| | | |
|---|---|---|
| 氏　　名 | 清見　礼 | |
| 学位（専攻分野） | 博士（情報学） | |
| 学位記番号 | 総研大甲第998号 | |
| 学位授与の日付 | 平成１８年９月２９日 | |
| 学位授与の要件 | 複合科学研究科　情報学専攻<br>学位規則第６条第１項該当 | |
| 学位論文題目 | Studies on Subgraph and Supergraph Enumeration Algorithms | |
| 論文審査委員 | 主　査　助教授 | 宇野　毅明 |
| | 教授 | 速水　謙 |
| | 教授 | 新井　紀子 |
| | 教授 | 松井　知己（中央大学） |
| | 助教授 | 上原　隆平（北陸先端科学技術大学院大学） |

# 論文内容の要旨

Listing all the objects that satisfy a specified property, with no duplications, is called "enumeration". For example, the enumeration of substrings contained by a string "abcab" is "a", "b", "c", "ab", "bc", "ca", "abc", "bca", "cab", "abca", "bcab" and "abcab". Enumeration has many applications in engineerings such areas as data mining, optimization, and statistics, for example to find frequent patterns or to draw on some rules satisfied by the inputs. We sometimes use enumeration to prove mathematical theorems. Proving some mathematical theorems, such as the four-color theorem, requires considering whether they are true in so many cases that the help of computers is needed. In such a case, we use enumeration. In this thesis, we study enumeration with particular focus on graph enumeration: enumerating graphs belonging to some graph classes, such as chordal graphs, interval graphs, etc.

From early on, techniques to enumerate things in good ways are studied. Gray created an encoding of numbers so that successive numbers differ in exactly one bit; this encoding is called "Gray code". The concept that two successive objects differ in small part has been used in enumeration. For example, Wells developed an algorithm to list permutations in this way. Bitner et~al.~used Gray code to enumerate k-element subsets of an n-element set. It is important for these researches that the successive two enumerated objects differ in small part. With this property, we can efficiently do some computations that need enumeration, and the question of whether there is a Gray code for a given class is itself an interesting mathematical problem. Classes to be enumerated by this method are often very simple in structure, such as permutations or k-element subsets. Since finding an encoding that satisfies Gray code like properties is difficult and differs for each enumeration problem, little research has treated the enumeration of complicated structures such as the graph classes that we treat in this thesis.

Enumeration is also used widely for solving problems in computer science, in which methods are often very easy, and one does not even recognize enumeration is being used. In the field of combinatorial optimization, we often use the branch-and-bound method to solve integer programming problems. The branching process does enumeration, enumerating all the feasible solutions that satisfy a given condition. The divide-and-conquer method also uses enumeration. It corresponds to the binary partition method in terms of enumeration. In column generation algorithms or in set covering approaches for some optimization problems, solutions of subproblems are enumerated, and an optimal solution is found by combining them. In combinatorial game theory, we use enumeration to find the best move of a player. Indeed, we enumerate all the possible moves and select the move that obtains the best evaluation value in end game. In these enumerations, the searches are done among tree structures, while enumerations using Gray codes search along path structures. Searching along tree structures enables us to enumerate more complicated structures. However, in these area, researchers have more actively studied how to omit enumeration where it is not needed; they have researched how to cut the feasible domain where optimal solutions never exist. With

problems whose feasible domain and objective function are defined strictly, such as integer programming, this is natural, since omitting vain enumeration means that we can solve the problems simply and in a short time.

Recently, due to the increase of computation power, we have come to be able to treat huge amounts of data in practical amounts of time. Additionally, in areas such as genome science, and data mining, enumeration has begun to be used. In these areas, problems are often defined vaguely rather than strictly. Researchers in these areas want o find some meaningful structure in huge data sets. Enumeration algorithms for graph structures such as paths, trees, and cliques are used in frequent pattern mining problems. For example, we can find clusters by enumeration of cliques. Enumeration of bipartite cliques is used in frequent item set mining. In these areas, since problems are not strictly defined, research to obtain many possibly optimal objects or to obtain objects that at least have good properties is important, and enumeration has become a strong tool.

Once enumeration got to be a strong tool, demand surfaced for enumeration to apply more complicated structures. For example, some want to enumerate objects that satisfy some properties and are maximal; others want to enumerate very complicated graph structures in a given graph. For researches to enumerate these complicated structures in short time is thus important.

