Subgraph-Feature Search for Learning Classifiers and Regressors under Fixed Budget Constraint

Graduate school of Infomation Science and Technology, Hokkaido Universisy

Ryo Shirakawa

Contents

List of Figures

List of Tables

Abstract

Recently, in various fields such as computational chemistry and bioinformatics, attention has been paid to classification and regression of data whose inputs are graphs of arbitrary size and shape , sometimes called graph classification/regression. In these problems, whether a certain subgraph is included or not is a fundamental feature for graphs that have no direct descriptors available. The simplest solution, enumerating subgraph patterns and use them as features, is so expensive since the number of possible subgraph patterns are intractably large due to the combinatorial explosion. Currently, discriminative pattern mining has been studied and given a good solution to this problem. The previous search method for such discriminative subgraph patterns tries to find the best one by the depth first search with some pruning rules. Such a exact search, however, is costly for heavily repeated use, which is necessary to obtain the number of features enough for high performance classifiers or regressors. To overcome this computational cost issue, we propose two approximate algorithms based on (*i*) best first search, (*ii*) Monte Carlo Tree Search(MCTS). We demonstrate effectiveness of our proposed methods by comparing performance in real problems using QSAR datasets.

1 Introduction

Graphs are fundamental data structures for representing combinatorial objects such as chemical compounds, networks and others. However, precisely because of their combinatorial nature, it is usually difficult to understand the underlying trends in large datasets of graphs. The rapid increase in data in recent years also includes data represented as graphs, and thus supervised learning in which the inputs are graphs of arbitrary size and shape has gained considerable attention in the fields of computer vision [1–4], chemoinformatics [5–13], bioinformatics [14–16] and computational chemistry [17, 18] and natural language processing [19].

Some of these problems have graph descriptors prepared in advance, such as molecular dataset [20,21], but others have no direct descriptors available. For these problems, the most fundamental features are subgraph indicators, i.e., if a subgraph is embedded in the given graph or not, because many reactions and events are attributed to those substructures. However, the number of all possible subgraph patterns is intractably large due to the combinatorial explosion, so that it is difficult to enumerate all subgraphs in a realistic time for a given dataset. Therefore, it is necessary to restrict the subgraph candidate on the basis of some criterion. Frequent subgraph mining [22,23] is one method in this approach. However, since it chooses subgraphs according to the frequencies, it is possible to overlook the importance of subgraphs. On the other hand, discriminative pattern mining techniques [8, 24, 25] choose subgraphs according to a model-based discriminative criteria, and these not likely to overlook important patterns for learning. Because of these merits, the present paper also adopts this approach.

1.1 Related Work

Related works [8,25] search the discriminative subgraph pattern using depth first policy with branch and bound method. While these methods perform efficient searches by designing tricky pruning, search policy of depth first is naive and they still suffer from computational costs for some domain or large datasets.

1.2 Our Approach

To overcome the above difficulty, we propose the more efficient two algorithms using Best First Search and Monte Carlo tree search (MCTS). The efficiency of these building-block algorithms are already known in some domains, so that in this paper we apply them to our graph classification/regression problem.

2 Preliminaries

2.1 Notations

Let $[n]$ be a set of integers $\{1, 2, \ldots, n\}$ and let $\mathbb{I}(P)$ be the indicator of a predicate P, i.e., $\mathbb{I}(P) = 1$ if P is true, else 0. A labeled graph is represented by a 4-tuple $G = (V, E, \mathcal{L}, l)$, where V is a set of vertices, $E \subset V \times V$ is a set of undirected edges, $\mathcal{L}$ is a set of labels, and $l : V \cup E \to \mathcal{L}$ is a mapping that assigns a label to each vertex or edge. We denote a subgraph isomorphism, i.e., if g is a subgraph of G, as $G \sqsupseteq g$ and its negation as $G \not\sqsupseteq g$. Thus, a subgraph indicator $\mathbb{I}(G \sqsupseteq g) = 1$ if $G \sqsupseteq g$, otherwise 0. We also denote the training set of pairs of input graph $G \in \mathcal{G}$ and its output responses $y \in \mathcal{Y}$ as

$$\mathcal{D} = \{(G_1, y_1), (G_2, y_2), \ldots, (G_N, y_N)\}. \tag{1}$$

We assume that $\mathcal{G}$ is a set of all finite-size, connected, discretely-labeled, undirected graphs. We denote any set of graphs of size N by $\mathcal{G}_N = \{G_i \mid i \in [N]\}$, and the set of all possible connected subgraphs as $\mathcal{S}_N = \bigcup_{G \in \mathcal{G}_N} \{g \mid G \sqsupseteq g\}$.

