On Inference of a Chemical Structure from Path Frequency

Tatsuya Akutsu[1] **Daiji Fukagawa**[2]
[1] *Bioinformatics Center, Institute for Chemical Research, Kyoto University, Kyoto 611-0011, Japan*
[2] *Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan*
Email: takutsu@kuicr.kyoto-u.ac.jp, daiji@kuicr.kyoto-u.ac.jp

ABSTRACT: This paper studies the problem of inferring a chemical compound from a feature vector consisting of the numbers of occurrences of vertex-labeled paths, which has potential future applications for designing new chemical compounds based on the kernel methods. This paper shows that the problem for outerplanar graphs of bounded degree can be solved in polynomial time if an alphabet is fixed and the maximum length of paths and the number of edges of each face are bounded by a constant. It is also shown that the problem is strongly NP-hard even for trees of unbounded degree.

1 INTRODUCTION

Kernel methods have been applied to various classification problems in bioinformatics [16]. In bioinformatics applications, it is usually required to develop a mapping from the set of objects in the target problem to a *feature space* (i.e., each object is transformed to a vector of reals) and a kernel function is defined as an inner product between two *feature vectors*. In some cases, a feature space can be an infinite dimensional space (Hilbert space) and some kernel trick is developed to compute the value of a kernel function efficiently without explicitly computing feature vectors [7].

Though kernel methods have been used mainly for classification problems, a new approach was recently proposed for designing and/or optimizing objects using kernel methods [4, 5] (see also Fig. 1). In this approach, a desired object is computed as a point in the feature space using suitable objective function and then the point is mapped back to the input space, where this mapped back object is called a *pre-image*. Let ϕ be a mapping from an input space $\mathcal{G}$ to a feature space $\mathcal{F}$. The pre-image problem is, given a point y in $\mathcal{F}$, to find x in $\mathcal{G}$ such that $y = \phi(x)$, where such x is called a pre-image. It should be noted that ϕ is not necessarily injective or surjective. If ϕ is not surjective, we need to compute the approximate pre-image x^* for which the distance between y and $\phi(x)$ is minimized: $x^* = \arg\min_x \; dist(y, \phi(x))$. Bakir, Weston and Scölkopf proposed a method to find pre-images in a general setting by using Kernel Principal Component Analysis and regression [4]. Bakir, Zien and Tsuda developed a stochastic search algorithm to find pre-images for graphs [5]. The pre-image problem for graphs is very important because it has potential application to drug design [5] by using a suitable objective function reflecting desired properties. Several studies have also been done for designing molecules with optimal values using heuristic methods [14, 17] though kernel methods were not used there.

In a previous work [1], we studied a theoretical aspect of the pre-image problem. Precisely, we studied the problem of inferring a graph from a feature vector consisting of the numbers of occurrences of vertex-labeled paths. It should be noted *marginalized graph kernels* are based on similar types of feature vectors and have been already applied to classification of chemical compounds [10, 12]. We showed that this inference problem can be solved in polynomial time in the size of an output graph if graphs are trees of bounded degree and the lengths of given paths are bounded by a constant, whereas this problem is strongly NP-hard even for planar graphs of bounded degree [1].