For that matter, even though we can enumerate objects in a wide class, it does not mean that we can enumerate objects in every subclasses of the class. It is characteristic of enumeration (in contrast, problems in areas such as optimization, can be solved if there is an algorithm for problems of their superclass's). For example, Chapter~4 contains a chordal subgraph enumeration algorithm that enumerates every chordal subgraph in a given graph and does so in a constant time for each, however, there is not developed a constant time interval subgraph enumeration algorithm, although interval graphs are a subclass of chordal graphs. Moreover, although we can of course enumerate every graph of $n$-vertices in constant time, for many graph classes we do not know whether or not we can enumerate them in constant time.

Hence, it is not sufficient to develop an algorithm for solving an enumeration problem that enumerate graphs in a large class. Our results in this thesis is to develop fast algorithms for graph enumeration. In general, the number of objects to be enumerated in an enumeration problem is very large. For example, think about enumeration of spanning trees in complete graph $K_n$. The number of spanning trees is $n^{n-2}$. Thus, even if we take only O(1) time to find each spanning tree, it costs $\Omega(n^{n-2})$ time. We thus need to reduce the time used to output each object in order to keep the total time reasonably short. Moreover, if the number of outputs is polynomial in the input size, enumerating each object in polynomial time in the input size automatically bounds the total time complexity to be polynomial in the input size. In order to use enumeration for solving wide-ranging problems, such as optimization or data mining, enumeration algorithms must be able to enumerate each object in polynomial time in the input size, and the faster this can be done, the better. Enumerating each object in constant time is the best in the sense of time complexity. Thus, we estimate the effic

iency of enumeration algorithms by the time complexities for each output. If the del ay between every consecutive two outputs is always polynomial in the input size, we c all the algorithm "polynomial delay" or simply "polynomial". Even though it is difficult to develop a polynomial delay algorithm for enumeration problems, we c an sometimes develop enumeration algorithms whose total time complexities to solve t he problem are polynomial in the output size. Such enumeration algorithms are called "output polynomial". Polynomial delay enumeration algorithms are always output pol ynomial. These criteria can be used to estimate how an algorithm is output-sensitive.

We also need to keep the total memory space reasonably small, as is the case for so lving other computational problems. As for the space complexity, we use the usual sp ace complexity criterion in enumeration problems, estimating the space complexity by the size of inputs. Subgraph and supergraph enumeration have many applications. T hese types of enumeration are special cases of graph enumeration in complete graphs.

We can thus use them for problems such as frequent pattern mining or optimizatio n, as stated above. One application of the subgraph/supergraph enumeration appears i n graphical modeling, in which we use graphs to model some systems. The vertices co rrespond to random variables, and if two variables have a dependency, we connect th em by an edge. If we know that graphs of a system belong to certain graph class suc h as chordal graphs, we can investigate the system by enumerating such graphs. For example, the system corresponding to chordal graphs is known as the decomposable m odel, and was researched by chordal graph enumeration. However, the number of gra phs of n vertices (n corresponds to the number of random variables) belonging to so me graph class is often very large and enumerating all of them is impractical. In suc h a case, if we know that some variables never have dependencies, we can omit enum erating many systems, and this is done by subgraph enumeration. Similarly, if we kno w that there must be some dependencies among some variables, we can use supergrap h enumeration to reduce the total enumeration time.

Naive algorithms for an enumeration problem often take much time and/or space (t ime exponential in the output size and/or space exponential in the input size). Develo ping an output-sensitive enumeration algorithm that uses a small amount of memory i s an important task. If we use an algorithm that finds neighbors (under some definiti on) recursively, the algorithm often needs to store the objects previously output in m emory in order to avoid making duplicate outputs. However, when we enumerate expo nentially many (in the input size) objects in output-sensitive computational time with a small memory, we have to avoid duplicate outputs without storing previously output objects in memory, since storing them would require the size of the to be exponenti ally large. Further, simple search strategies may fail with some problems. For examp le, branch-and-bound type algorithms are not efficient if the subproblems related to t he bounding operations are hard. Though it is not easy to develop an efficient algorit hm for enumeration problems, efficient algorithms have been provided for some enum eration problems, such as enumerations of vertices of a polytope, all cells in a hyper plane arrangement, spanning trees of a graph, maximal cliques of a graph and perfec t elimination orderings of a chordal graph.

There are some known results about subgraph enumeration. For example, given a graph G=(V,E), we can enumerate paths and cycles in it in polynomial time. The time complexity for one output is O(|E|). Given a graph G, we can enumerate every tree spanning G in constant time. Here, the number of edges of a spanning tree is O(|V|). Thus, we must need O(|V|) time to output a spanning tree of G in the naive sense. However, if the differences of any two consecutive outputs are in constant size, and the algorithm always takes only a constant time to obtain a graph from the previous graph, we say the algorithm takes constant time to enumerate each graph. After the establishment of the reverse search method for enumeration problems by Avis and Fukuda, enumeration algorithms have made a notable amount of progress. Many classes have been proved to be enumerated in polynomial time in the input size. However, many graph classes remain that we do not know whether or not we can enumerate even in polynomial time. Moreover, as for supergraph enumeration problems, little research has been done on them to the best of our knowledge.