Definition 1 (subgraph isomorphism) For two graphs, $G' = (V', E', \mathcal{L}', l')$ and $G = (V, E, \mathcal{L}, l)$, G' is subgraph isomorpic to G iff there exist an injective mapping $\phi : V' \to V$, s.t., (1) $\forall v \in V', l'(v) = l(\phi(v))$, (2) $\forall (v_1, v_2) \in E', (\phi(v_1), \phi(v_2)) \in E$ and (3) $l'(v_1, v_2) = l(\phi(v_1), \phi(v_2))$.

2.2 Search Space for Subgraphs

In supervised learning from graphs, we represent each input graph $G_i \in \mathcal{G}_N$ by the characteristic vector $(\mathbb{I}(G_i \sqsupseteq g) \mid g \in \mathcal{S})$ with a set $\mathcal{S}$ of relevant subgraph features. However, since $\mathcal{S}$ is not explicitly available when the learning phase starts, we need to simultaneously do searching and constructing of $\mathcal{S}$ during the learning process. In order to define an efficient search space for $\mathcal{S}_N$, i.e., any subgraphs occurring in $\mathcal{G}_N$, the techniques for *frequent subgraph*

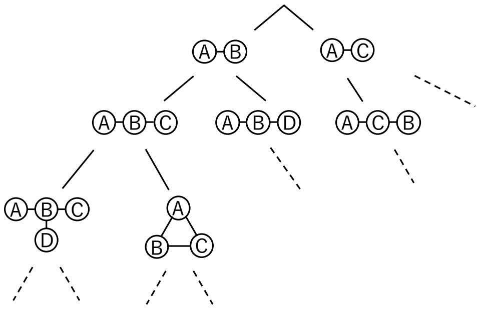


Figure 1: DFS code tree. Search space for subgraphs that appear in the training dataset.

mining [22,23] are useful. Note that any subgraph feature $g \in \mathcal{S}_N$ can occur multiple times at multiple locations in a single graph, even if $\mathbb{I}(G_i \sqsupseteq g) = 1$.

In the present paper, we use the search space of the gSpan algorithm [22], which performs a depth-first search on the tree-shaped search spaces on $\mathcal{S}_N$, referred to collectively as an *DFS code tree*, as shown in Figure 1. Each node of the DFS code tree holds a subgraph feature g' that extends the subgraph feature g at the parent node by one edge, namely, $g' \sqsupseteq g$.

The following *anti-monotone* property of subgraph isomorphism over the DFS code tree on $\mathcal{S}_N$ can be used to derive the efficient search-space pruning of the gSpan algorithm:

$$g' \sqsupseteq g, \quad G_i \not\sqsupseteq g \Rightarrow G_i \not\sqsupseteq g'. \tag{2}$$

2.3 Discriminative Criterion

We the discriminating power of each subgraph by the following formula. When predicting the graph labels with discrete values, *DiscriminativeScore* (*DScore*) is used defined as (3).

$$\begin{aligned} DScore(g) &= \mathrm{TSS}(\mathcal{D}_1(g)) + \mathrm{TSS}(\mathcal{D}_0(g)) \\ \mathrm{TSS}(\mathcal{D}) &= \sum_{i\in[N]} (y_i - \bar{y})^2,\ \bar{y} = \frac{1}{N}\sum_{i\in[N]} y_i. \end{aligned} \tag{3}$$

where TSS is the total sum of squared error and $\mathcal{D}_1(g) = \{(G_i, y_i) \in \mathcal{D} \mid G \sqsupseteq g\}$ and $\mathcal{D}_0(g) = \{(G_i, y_i) \in \mathcal{D} \mid G \not\sqsupseteq g\}$.

3 Problem Setting and Challenges

Our problem is to search for a subgraph g that minimizes (3). However, the search space of subgraphs (Section 2.2) is intractably large due to the combinatorial explosion. The naive search policy require very high computational costs and it is difficult to find the suitable solution. Our challenge is to design an efficient policy for searching subgraphs in a large search space based on the discriminative criteria.