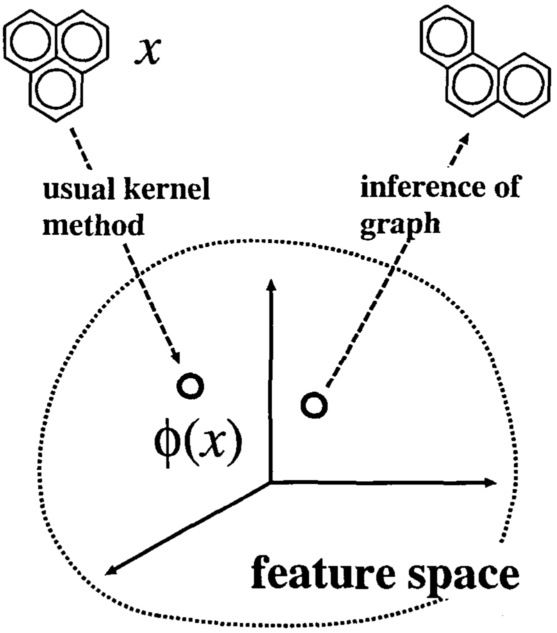


Figure 1: Inference of a graph from a feature vector

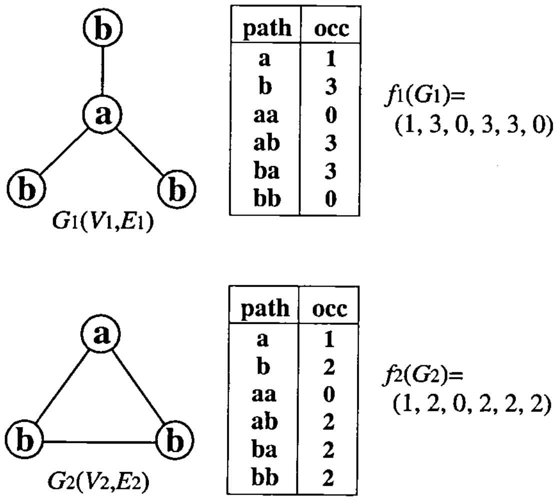
In this paper, we show further results on inference of a graph from path frequency. We show that the inference problem can be solved in polynomial time for *outerplanar graphs* with some reasonable restrictions. This result is important from a chemical viewpoint because many chemical compounds have outerplanar graph structures. We also show that the problem is strongly NP-hard even for trees of unbounded degree. Though these results are still theoretical, we expect that these are an important step towards development of practical methods for designing new chemical compounds using kernel methods.

2 PROBLEM

Here, we review the definition of the problem of inferring a graph from path frequency [1]. Let $G(V,E)$ be an undirected vertex-labeled graph and Σ be a set of vertex labels. A sequence of vertices $(v_0, v_1, \ldots, v_h)$ of G is called a *path* of length h ($h \geq 0$) if $\{v_i, v_{i+1}\} \in E$ holds for $i = 0, \ldots, h-1$. It should be noted that the same vertex and the same edge can appear more than once in this definition. Since most papers on marginalized graph kernels [10, 12, 16] use this notation for a path, we employ this definition. Let $\Sigma^{\leq k}$ be the set of label sequences (i.e., the set of strings) over Σ whose lengths are between 1 and k. Let $l(v)$ be the label of vertex v. For a path $P = (v_0, \ldots, v_h)$ of G, $l(P)$ denotes the label sequence of P (i.e., $l(P) = l(v_0)l(v_1)\ldots l(v_h)$). It should be noted that the length of $l(P)$ is the length of P plus one. For graph G and label sequence t, $occ(t,G)$ denotes the number of paths P in G such that $l(P) = t$. Then, the feature vector $\mathbf{f}_K(G)$ of level K for $G(V,E)$ is an integer vector such that the coordinate indexed by $t \in \Sigma^{\leq K+1}$ is $occ(t,G)$. That is, $\mathbf{f}_K(G)$ is defined by

$$\mathbf{f}_K(G) = (occ(t,G))_{t \in \Sigma^{\leq K+1}}.$$

For example, consider a star $G_1(V_1,E_1)$ consisting of four vertices where the center vertex has label 'a' and the other three vertices have label 'b'. Then, $\mathbf{f}_1(G_1) = (1,3,0,3,3,0)$ because $occ(a,G_1) = 1$, $occ(b,G_1) = 3$, $occ(aa,G_1) = 0$, $occ(ab,G_1) = 3$, $occ(ba,G_1) = 3$ and $occ(bb,G_1) = 0$. Another example is also given in Fig. 2.



path	occ
a	1
b	3
aa	0
ab	3
ba	3
bb	0

$f_1(G_1)=$ $(1,3,0,3,3,0)$

path	occ
a	1
b	2
aa	0
ab	2
ba	2
bb	2

$f_2(G_2)=$ $(1,2,0,2,2,2)$

Figure 2: Examples of feature vectors for graphs

In this paper, we assume for simplicity that *tottering paths* (paths for which there exists some i such that $v_i = v_{i+2}$) are not counted in feature vectors because removal of tottering paths does not decrease the prediction accuracy [12]. However, all the results on graphs in this paper are also valid even if tottering paths are not removed.

