from the tree spanning another connected component. Let C be any k-connected component that contains two or more spanning trees. Then the distinct spanning trees of C are different labeled trees with each other. Thus, the number of k-spanning trees including any fixed vertex can also be bounded by $t_\Delta(k)$. ∎

3.2 The Lower Bounds of Connected Components

In this subsection, we show the lower bound of the number of the k-connected components. We first remark that $t_d(k)$ gives the lower bound of k-connected components in a d-ary tree for very small k. More precisely, for $k \le \lfloor \log_d n \rfloor$, $t_d(k) = \binom{dk}{k}/((d-1)k+1)$ is the lower bound of the number of k-connected components in a complete d-ary tree of size n, that is a planar graph with the maximum degree $d+1$. To show the lower bound for large k, we define a graph $\mathcal{C}(n, d)$ as follows; all vertices are grouped into n distinct subsets $V_0, V_1, \cdots, V_{n-1}$ with $|V_i| = d$ $(0 \le i \le n-1)$; and each vertex in V_i is connected to all vertices in $V_{i+1 \pmod n}$ (and therefore all vertices in $V_{i-1 \pmod n}$) for each $0 \le i \le n-1$. Figure 1 depicts $\mathcal{C}(5, 3)$. The number of vertices in $\mathcal{C}(n, d)$ is nd and the maximum degree

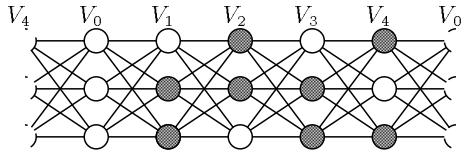

Figure 1: $\mathcal{C}(5,3)$

of $\mathcal{C}(n,d)$ is $2d$. For each i and j with $0 \leq i,j \leq n$, We let $V_{i:j}$ denote $V_i \cup V_{i+1 \pmod n} \cup \cdots \cup V_j$. Let V' be any vertex set of V that contains vertices in consecutive subsets. More precisely, there exists integers k and h such that $V' \subseteq V_{k:h}$, and $V' \cap V_k \neq \emptyset$, $V' \cap V_{k+1 \pmod n} \neq \emptyset$, $\cdots$, $V' \cap V_h \neq \emptyset$. Then, it is clear that V' is a $|V'|$-connected component of $\mathcal{C}(n,d)$.

Lemma 4 There exists a graph with n vertices and the maximum degree Δ such that the number of k-connected components is greater than $\frac{2n}{\Delta}\left(\frac{\Delta}{2c}\right)^k$, where c is any integer satisfying $\frac{\Delta k}{2n} \leq c \leq \frac{\Delta}{2}$.

Proof. Without loss of generality, we assume that Δ is even and Δ divides $2n$. We show that $\mathcal{C}\left(\frac{2n}{\Delta}, \frac{\Delta}{2}\right)$ contains at least $\frac{2n}{\Delta}\left(\frac{\Delta}{2c}\right)^k$ k-connected components. We first remark that the maximum degree of $\mathcal{C}\left(\frac{2n}{\Delta}, \frac{\Delta}{2}\right)$ is Δ. For each integers i and j such that $j - i = \frac{k}{c} \pmod n$, we consider the following subset V' of the vertices in the graph; $V' \subseteq V_{i:j}$, and for each subset V_h in $V_{i:j}$, $|V' \cap V_h| = c$. The gray vertices in Figure 1 depicts an example of the subset V' for $i = 1$, $j = 4$, $c = 2$, and $k = 8$. It is easy to see that V' contains k vertices. Moreover, from its definition, $\mathcal{C}\left(\frac{2n}{\Delta}, \frac{\Delta}{2}\right)$ is grouped into $\frac{2n}{\Delta}$ (greater than $\frac{k}{c}$) distinct subsets of size $\frac{\Delta}{2}$ (greater than c). Thus the possible number of the subset V' is equal to

$$\frac{2n}{\Delta}\binom{\Delta/2}{c}^{\frac{k}{c}} \geq \frac{2n}{\Delta}\left(\frac{\Delta}{2c}\right)^k.$$

∎

Letting $c = 1$ in Lemma 4, we have the following theorem:

Theorem 5 When $k < \frac{2n}{\Delta}$, the lower bound of the number of k-connected components is $n\left(\frac{\Delta}{2}\right)^{k-1}$.

We also show that the factor Δ^k is essential even if the input graph is tree that is acyclic and planar:

Theorem 6 When $k < \frac{n}{\Delta} + 1$, there exists a tree with n vertices and the maximum degree Δ such that the number of k-connected components in the tree is greater than $\binom{\Delta+k-2}{k-1} > \left(\frac{\Delta+k-2}{k-1}\right)^{k-1}$.