4 Depth First Search with Branch and Bound

The state of the art methods [8, 25] searching for a discriminative subgraph pattern is typically based on the depth first search, as shown in Figure 2 (a), with a branch and bound technique. [25] searches the best subgraph pattern using $DScore$. This method calculates the lower bound of $DScore$ obtained by child nodes of search space, and if this lower bound does not reach the provisional solution the child is pruned.

Theorem 1 *Given $\mathcal{D}_1(g)$ and $\mathcal{D}_0(g)$, for any subgraph $g' \sqsupseteq g$,*

$$DScore(g') \;\; \geq \;\; \min_{(\diamond,\, k)} \Big[\mathrm{TSS}(\mathcal{D}_1(g) \; \setminus \; S_{\diamond,k}) \; + \; \mathrm{TSS}(\mathcal{D}_0(g) \; \cup \; S_{\diamond,k}) \Big] \quad (4)$$

where $(\diamond, k) \in \{\leq, >\} \times \{2, \ldots, |\mathcal{D}_1(g) - 1|\}$, and $S_{\diamond,k} \subset \mathcal{D}_1(g)$, such that $S_{\leq,k}$ is a set of k pair (G_i, r_i) selected from $\mathcal{D}_1(g)$ in descending order of residual error r_i, and $S_{>,k}$ is that in increasing order. Note that $\setminus$, $\cup$ are set difference and set union respectively.

Proof 1 *Given $\mathcal{D}_1(g)$ and $\mathcal{D}_0(g)$,*

$$\begin{aligned}\mathsf{bound} &= \min_{g'}\big[\mathrm{TSS}(\mathcal{D}_1(g')) + \mathrm{TSS}(\mathcal{D}_0(g'))\big] \\ &= \min_{S\,\subset\,\mathcal{D}_1(g)}\big[\mathrm{TSS}(\mathcal{D}_1(g)\setminus S) + \mathrm{TSS}(\mathcal{D}_0(g)\cup S)\big] \qquad (5) \\ &= \min_{(\diamond,k)}\Big[\mathrm{TSS}(\mathcal{D}_1(g)\setminus S_{\diamond,k}) + \mathrm{TSS}(\mathcal{D}_0(g)\cup S_{\diamond,k})\Big] \qquad (6)\end{aligned}$$

where $(\diamond, k) \in \{\leq, >\} \times \{2, \ldots, |\mathcal{D}_1(g) - 1|\}$. From the anti-monotone property (2), we have $\mathcal{D}_1(g') \subseteq \mathcal{D}_1(g)$ for $g' \sqsupseteq g$ for the training set $\mathcal{D}$ from which the equation (5) directly follows. Thus, we show (6) in detail. For simplicity, let $A =, \{a_1, \ldots, a_n \mid a_i \in \mathbb{R}\}$ denote $\mathcal{D}_1(g)$, and $B = \{b_1, \ldots, b_m \mid b_i \in \mathbb{R}\}$ denote $\mathcal{D}_0(g)$. Then, the goal of (5) is to minimize the total sum of squares $\mathrm{TSS}(A \setminus S) + \mathrm{TSS}(B \cup S)$ by tweaking $S = \{s_1, \ldots, s_k\} \subset A$. The key fact is that $\mathrm{TSS}(A \setminus S) + \mathrm{TSS}(B \cup S)$ can be regarded as a quadratic equation of $\sum_{i=1}^k s_i$ when the size of S is fixed to k. More precisely,

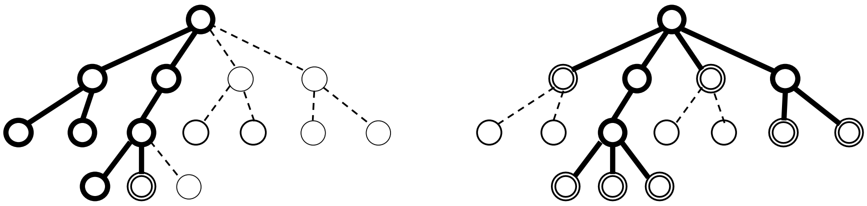
$$\mathrm{TSS}(A) = \sum_{i\in[n]}\Big(a_i - \frac{\sum_{i\in[n]} a_i}{|A|}\Big)^2 = \sum_{i\in[n]} a_i^2 - \frac{\big(\sum_{i\in[n]} a_i\big)^2}{|A|}$$