It should be noted that there exist the following cases: (i) there may not exist a graph corresponding to the specified feature vector, (ii) different graphs are mapped into the same feature vector. Therefore, we defined the graph inference problem as follows [1].

Graph Inference from Path Frequency (GIPF) Given a feature vector $\mathbf{v}$ of level K, output a graph $G(V,E)$ satisfying $\mathbf{f}_K(G) = \mathbf{v}$. If there does not exist such $G(V,E)$, output "no solution".

For the case of "no solution", we can consider the problem **(GIPF-M)** of finding $G(V,E)$ which minimizes the L_1 distance between $\mathbf{v}$ and $\mathbf{f}_K(G)$ (see also Fig. 3) [1]. Though we mainly show results for GIPF here, similar results hold for GIPF-M.

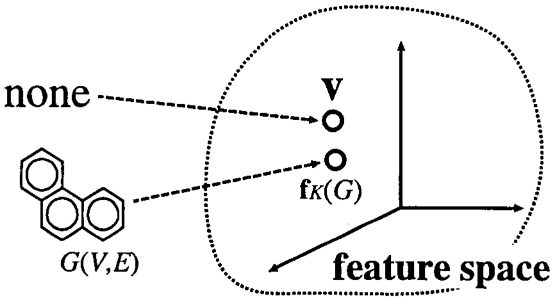


Figure 3: If there does not exist $G(V,E)$ satisfying $\mathbf{f}_K(G) = \mathbf{v}$, "no solution" is output in GIPF, whereas $G(V,E)$ which minimizes L_1 distance between $\mathbf{f}_K(G)$ and $\mathbf{v}$ is output in GIPF-M

It is worthy to note here that related graph theoretic problems have been studied, which include graphical degree sequence problems [2], graph inference from walks [13] and the graph reconstruction problem [11]. However, GIPF and GIPF-M are considerably different from these and thus the results in these problems can not be directly applied.

3 ALGORITHM FOR OUTERPLANAR GRAPHS

In this section, we present a polynomial time algorithm for GIPF for outerplanar graphs. Before presenting it, we briefly review our previous results on trees [1].

Theorem 1 [1] *GIPF for trees is solved in polynomial time in n for $K = 1$ and fixed alphabet.*

We briefly describe the algorithm used in the proof [1]. For simplicity, we consider the case of binary alphabet $\Sigma = \{0,1\}$. We construct the table $D(\ldots)$ defined by

$$D(n_0,n_1,n_{00},n_{01},n_{10},n_{11}) = \begin{cases} 1, & \text{if there exists tree } T \text{ such that} \\ & \mathbf{f}_1(T) = (n_0,n_1,n_{00},n_{01},n_{10},n_{11}), \\ 0, & \text{otherwise.} \end{cases}$$

This table can be constructed by the following dynamic programming procedure.

$$\begin{array}{l} D(n_0,n_1,n_{00},n_{01},n_{10},n_{11}) = 1 \textbf{ iff.} \\ \quad D(n_0-1,n_1,n_{00}-2,n_{01},n_{10},n_{11}) = 1 \quad \textbf{or} \\ \quad D(n_0-1,n_1,n_{00},n_{01}-1,n_{10}-1,n_{11}) = 1 \quad \textbf{or} \\ \quad D(n_0,n_1-1,n_{00},n_{01}-1,n_{10}-1,n_{11}) = 1 \quad \textbf{or} \\ \quad D(n_0,n_1-1,n_{00},n_{01},n_{10},n_{11}-2) = 1. \end{array}$$

The correctness of the algorithm follows from the fact that any tree can be constructed incrementally by adding a vertex (leaf) one by one. The required tree (if exists) can be obtained by using the *traceback* technique.

