COMPUTATIONAL AND STRUCTURAL BIOTECHNOLOGY JOURNAL

Volume No: 5, Issue: 6, February 2013, e201302004, http://dx.doi.org/10.5936/csbj.201302004

CSBJ

Comparison and Enumeration of Chemical Graphs

Tatsuya Akutsu [a,*], Hiroshi Nagamochi [b]

Abstract: Chemical compounds are usually represented as graph structured data in computers. In this review article, we overview several graph classes relevant to chemical compounds and the computational complexities of several fundamental problems for these graph classes. In particular, we consider the following problems: determining whether two chemical graphs are identical, determining whether one input chemical graph is a part of the other input chemical graph, finding a maximum common part of two input graphs, finding a reaction atom mapping, enumerating possible chemical graphs, and enumerating stereoisomers. We also discuss the relationship between the fifth problem and kernel functions for chemical compounds.

Review Article

Introduction

In computer analysis of chemical compounds, chemical structures are usually represented as graph structured data. Mathematically, a graph consists of a set of vertices and a set of edges, where a vertex represents some object and an edge represents a relation between two objects. From a chemical viewpoint, a graph corresponds to a chemical structural formula, in which a vertex and an edge correspond to an atom and a chemical bond, respectively. Furthermore, an atom type and a bond type are represented by labels of a vertex and an edge, respectively. In this article, graphs with such labels are called *chemical graphs*.

Graphs are a very important concept in computer science, and extensive studies have been done to develop efficient algorithms for a number of graph problems. Among these problems, we focus on fundamental problems relevant to chemical graphs. In particular, we consider comparison and enumeration of chemical graphs because these are fundamental and have a long history in chemoinformatics. For example, unique naming of chemical compounds and enumeration of isomers have been studied for more than 100 years [1], much before the invention of computers. In this article, we consider the following problems:

(i) determining whether two chemical graphs are identical,
(ii) determining whether one input chemical graph is a part of the other input chemical graph,
(iii) finding a maximum common part of two input graphs,
(iv) finding a reaction atom mapping,
(v) enumerating possible chemical graphs,
(vi) enumerating stereoisomers,

where (i) and (v) are closely related to unique naming and enumeration of structural isomers, respectively. We do not intend to provide a comprehensive review on these problems because there are too many methods even for any of these six problems. Instead, we try to clarify the *computational complexities* of them. We also introduce some recent developments on problems (v) and (vi) with focusing on our recent work because our algorithms are based on somewhat different approaches than traditional approaches in chemoinformatics [2] and they have guaranteed computational complexities. Since this review article focuses on time complexity aspects of the problems and algorithms, readers interested in practical and heuristic methods in chemoinformatics are referred to existing books and review articles: [2] for fundamental algorithms, [3,4] for pattern matching algorithms, [2,5,6] for prediction and regression methods, and [2,7] for enumeration algorithms.

As discussed later, most of the above problems are intractable for general graphs from a viewpoint of computational complexity. However, chemical graphs have several restrictions. For example, the maximum number of bonds connecting to an atom is usually less than 8. Making use of these constraints, it is often possible to develop theoretically efficient algorithms. Therefore, before discussing individual problems, we briefly review graph classes that are relevant to chemoinformatics.

The organization of this article is as follows. First, we review graph classes relevant to chemoinformatics and give a brief introduction of computational complexity. Next, we review theoretical results and some algorithms on problems (i)-(iv). Next, we describe a relationship between kernel methods and enumeration problems, where kernel methods are a kind of machine learning method and have been applied to various chemoinformatics problems. Then, we review our recent algorithms for problems (v) and (vi) because they are based on state-of-the-art techniques in graph algorithms and thus may bring new methodologies into chemoinformatics. Finally, we conclude with future work. For the purpose of simplicity of presentation, we do not give formal definitions, instead explain terms and results by using words and figures.

[a]*Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho, Uji, Kyoto 611-0011, Japan*
[b]*Graduate School of Informatics, Kyoto University, Yoshida, Kyoto 606-8501, Japan*

* Corresponding author. Tel.: +81 774383015 ; Fax: +81 774383022
E-mail address: takutsu@kuicr.kyoto-u.ac.jp (Tatsuya Akutsu)

OPEN ACCESS

Graphs and chemical compounds

Chemical graphs

A *graph* consists of a set of *vertices* and a set of *edges*, where a vertex and an edge correspond to an atom and a chemical bond, respectively. Each graph is denoted as $G(V,E)$ where V denotes a set of vertices and E denotes a set of edges. There are two kinds of graphs: *directed graphs* and *undirected graphs*. Each edge has a direction in directed graphs, whereas no edge has a direction in undirected graphs. Since there is usually no explicit direction in chemical bonds, we only consider undirected graphs in this article and thus each edge is represented by a set of two vertices (i.e., two atoms connected by the corresponding chemical bond).

In order to associate chemical structures to graphs, we employ *labels* of vertices and edges. Each vertex v has a label $l(v)$, which represents an atom type (e.g., $l(v)$=`C' if v corresponds to a carbon atom). In this article, a graph with vertex and edge labels defined as above is called a *chemical graph*. There are two ways to represent a chemical bond with multiplicity:

(i) multiplicity is represented by multi-edges (e.g., double bond is represented by two edges),
(ii) multiplicity is represented by a label of an edge (e.g., $l(e)$=2 if an edge e corresponds a double bond).