$$\mathrm{TSS}(A) + \mathrm{TSS}(B) = \sum_{i\in[n]} a_i^2 + \sum_{i\in[m]} b_i^2 - \frac{\big(\sum_{i\in[n]} a_i\big)^2}{|A|} - \frac{\big(\sum_{i\in[m]} b_i\big)^2}{|B|}$$

$$\begin{aligned}&\mathrm{TSS}(A\backslash S) + \mathrm{TSS}(B \cup S) \\ &= \sum_{i\in[n]} a_i^2 + \sum_{i\in[m]} b_i^2 - \frac{\big(\sum_{i\in[n]} a_i - \sum_{i\in[k]} s_i\big)^2}{|A\backslash S|} - \frac{\big(\sum_{i\in[m]} b_i - \sum_{i\in[k]} s_i\big)^2}{|B \cup S|} \\ &= \sum_{i\in[n]} a_i^2 + \sum_{i\in[m]} b_i^2 - \frac{\big(\sum_{i\in[n]} a_i - \sum_{i\in[k]} s_i\big)^2}{n-k} - \frac{\big(\sum_{i\in[m]} b_i - \sum_{i\in[k]} s_i\big)^2}{m+k}\end{aligned}$$

Therefore, $\mathrm{TSS}(A\backslash S)+\mathrm{TSS}(B\cup S)$ is minimized when $\sum_{i=1}^k s_i$ is maximized or minimized. In other words, (5) becomes minimum when the mean of $S \subset \mathcal{D}_1(g)$ is maximized or minimized. □

order



(a) Depth-first Search

(b) Best-first Search

Figure 2: Depth-first vs. Best-first. The bold circles show already searched nodes and double circles show the candidate nodes for next expansion

5 Proposed Method

The previous method [25] employs a simple depth-first search policy without referring to the discriminative criterion value ($DScore$) and its bound of each node. In some cases, this criterion and its bound are useful for designing search policies. In this section, using this criterion, we present two efficient methods to search a discriminative subgraph pattern based on (i) Best First Search and (ii) Monte Carlo Tree Search.

5.1 Best First Search

Best-first searching [26] is a search policy that expands from the best node based on some evaluation value, shown as Figure 2 (b). The A* algorithm [27] and Dijkstra algorithm [28] are representative.

The previous method designed the pruning rule by calculating the bound value in (4). In the proposed method, this bound value is set as the search priority of each node, and is applied to the best-first search. The pseudocode is shown in Algorithm. 1. The search starts from the root node, and selects the node with the highest bound from the expansion candidate set. The child nodes of the selected node are expanded, and the score is calculated and added to the expansion candidate set. The above operation is repeated until pruning is possible for all expansion candidate set.

Alg. 1: Subgraph Search by Best First Search

```
Input: Training data $\mathcal{D} = \{(G_1, r_1), (G_2, r_2), \ldots, (G_N, r_N)\}$
Output: Best score subgraph $g^*$
Function Best First Search($\mathcal{D}$)
    $candidate \leftarrow \{root\}$ repeat
        $g \leftarrow \underset{candidate\ c}{\arg\max}\ bound(c)$ ;
        $candidate \leftarrow candidate / g$ ;
        forall child c of g do
            if Score(c) is better than score* then
                $score^* \leftarrow Score(c)$ ;
                $g^* \leftarrow c$ ;
            end
            $candidate \leftarrow candidate \cup c$ ;
        end
    until all candidate can be pruned;
    return $g^*$ ;
```

5.2 Monte Carlo Tree Search

We consider to apply Monte Carlo tree search (MCTS) [29–31] to our problem. MCTS is widely known as the method used in computer Go. IT is also applied in various fields such as feature selection. One of them, Upper Confidence Bounds for Tree (UCT) algorithm [29] is empirically highly evaluated as a way to find good solutions within a limited budget. This method resolves the exploration-exploitation problem using Upper Confidence Bound (UCB), which allows to estimate the expected reward of each child. The simplest UCB policy is called UCB1 and does not require the prior specific knowledge. The policy selects to search child node i that maximizes