In [1], we extended the above result for more general trees. The core part of the extension is the definition of the dynamic programming table, which is explained below.

Though we only consider undirected trees, we will treat an undirected tree as if it were a rooted tree. Let r be the root of a tree T. The *depth* (denoted by $d(v)$) of a vertex $v \in T$ is the length of the (shortest) path from r to v. The *depth of a tree* ($d(T)$) is the depth of the deepest vertex.

For each vertex $v \in T$, $T_K(v)$ denotes the subtree of T induced by the vertex set $\{v\} \cup \{w | w \text{ is a descendant of } v, |P(v,w)| \leq K\}$, where $P(v,w)$ denotes the (shortest) path from v to w.

$ID(v)$ denotes the signature (i.e., canonical labeling in [8]) of v where the signature is an integer number of value $O(n)$ such that $ID(v) = ID(v')$ iff. $T_K(v)$ is isomorphic to $T'_K(v')$. Since we consider constant K and trees of bounded degree, $ID(v)$ can be computed in $O(1)$ time for each v.

Each vertex v maintains the set of paths which contain v as the shallowest vertex. It should be noted that each vertex needs to maintain $O(1)$ paths since we consider constant K and trees of bounded degree. It should also be noted that each path is maintained by exactly one vertex.

For each tree T, we associate a table $E(d,id)$ where $E(d,id)$ denotes the number of vertices v such that $d(v) = d$ and $ID(v) = id$. Since there are $O(1)$ different signatures, $E(d,id)$ consists of $O(d(T))$ elements.

Let **e** denotes the vector consisting of $E(d,id)$ for $d = d(T), d(T)-1, d(T)-2, \ldots, d(T)-K$. Let $\mathbf{g}_K(T)$ denotes **e** for T. It should be noted that the number of dimensions of **e** is bounded by a constant.

Then, we construct table $D(\mathbf{v},\mathbf{e},d)$ defined by: $D(\mathbf{v},\mathbf{e},d) = 1$ iff. there exists a tree T such that $\mathbf{f}_K(T) = \mathbf{v}$, $\mathbf{g}_K(T) = \mathbf{e}$ and $d(T) = d$. Construction of the table is done in an incremental manner as in the case of $K = 1$. We only add a new vertex at depth either d or $d+1$. It should be noted that any tree can be constructed in this manner. Based on this table, we obtained the following theorem.

Theorem 2 [1] *GIPF for trees of bounded degree is solved in polynomial time in n if K and Σ are fixed.*

Now, we show a new algorithm for outerplanar graphs. A graph is called *planar* if it can be drawn in the plane such that no two edges cross. A graph is called *outerplanar* if it is planar and all vertices lie on the outer face of the drawing. It is well-known that an outerplanar graph can be represented by a tree [3, 15], where each face of the outerplanar graph corresponds to a vertex in a tree (see Fig. 4). Though a tree may not be determined uniquely (as shown in Fig. 5), the uniqueness is not required here. We denote a tree representation of graph G by $tr(G)$.

As in the case of GIPF for trees, we construct the dynamic programming table $D(\mathbf{v},\mathbf{e},d)$. However, in this case, the meanings of **e** and d are different from those for trees. The differences (compared with GIPF for trees) are summarized as follows.

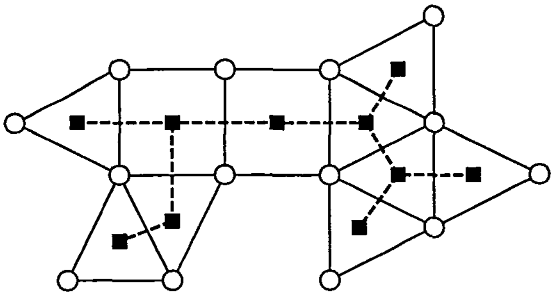


Figure 4: Outerplanar graph and its tree representation, where white circles and black boxes correspond to vertices of the original graph and the corresponding tree, respectively

