# Chapter 6
# Graph Mining in Chemoinformatics

**Hiroto Saigo**
*Kyushu Institute of Technology, Japan*

**Koji Tsuda**
*AIST Computational Biology Research Center, Japan*

## ABSTRACT

*In standard QSAR (Quantitative Structure Activity Relationship) approaches, chemical compounds are represented as a set of physicochemical property descriptors, which are then used as numerical features for classification or regression. However, standard descriptors such as structural keys and fingerprints are not comprehensive enough in many cases. Since chemical compounds are naturally represented as attributed graphs, graph mining techniques allow us to create subgraph patterns (i.e., structural motifs) that can be used as additional descriptors. In this chapter, the authors present theoretically motivated QSAR algorithms that can automatically identify informative subgraph patterns. A graph mining subroutine is embedded in the mother algorithm and it is called repeatedly to collect patterns progressively. The authors present three variations that build on support vector machines (SVM), partial least squares regression (PLS) and least angle regression (LARS). In comparison to graph kernels, our methods are more interpretable, thereby allows chemists to identify salient subgraph features to improve the drug-likeliness of lead compounds.*

## INTRODUCTION

In the first step of drug discovery process, a large number of lead compounds are found by high throughput screening. To identify physicochemical properties of the lead compounds, SAR and QSAR analyses are commonly applied (Gasteiger & Engel, 2003). In machine learning terminology, SAR is understood as a classification task where a chemical compound is given as an input, and the learning machine predicts the value of a binary output variable indicating the activity. In QSAR, the output variable is real-valued and it is a regression task.

For accurate prediction, numerical features that characterize physicochemical properties are

DOI: 10.4018/978-1-61520-911-8.ch006

necessary. A wide range of feature *descriptors* are proposed (Gasteiger & Engel, 2003). A classical example is *structural keys*, where each chemical compound is represented as a binary vector representing the presence or absence of major functional groups such as alcohols and amines. Notice that relevant structural keys differs from problem to problem. For example, structural keys used in pharmaceutics is by far different from the ones in petrochemics. A *fingerprint* is another approach that enumerates all the structures under a given constraint. This approach enumerates all paths up to a certain length in each chemical compound. The paths are used to represent the compounds as a fixed length binary vector. To save the storage, several different patterns have to be assigned to the same bit (*collision*). For this reason, a fingerprint does not always gives us a transparent and interpretable model. Another limitation of the current fingerprinting is the use of path patterns, despite the fact that tree and graph patterns are more informative (Yan, Yu, & Han, 2004).

The use of *frequent subgraphs* as descriptors is studied recently in the data mining community (Wale & Karypis, 2006, Kazius, Nijssen, Kok, Bäck, & Ijzerman, 2006, Helma, Cramer, Kramer, & Raedt, 2004). Frequent subgraph enumeration algorithms such as AGM (Inokuchi, 2005), Gaston (Nijssen & Kok, 2004) and gSpan (Yan & Han, 2002a) can enumerate all the subgraph patterns that appear more than m times in a graph database. The threshold m is called *minimum support*. Frequent subgraph patterns are found by branch-and-bound search in a tree shaped search space (Figure 2). Frequent subgraphs contain good descriptors, but they are often redundant. It is known that, to achieve the best accuracy, the minimum support has to be determined to a small value (e.g., 3-5) (Wale & Karypis, 2006, Kazius et al., 2006, Helma et al., 2004). Such setting creates millions of patterns, which makes subsequent processing difficult. Frequent patterns are not informative, for example, patterns like C-C and C-C-C would be frequent but have almost no information.

*Figure 2. Schematic figure of the tree-shaped search space of graph patterns (i.e., the DFS code tree). To find the optimal pattern efficiently, the tree is systematically expanded by rightmost extensions*