$$UCB1 = \bar{X}_i + C \times \sqrt{\frac{\ln n}{2n_i}} \tag{7}$$

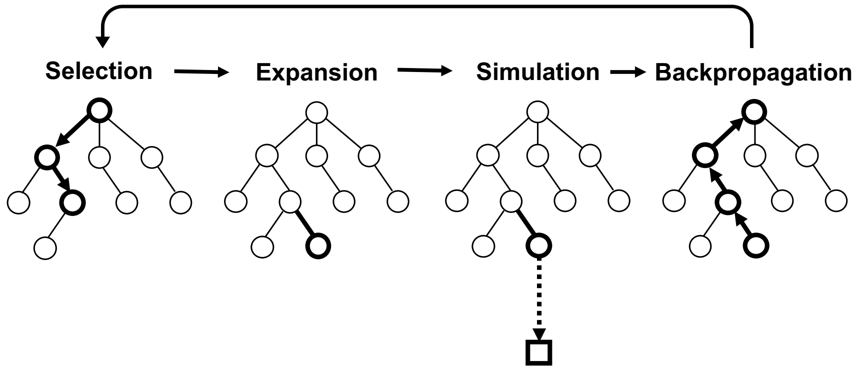


Figure 3: One circle of processes in UCT

where $\bar{X}_i$ is the average reward from child node i, n_i is the number of times child node i was selected, n is the number of times parent node of i was selected, and C is an exploration constant. The left term encourages the exploitation of high expected reward child, and the right term encourages the exploration of less visited child.

The UCT algorithm is an iterative method doing the four operations of selection, expansion, simulation, and backpropagation up to a predefined computational cost, shown as Figure 3.

1) Selection:

   Start from the root node, and select the child repeatedly until the expandable node is reached based on the value of (7). A node is expandable if it represents a non-terminal state and has unvisited children.

2) Expansion:

   One random child node that is unvisited is expanded. In this operation, we consider the same pruning condition as (4). If the random select node can be pruned, another node is reselected and added.

3) Simulation:

A Monte Carlo simulation is run from the new expanded node, and repeatedly select a child from the available children uniformly and randomly. This simulation stops if the simulation node has no children or is based on a stochastic condition known as a stopping feature used in [30]. The stochastic condition is $1 - q^d$ depending on the simulation node level d, where $q < 1$ is a parameter and $q = 1 - 10^{-1}$ in the present paper.

4) Backpropagation:

Calculate the reward of the node g obtained by simulation. To align the range to $[0, 1]$, reward is defined,

$$reward(g) = -DScore/\mathrm{TSS}(\mathcal{D}) \tag{8}$$

these value will be large if discrimination is successful, otherwise small. Then backpropagate this reward to the nodes selected in selection and expansion phase.

Low Cost Simulation

In this problem setting, the search space is not given in advance. Therefore, we need to construct a search space that matches the given dataset and subgraph search at the same time. When Monte Carlo simulation is performed in this problem, it is necessary to enumerate all the child nodes for each simulation node, which requires a large cost. We design a low-cost simulation method since the simulation cost has a big influence on the search efficiency.

First, a graph including simulation node is randomly selected from the dataset. Expand the subgraph represented by the simulation node on the selected graph. If an expanded subgraph is available(minimum order defined by [22]), this simulation is taken, otherwise it is repeated.

The entire procedure of the proposed algorithm is illustrated with pseudocode in Alg. 2, 3.

Alg. 2: Subgraph Search by UCT

```
Input: Training data $\mathcal{D} = \{(G_1, r_1), (G_2, r_2), \ldots, (G_N, r_N)\}$
Output: Best score subgraph $g^*$
Function UCT($\mathcal{D}$)
    repeat
        $g \leftarrow$ Selection(root) ;
        $g \leftarrow$ Expansion($g$) ;
        if Score(g) is better than score$^*$ then
            score$^* \leftarrow Score(g)$ ;
            $g^* \leftarrow g$;
        end
        $s \leftarrow$ Simulation($g$) ;
        Backpropagation($s, g$) ;
    until predefined cost is exhausted;
    return $g^*$ ;
```