Proof. We consider the following tree with n vertices: The root has Δ children; every internal vertex except the root has one child; and every leaf has depth $\lceil n/\Delta \rceil$ or $\lfloor n/\Delta \rfloor$ (greater than k). We consider each subtree rooted at the root with k vertices as a k-connected component. Then the number of the subtrees is equal to how many placement of $(k-1)$ balls (vertices) in Δ bins (subtrees), that is known as "occupancy problem" [8, II.5]. Thus the number of the subtrees is equal to $\binom{\Delta+k-2}{k-1}$. ∎

4 Applications

In this section, we state the problems which are put into tractable under the assumption that Δ^k is bounded by a polynomial of n.

4.1 Topological Containment (Subgraph Isomorphism)

Topological Containment is the problem defined below:

Input: Two graphs $G = (V,E)$ and $H = (V',E')$ with $|V'| = k$

Question: Is H topologically contained in G?

When $k = n$, the problem is well known the Graph Isomorphism problem whose complexity is still open (see [10]). For general k, the problem is NP-complete (see [10, p.24]). The parameterized complexity of the problem is remained open [7, p.481]. However, since some $W[1]$-complete problems (e.g., the k-Clique problem and the Bipartite Graph Embedding problem) are contained in the problem as the special case, the Topological Containment problem is $W[1]$-hard.

Our algorithm is the following: compute the maximum degree Δ of G, and check whether H is isomorphic to each k-connected component. Zemlyachenko, Korneenko, and Tyshkevich showed the algorithm solving the Graph Isomorphism problem in $O(e^{n^{1-c}})$ time, where c is a positive constant less than 1 [15]. (The result is reviewed by Babai in [3].) Using their algorithm, we can solve the problem in $O\left(\frac{nt_{\Delta}(k)}{k}e^{k^{1-c}}\right)$ time. When $\Delta \geq 3$, $e^{k^{1-c}}$ is less than Δ^k. When $\Delta < 3$, each graph is either a path or a cycle. Then the isomorphism of the graphs is easy to check in polynomial time. Thus, in each case, we can check the isomorphism in polynomial time under the assumption. Thus we can conclude the Topological Containment problem is tractable under the assumption.

4.2 Other $W[1]$-hard Problems

We next show the tractability of the other $W[1]$-hard problems under the assumptions. We remark that $W[1]$-hard problems are believed to be parameterized intractable in general case [5, 7]. The following problems seem to be difficult to solve by checking each leaf-root path of search trees.

Small Minimum Degree 4 Subgraph

Input: A graph $G = (V, E)$, an integer k.

Output: Is there a subgraph of G of minimum degree at least 4 and of cardinality at most k?

Comments: This problem is $W[1]$-complete [7, p.457]. For each $i = 1, 2, \cdots, k$, we can check the property every i-connected component in polynomial time under the assumption.

Bipartite Graph Embedding

Input: Two bipartite graphs $G = (V, E)$ and $H = (V', E')$ with $|V'| = k$.

Question: Is H embeddable G?

Comments: This problem is $W[1]$-complete [6, p.129]. This problem is a special case of the Topological Containment problem, and it is solvable in polynomial time under the assumption as stated in 4.1.

Steiner Tree

Input: A graph $G = (V, E)$, a subset S of V with k_1 vertices, an integer k_2.

Question: Is there a set of vertices $T \subseteq V - S$ such that $|T| \leq k_2$ and $G[S \cup T]$ is connected?

Comments: The Steiner Tree problem is $W[2]$-hard [7, p.453]. However, we can solve it checking whether each k_1+i-connected component contains S for $0 \leq i \leq k_2$. That can be done in polynomial time under the assumption that $\Delta^{k_1+k_2}$ is bounded by a polynomial of n.

4.3 k-Minimum Spanning Tree

The *k-Minimum Spanning Tree* (k-MST) problem is defined below:

Input: A weighted graph $G = (V, E)$, a weighted function $w : E \rightarrow \mathcal{Z}$, a positive integer k.

Output: A minimum cost k-spanning tree, where the *cost* of the tree is the sum of the weight of the edges in the tree.

It is well known that ordinary MST is easy to solve even in NC (see e.g. [11]), however, interestingly, the k-MST problem is NP-complete for general k [12, 4].

For each k-connected component, using the standard greedy algorithm, we can find (ordinary) minimum spanning tree in the k-connected component in polynomial time of k. Thus, under the assumption, we can find the minimum cost k-spanning tree in polynomial time by checking all ordinary minimum spanning trees in the k-connected components.