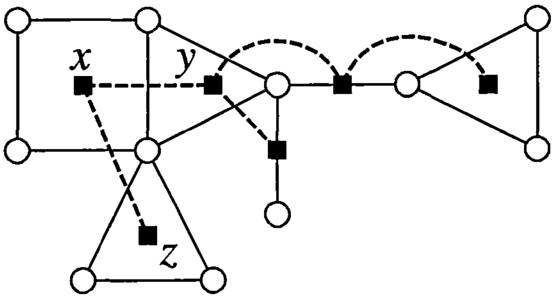


Figure 5: Example where a corresponding tree is not uniquely determined. In this figure, vertex z is connected to x. But, z can be connected to y

- d denotes the depth of $tr(G)$,
- **e** denotes the vector consisting of the numbers of identical subtrees of $tr(G)$, where information about the corresponding face should be attached to each vertex in a subtree,
- Subtrees of $tr(G)$ should be ordered ones (because edges in a face has some ordering),
- Vertices (leaves) of $tr(G)$ is added one by one .

In order to develop a polynomial time algorithm, the number of dimensions of **e** should be bounded by a constant. That is, the number of possible subtrees of depth at most K should be bounded by a constant. For that purpose, we assume that K, the size of an alphabet, the maximum degree of vertices in G, and the maximum number of edges of each face are bounded by a constant. Then, we have the following theorem.

Theorem 3 *GIPF for outerplanar graphs of bounded degree is solved in polynomial time in the size of an output graph if K and Σ are fixed and the number of edges of each face is bounded by a constant.*

As in [1], we can extend the result for GIPF-M.

Corollary 1 *GIPF-M for outerplanar graphs of bounded degree is solved in polynomial time in the size of an output graph if K and Σ are fixed and the number of edges of each face is bounded by a constant.*

4 HARDNESS RESULTS FOR TREES

Here, we show that GIPF is strongly NP-hard even for trees. For that purpose, we modify the proof given in [1] where an NP-hardness result was shown for non-tree graphs.

In the proof for non-tree graphs [1], we used a pseudo polynomial time transformation from 3-PARTITION [9]. 3-PARTITION is known to be strongly NP-complete, and is defined as follows: given a set X which consists of $3m$ elements x_i along with their integer weights $w(x_i)$ and a positive integer B where each $w(x_i)$ satisfies $B/4 < w(x_i) < B/2$, find a partition of X into $A_1, A_2, \ldots, A_m$ such that each A_i consists of 3 elements and $\sum_{x_j \in A_i} w(x_j) = B$ holds for each A_i.

In the transformation, a feature vector of level 4 (i.e., $K = 4$) is constructed from subgraphs of the target graph $G(V,E)$, where $G(V,E)$ corresponds to a solution to 3-PARTITION. The target graph has the form shown in Fig. 6 and is constructed as below.

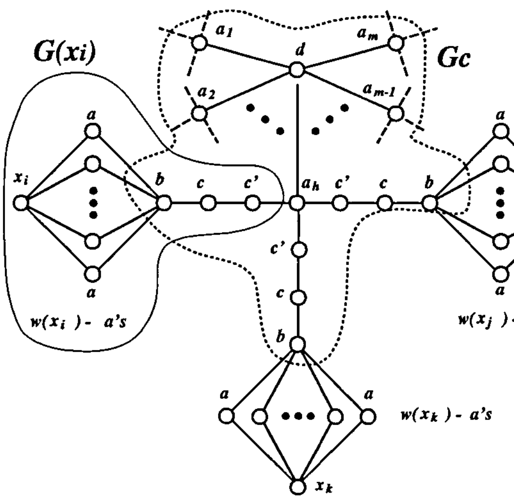


Figure 6: Reduction from 3-PARTITION to GIPF [1], where a_h corresponds to set $A_h = \{x_i, x_j, x_k\}$

We let $\Sigma = X \cup \{a_i | i = 1, \ldots, m\} \cup \{a, b, c, c', d\}$. We identify a vertex with its label if the vertex with the same label appears only once in a graph. For each x_i, we construct a subgraph (called a *block*) $G(x_i)$ shown in Fig. 6. Note that there are $w(x_i)$ vertices with label a in $G(x_i)$, and three blocks will be connected to the same vertex labeled a_h though it is not explicitly specified by the feature vector which blocks are connected to the same vertex. We connect vertex d to m vertices with labels a_h's as in Fig. 6, where three paths of the form c'-c-b are also connected to each a_h. The subgraph consisting of vertices with labels d, a_h's, b, c and c' is called the *center graph* and is denoted by G_c.