We mainly consider the latter way of representing chemical bonds in this article, where $l(e)$=1.5 may represent an aromatic bond. The *degree* of a vertex is defined as the number of edges connecting to it, and is closely related to the valence of an atom. A graph with a designated vertex r is called a graph *rooted* at a vertex r. Isomorphism between two rooted graphs assumes that the roots of the two graphs correspond each other. In this paper, we utilize a fast algorithm designed for enumerating rooted trees. However, we designate as the root of a tree a special vertex of the tree which is uniquely determined by the topological structure only, and thereby our algorithms effectively enumerate ``unrooted'' trees.

Graph classes

As mentioned above, chemical structures can be represented as graphs. However, we need not consider all kinds of graphs. For example, it is known that most atoms have valence at most 8, which implies that the maximum degree of chemical graphs is at most 8. Therefore, it is enough for chemical structures to consider graphs with bounded degree. In what follows, we only consider bounded degree graphs.

As discussed later, many graph problems can be solved much faster if we restrict types of graphs. Therefore, we review here several graph classes that are relevant to chemical applications (see Figure 1). For details of graph classes, see [8].

Tree: A graph is called a *tree* if it is connected and does not have a loop, where `connected' means that there exists a path (a sequence of connected edges) connecting any pair of vertices. Trees are one of the simplest graphs and many problems can be solved much more efficiently for trees than for general graphs.

Outerplanar graph: A graph is called *outerplanar* if it can be drawn on a plane in such a way that all vertices lie on the outer face without crossing of edges, where the outer face is the unbounded exterior region. Trees are a subclass of outerplanar graphs.

Almost tree: A graph is called an *almost tree* (with parameter k) if each biconnected component (i.e., maximal non-tree part) is obtained by adding at most k edges to a tree. Trees are a subclass of almost trees (i.e., k=0). However, outerplanar graphs are not a subclass of almost trees or almost trees are not a subclass of outerplanar graphs.

Partial k-tree: A graph is called a *partial k-tree* if it is transformed into a tree by regarding a family of subsets of vertices as a set of new vertices (i.e., by tree decomposition), where each subset consists of at most k+1 vertices (Figure 2). Trees, outerplanar graphs, and almost trees with parameter k are subclasses of partial 1-trees, partial 2-trees, and partial k+1-trees, respectively [9].

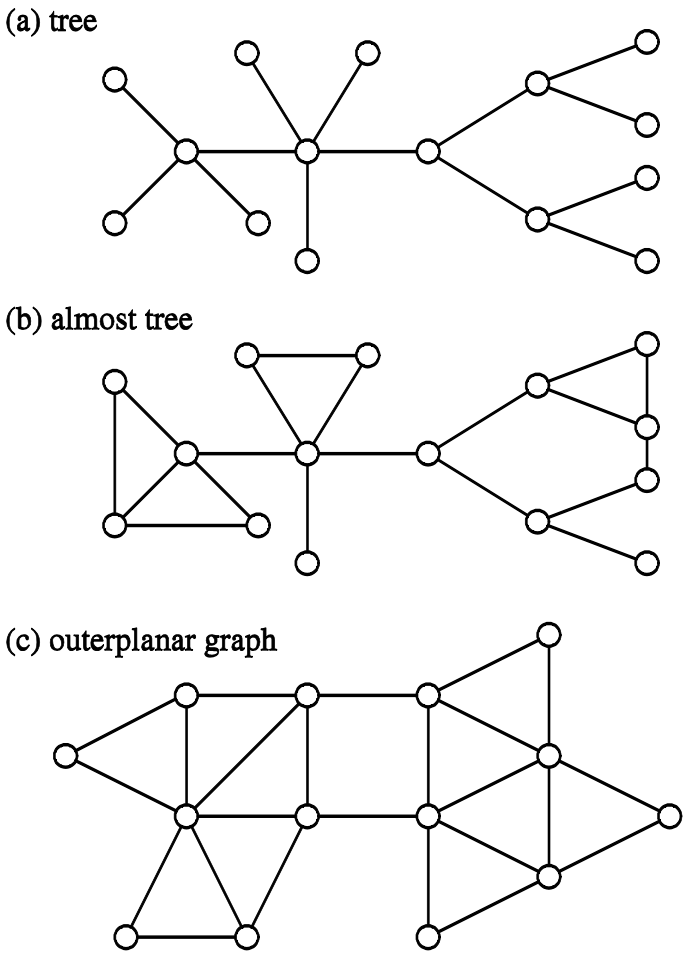

Figure 1. Examples of (a) tree, (b) almost tree, and (c) outerplanar graph, where k=2 in (b).

Yamaguchi et al. studied the distribution of partial k-trees in chemical graphs [10]. Horváth and Ramon also studied the distribution of partial k-trees in some dataset and reported that 8.77%, 97.35% and 99.97% of compounds are partial 1-trees, 2-trees, and 3-trees, respectively and most partial 2-tree compounds are outerplanar [11].

Computational complexity

Here, we briefly review some basic concepts in computational complexity. If the computation time of an algorithm is proportional to n^d (i.e., the computation time is $O(n^d)$) for some constant d where n denotes the size of an input data, the algorithm is said to be a *polynomial-time algorithm*. There exist many problems that do not have polynomial-time algorithms. Although we do not explain details, *NP-hard* problems are widely believed not to have polynomial-time algorithms. In theoretical computer science, polynomial-time algorithms are regarded as efficient algorithms whereas NP-hard

Download English Version:

https://daneshyari.com/en/article/2079243

Download Persian Version:

https://daneshyari.com/article/2079243

Daneshyari.com