4.4 Enumeration Algorithm

Avis and Fukuda introduced the notion of *reverse search*, that is useful for enumeration of every induced subgraph in a graph, and showed the following theorem:

Theorem 7 [2, Theorem 3.9] For a given graph $G = (V, E)$ and k, let K_1 be the sum of the number of i-connected components for $1 \leq i \leq k$, K_2 be the number of k-connected components, and k' be the time required to output each k-connected component. Then, we can enumerate all k-connected components in $O(n + m + K_1 + K_2 k')$ time using $O(n + m)$ space.

Here, $K_2 = t_\Delta(k) \leq \frac{(e\Delta)^k}{(\Delta-1)k}$, $K_1 = \sum_{i=1}^{k} \frac{n}{i} t_\Delta(i) \leq \sum_{i=1}^{k} \frac{(e\Delta)^n}{(\Delta-1)i} \leq \frac{(e\Delta)^n}{\Delta-1} \log k$, and $k' = O(k^2)$. Thus, $n + m + K_1 + K_2 k' = O\left(n + m + \frac{(e\Delta)^k}{\Delta-1} k^2 \log k\right)$, and we immediately have the following:

Theorem 8 For a given graph, all k-connected components can be enumerated in $O\left(n + m + \frac{(e\Delta)^k}{\Delta-1} k^2 \log k\right)$ time with $O(n + m)$ space.

We remark that the algorithm can "check" each connected component instead of "output". Moreover, it is easy to modify the algorithm to check whether "all", "at least one", or "at least half" connected components satisfy some graph property.

5 Concluding Remarks

Theorem 8 states that for the applications in Section 4, every solution or every candidate for a solution can be enumerated in a polynomial time under the assumption. However, that is stated in general cases, and sometimes we can enumerate them more efficiently. For example, all spanning trees can be enumerated by the optimal algorithm by Shioura, Tamura, and Uno that requires $O(n + m + N)$ time and $O(n + m)$ space, where m is the number of edges, and N is the number of spanning trees [13]. Their algorithm first produces a (depth-first) spanning tree, and then outputs each pair of edges (e_1, e_2) that means "the difference of two spanning trees". That is, when we have a tree T and receive the output (e_1, e_2), we get the next tree $T - \{e_1\} + \{e_2\}$. Using the idea, we can enumerate all k-spanning trees and all k-connected components of a given graph more efficiently [14].

Acknowledgment

I thank Dr. Takeaki Uno, who told me about the efficient enumeration algorithm for k-spanning trees. My thanks also to Professor Seinosuke Toda and Dr. Koichi Yamazaki for discussions and encouragement. This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Encouragement of Young Scientists, 10780203, 1999.

References

[1] N. Alon, R. Yuster, and U. Zwick. Color-Coding. *Journal of the ACM*, 42(4):844–856, 1995.

[2] D. Avis and K. Fukuda. Reverse Search for Enumeration. *Discrete Applied Mathematics*, 65:21–46, 1996.

[3] L. Babai. Moderately Exponential Bound for Graph Isomorphism. In *Fundamentals of Computer Theory*, pages 34–50. Lecture Notes in Computer Science Vol. 117, Springer-Verlag, 1981.

[4] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan. The Minimum Latency Problem. In *Proc. 26th ACM Symp. on the Theory of Computing*, pages 163–171. ACM, 1994.

[5] R.G. Downey and M.R. Fellows. Fixed-Parameter Tractability and Completeness I: Basic Results. *SIAM Journal on Computing*, 24(4):873–921, 1995.

[6] R.G. Downey and M.R. Fellows. Fixed-Parameter Tractability and Completeness II: On Comleteness for $W[1]$. *Theoretical Computer Science*, 141:109–131, 1995.

[7] R.G. Downey and M.R. Fellows. *Parameterized Complexity.* Springer, 1999.

[8] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. John Wiley & Sons, Inc., third edition, 1968.

[9] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics.* Addison-Wesley Publishing Company, 1989.

[10] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity.* Birkhäuser, 1993.

[11] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge, 1995. 1995.

[12] R. Ravi, R. Sundaram, M.V. Marathe, D.J. Rosenkrantz, and S.S. Ravi. Spanning Trees Short or Small. In *Proc. 5th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 546–555, 1994.

[13] A. Shioura, A. Tamura, and T. Uno. An Optimal Algorithm for Scanning All Spanning Trees of Undirected Graphs. *SIAM Journal on Computing*, 26(3):678–692, 1997.

[14] T. Uno. Personal communication. 1999.

[15] V.N. Zemlyachenko, N.M. Korneenko, and R.I. Tyshkevich. Graph Isomorphism Problem. *Journal of Soviet Mathematics*, 29(4):1426–1481, 1985.