The feature vector $\mathbf{v}$ is constructed from the following paths:

PATHS-A: all paths at most length 4 in each block or in the center graph,

PATHS-B: for each a_h, we construct B paths of the form of a_h-c'-c-b-a and the corresponding B paths in the reverse direction.

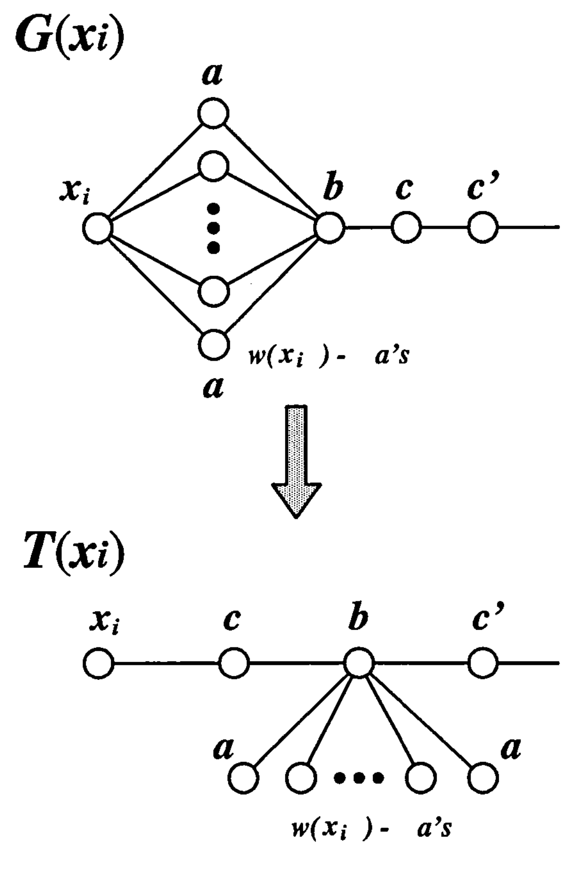


Figure 7: Modification of a subgraph for showing NP-hardness for trees

Then, it was shown in [1] that there exists a graph $G(V,E)$ such that $\mathbf{f}_K(G) = \mathbf{v}$ if and only if there exists a solution for 3-PARTITION.

Now, we modify the reduction for trees of unbounded degree. Modification is very simple. We only need to replace $G(x_i)$ with $T(x_i)$ as shown in Fig. 7, and $K = 4$ with $K = 3$. Then, we can prove the following in an analogous way as in [1].

Theorem 4 *GIPF is strongly NP-hard even for* $K = 3$ *and trees of unbounded degree.*

It is worthy to mention that the degree and the size of Σ are not bounded and thus this result does not contradict with the results in the previous section. We can further modify the NP-hardness result for trees of bounded degree and of a fixed Σ though K can not be any more bounded by a constant (K is $O(\log|V|)$). The modification can be done as in [1] by using binary tree-like substructures for representing high degree vertices and binary encoding for representing labels of vertices.

Theorem 5 *GIPF is strongly NP-hard even for trees of bounded degree and of a fixed* Σ.

5 CONCLUDING REMARKS

We have shown that inference of a graph from path frequency can be solved in polynomial time in the size of an output graph if graphs are restricted to be outerplanar graphs of bounded degree, K and Σ are fixed, and the number of edges of each face is bounded by a constant. We have also shown that the inference problem is strongly NP-hard even for trees of unbounded degree. These results shorten the theoretical gaps between the positive and negative results in our previous work [1]. However, there still remain large gaps. For example, the complexity (polynomial or NP-hard) of the following cases should be studied:

- inference of general graphs of bounded degree from paths with constant K,
- inference of trees from paths with large K (e.g., K is $O(|V|)$).