Alg. 3: Four Basic Operations in UCT

```
Function Selection(g)
    while g is expandable do
        $g \leftarrow \underset{children\ c\ of\ g}{\arg\max}\ \bar{X}_c + C \times \sqrt{\frac{\ln n_g}{2n_c}}$ ;
    end
    return $g$ ;
Function Expansion(g)
    repeat
        $g' \leftarrow random(\{children\ c\ of\ g\})$ ;
    until g' is not pruned;
    return $g'$ ;
Function Simulation(g)
    $s \leftarrow g$ ;
    while s has some child and not enough stochastic condition do
        repeat
            $G \leftarrow random(\mathcal{D})$ ;
            $s' \leftarrow random\ expansion\ from\ s\ on\ G$ ;
        until $s'$ is enough to minimum order;
        $s \leftarrow s'$ ;
    end
    return $s$ ;
Function Backpropagation(s, g)
    $X = Score(s)$ ;
    while $g \neq NULL$ do
        $n_g \leftarrow n_g + 1$ ;
        $\bar{X}_g \leftarrow \frac{\bar{X}_g \times (n_g - 1) + X}{n_g}$ ;
        $g \leftarrow parent\ of\ g$ ;
    end
```

6 Experiments

6.1 Datasets

We also evaluate the performance based on the most typical benchmark on real datasets: the quantitative structure-activity relationship (QSAR). We select Four binary-classification datasets (CPDB, Mutag, AIDS(CAvsCM), CAS) in Table 1: CPDB and Mutag for mutagenicity tests and AIDS(CAvsCM) for antiviral tests. All chemical structures are encoded as molecular graphs using RDKit[1], and some structures in the raw data are removed by chemical sanitization[2]. We simply apply a node labeling by the RDKit default atom invariants (edges not labeled), i.e., atom type, # of non-H neighbors, # of Hs, charge, isotope, and inRing properties. These default atom invariants use connectivity information similar to that used for the well-known ECFP family of fingerprints [32]. See [17] for more elaborate encodings.

Table 1: Dataset summary

Dataset	CPDB	Mutag	AIDS(CAvsCM)	CAS
# data	684	188	1503	4337
# ($y = +1, -1$)	(341, 343)	(125, 63)	(422, 1081)	(2401, 1936)
# nodes	25.2	26.3	59.0	30.3
# edges	25.6	28.1	61.6	31.3

# of nodes and edges are average.

6.2 UNDER CONSTRUCTION

7 Conclusion

[1] `http://www.rdkit.org/`

[2] Due to this pre-processing, the number of datasets differs from that in the simple molecular graphs in the literature, where the nodes are labeled by atom type, and the edges are labeled by bond type.

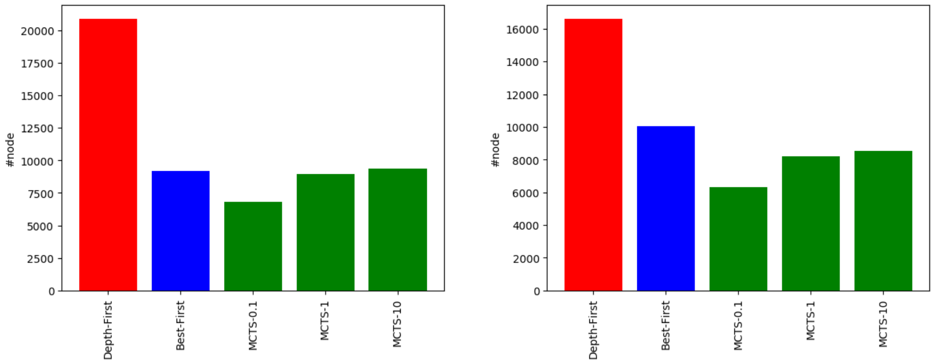


Figure 4: 一つめの図

acknowlege

jadosj

References

[1] Zaïd Harchaoui and Francis Bach. Image classification with segmentation graph kernels. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, Minneapolis, Minnesota, USA, 2007.

[2] Sebastian Nowozin, Koji Tsuda, Takeaki Uno, Taku Kudo, and Gökhan Bakır. Weighted substructure mining for image analysis. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, Minneapolis, Minnesota, USA, 2007.

[3] Vincent Barra and Silvia Biasotti. 3D shape retrieval using kernels on extended reeb graphs. *Pattern Recognition*, Vol. 46, No. 11, pp. 2985–2999, 2013.

[4] Lu Bai, Luca Rossi, Horst Bunke, and Edwin R. Hancock. Attributed graph kernels using the jensen-tsallis q-differences. In *Proceedings of the 2014 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2014)*, pp. 99–114, Nancy, France, 2014.

[5] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pp. 321–328, Washington, DC, USA, 2003.

[6] Koji Tsuda. Entire regularization paths for graph data. In *Proceedings of the 24th International Conference on Machine learning (ICML)*, pp. 919–926, Banff, Alberta, Canada, 2007.