In this thesis, we introduce some schemes for graph enumeration for both subgraph enumeration problems and supergraph enumeration problems, and we develop our enumeration algorithms that enumerate each graph in polynomial time using the scheme. The algorithms are for chordal graphs, interval graphs, split graphs, block graphs, Ptolemaic graphs, strongly chordal graphs and weakly chordal graphs. These graph classes (except for chordal graphs itself and weakly chordal graphs) are subclasses of chordal graphs. To the best of our knowledge, our results are the first results about enumeration of these graphs.

The organization of this thesis is as follows. We first introduce enumeration, focusing particularly on graph enumeration. Chapter 2 provides the preliminaries, notes about terms that we use in this thesis, and explanations about graph classes. In Chapter 3, we discuss the difficulties of our enumeration problems, and explain the framework of the classical reverse search method. In Chapter 4, we develop algorithms for our enumeration problems. They are of two types: one defines parents such that the difference between a graph and its parent is exactly one, and the other defines parents such that the parent of a graph is obtained by eliminating a simplicial vertex. And, we conclude the thesis in Chapter 5.

# 論文の審査結果の要旨

この論文では、与えられたグラフの、ある種の性質を満たす部分グラフを列挙する問題と、ある種の性質を満たし、かつ入力したグラフを含むようなグラフを全て列挙する問題に関して研究を行い、効率良いアルゴリズムを提案している。列挙問題とは、与えられた性質を満たす解をすべて重複なく見つけて出力する問題であり、情報科学の中では基本的な問題に属する。列挙問題は非常に多くの解を持ち、それゆえに多大な計算時間を要するので、過去の研究においては、あまり注目されてこなかった。しかし現在、計算機パワーの増大と、解の数をコントロールするなどといったモデルの発達により、列挙は現実に解ける問題となり、データマイニングをはじめとする多くの分野で実際に使われ始めている。また列挙問題そのものが興味深い構造を持っており、数学などの分野では実際に列挙を行うことにより、問題の構造を解明することも多い。

このような歴史から、既存の列挙問題とアルゴリズムに関する研究は、比較的単純なものが多かった。つまり、ある程度自明にアルゴリズムが構築できるようなものに対してのみ、アルゴリズムの研究が行われてきたという側面がある。しかし、列挙アルゴリズムの応用での必要性が増すとともに、より複雑で自明でない問題も注目されてきている。近年、いくつかそのような問題が実際に応用の場面で使われ始めている。今回の清見君の研究では、このような自明でない問題に対して、効率の良いアルゴリズムを提案している。

与えられたグラフに対し、その一部分となっているようなグラフを部分グラフ、そのグラフを含むようなグラフをスーパーグラフという。清見君の研究では、与えられたグラフの部分グラフ、あるいはスーパーグラフの中で、ある種のグラフのクラスに属するもののみを見つけるような問題を扱っている。ある種のグラフのクラスとは、コーダルグラフやインターバルグラフといったグラフのクラスであり、ある種のモデル化に用いられる。これらのグラフを効率良く列挙することで、これらのグラフを用いたモデルに対して、既存の研究では行えなかった効率的な解法が構築できる。

この論文では，1章でこのような列挙アルゴリズムに対する背景の解説を行った後、2章で用語の定義および扱うグラフクラスについて既知の事実を説明している．3章では列挙について，特に，どのような点が高速なアルゴリズムを開発する上で難しいのか，ということを、理論的な事実を元に解析・解説している。特に、この論文で扱っているコーダルグラフやインターバルグラフの列挙問題が、通常の素朴な手法で解こうとすると計算量の意味で困難性があり、高速なアルゴリズムを構築することは自明でないということを示している。また、効率良いアルゴリズムを構築するための技術、逆探索についての説明を行っている．4章で，これらグラフクラスの性質をいくつか証明し、これらの性質を用いてこれらの列挙問題に対する効率的なアルゴリズムを提案している。これらのアルゴリズムはいずれも、多項式時間となっており、これらの問題に対する初の多項式時間アルゴリズムとなっている。

これらの結果は、国内英文論文誌、2つの海外の査読付き国際会議編にて発表ずみであり、研究のレベルも世界的に見て十分高いと考えている。アルゴリズムの構築、グラフクラスに対する性質の証明手法も自明でなく、高度なレベルにあるといえる。以上のことから、本論分は博士（情報学）の学位論文として十分な価値があるものと認められる。