Development of approximation algorithms for NP-hard cases of GIPF-M is also interesting future work.

Though we shorten the theoretical gaps, the proposed algorithms are still not useful in practice because constant factors depending on K and Σ are quite large. Therefore, faster and practical algorithms should be developed. The class of outerplanar graphs in this paper covers a large class of chemical compounds. However, it is known that some chemical compounds are not planar. Thus, it is important to extend the class of graphs for which GIPF can be solved in polynomial time. In this paper, we used feature vectors defined by path frequency. However, probabilities of paths are used in more practical kernels [10, 12]. Therefore, inference of a graph from a feature vector defined by probabilities of paths should also be studied.

ACKNOWLEDGMENTS

This work was supported in part by Grants-in-Aid #17017019 and #16300092 from the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan, and by the Ohkawa Foundation for Information and Telecommunications.

REFERENCES

[1] T. Akutsu and D. Fukagawa, Inferring a graph from path frequency. In Proc. 16th Symp. Combinatorial Pattern Matching (LNCS No. 3537), pages 371–382, Jeju, Korea, 2005.

[2] T. Asano. An $O(n \log \log n)$ time algorithm for constructing a graph of maximum connectivity with prescribed degrees. J. Computer and System Sciences, 51:503–510, 1995.

[3] B.S. Baker. Approximation algorithms for NP-complete problems on planar graphs. Journal of ACM, 153–180, 1994.

[4] G.H. Bakir, J. Weston, and B. Schölkopf. Learning to find pre-images. Advances in Neural Information Processing Systems, 16:449-456, 2004.

[5] G.H. Bakir, A. Zien and K. Tsuda, Learning to find graph pre-images, In Proc. the 26th DAGM Symposium (LNCS No. 3175), pages 253–261, 2004.

[6] C. Cortes and V. Vapnik. Support vector networks. Machine Learning, 20:273–297, 1995.

[7] Nello Cristianini and Jhon Shawe-Taylor. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge Univ. Press, Cambridge, UK, 2000.

[8] Y. Dinitz, A. Itai, and M. Rodeh. On an algorithm of Zemlyachenko for subtree isomorphism. Information Processing Letters, 70:141–146, 1999.

[9] Michael R. Garey and David S. Johnson. Computers and Intractability. A Guide to the Theory of NP-Completeness. W.H. Freeman and Co., New York, 1979.

[10] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In Proc. 20th Int. Conf. Machine Learning, pages 321–328, 2003.

[11] Josef Lauri and Raffaele Scapellato. Topics in Graph Automorphisms and Reconstruction. Cambridge Univ. Press, Cambridge, UK, 2003.

[12] P. Mahé, N. Ueda, T. Akutsu, J-L. Perret, and J-P. Vert. Extensions of marginalized graph kernels. In Proc. 21st Int. Conf. Machine Learning, pages 552–559, 2004.

[13] O. Maruyama and S. Miyano. Inferring a tree from walks. Theoretical Computer Science, 161:289–300, 1996.

[14] R.B. Nachbar. Molecular evolution: automated manipulation of hierarchical chemical topology and its application to average molecular structures. Genetic Programming and Evolvable Machines, 1:57–94, 2000.

[15] Takao Nishizeki and Norishige Chiba. Planar Graphs: Theory and Algorithms. North-Holland, Amsterdam, 1988.

[16] Bernhard Schölkopf, Kouji Tsuda, and Jean-Philippe Vert (eds.). Kernel Methods in Computational Biology. The MIT Press, Cambridge, MA, 2004

[17] H.M. Vinkers, M.R. de Jonge, F.F.D. Daeyaert, J. Heeres, L.M.H. Koymans, J.H. van Lenthe, P.J. Lewi, H. Timmerman, K. van Aken, and P.A.J. Janssen. Synopsis: synthesize and optimize system in silico. Journal of Medical Chemistry, 46:2765–2773, 2003.