[7] Hiroto Saigo, Nicole Krämer, and Koji Tsuda. Partial least squares regression for graph mining. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 578–586, Las Vegas, Nevada, USA, 2008.

[8] Hiroto Saigo, Sebastian Nowozin, Tadashi Kadowaki, Taku Kudo, and Koji Tsuda. gBoost: a mathematical programming approach to graph classification and regression. *Machine Learning*, Vol. 75, pp. 69–89, 2009.

[9] Pierre Mahé and Jean-Philippe Vert. Graph kernels based on tree patterns for molecules. *Machine Learning*, Vol. 75, No. 1, pp. 3–35, 2009.

[10] S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. Graph kernels. *Journal of Machine Learning Research*, Vol. 11, pp. 1201–1242, 2010.

[11] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research*, Vol. 12, No. Sep, pp. 2539–2561, 2011.

[12] Ichigaku Takigawa and Hiroshi Mamitsuka. Graph mining: procedure, application to drug discovery and recent advances. *Drug Discovery Today*, Vol. 18, No. 1-2, pp. 50–57, 2013.

[13] Ichigaku Takigawa and Hiroshi Mamitsuka. Generalized sparse learning of linear models over the complete subgraph feature set. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 39, No. 3, pp. 617–624, 2017.

[14] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, Vol. 21, No. 1, pp. i47–i56, 2005.

[15] Yan Karklin, Richard F. Meraz, and Stephen R. Holbrook. Classification of non-coding RNA using graph representations of secondary structure. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, pp. 4–15, Hawaii, USA, 2005.

[16] Ichigaku Takigawa, Koji Tsuda, and Hiroshi Mamitsuka. Mining significant substructure pairs for interpreting polypharmacology in drug-target network. *PLoS ONE*, Vol. 6, No. 2, p. e16999, 2011.

[17] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, Vol. 30, No. 8, pp. 595–608, Aug 2016.

[18] Justin Gilmer, Samuel Schoenholz, Patrick F. Riley, Oriol Vinyals, and George Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, Washington, DC, USA, 2017.

[19] Taku Kudo, Eisaku Maeda, and Yuji Matsumoto. An application of boosting to graph classification. In Lawrence K. Saul, Yair Weiss, and Léon Bottou,

editors, *Advances in Neural Information Processing Systems 17*, pp. 729–736. MIT Press, Cambridge, MA, 2005.

[20] C. A. James, Weininger, D., and J. Delany. Daylight theory manual. 2004.

[21] Joseph L Durant, Burton A Leland, Douglas R Henry, and James G Nourse. Reoptimization of mdl keys for use in drug discovery. *Journal of chemical information and computer sciences*, Vol. 42, No. 6, pp. 1273–1280, 2002.

[22] Xifeng Yan and Jiawei Han. gSpan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM)*, pp. 721–724, Washington, DC, USA, 2002.

[23] Siegfried Nijssen and Joost N. Kok. A quickstart in frequent structure mining can make a difference. In *Proceeding of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 647–652, Seattle, Washington, USA, 2004.

[24] Wei Fan, Kun Zhang, Hong Cheng, Jing Gao, Xifeng Yan, Jiawei Han, Philip Yu, and Olivier Verscheure. Direct mining of discriminative and essential frequent patterns via model-based search tree. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pp. 230–238, New York, NY, USA, 2008. ACM.

[25] Ryo Shirakawa, Yusei Yokoyama, Fumiya Okazaki, and Ichigaku Takigawa. Jointly learning relevant subgraph patterns and nonlinear models of their indicators. *CoRR*, Vol. abs/1807.02963, , 2018.

[26] J. Pearl. Heuristics: Intelligent search strategies for computer problem solving. 1 1984.

[27] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, pp. 100–107, 1968.

[28] Edsger W Dijkstra, et al. A note on two problems in connexion with graphs. *Numerische mathematik*, Vol. 1, No. 1, pp. 269–271, 1959.

[29] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings*, pp. 282–293, 2006.

[30] Romaric Gaudel and Michèle Sebag. Feature selection as a one-player game. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pp. 359–366, 2010.

[31] Cameron Browne, Edward Jack Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez Liebana, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Trans. Comput. Intellig. and AI in Games*, Vol. 4, No. 1, pp. 1–43, 2012.

[32] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, Vol. 50, No. 5, pp. 742–754, 2010.