**RC23566 (W0503-075) March 11, 2005**
**Computer Science**

# Discovering Topological Motifs Using a Compact Notation

**Laxmi Parida**
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

# Discovering Topological Motifs Using a Compact Notation

Laxmi Parida[1]

Computational Biology Center, IBM T. J. Watson Research Center
Yorktown Heights, NY10598, USA
parida@us.ibm.com

**Abstract.** Discovering topological motifs or common topologies in one or more graphs is an important and an interesting problem. It had been classically viewed as the subgraph isomorphism problem. This problem and its various flavors are known to be NP-Complete. However, this does not minimize the importance of solving this problem accurately in application areas such as bioinformatics or even large network studies. The explosion in the output is usually caused by isomorphisms in the motif or graph: we present a method to handle this without sacrificing the correct answers. In this paper, we apply the natural notion of maximality, used extensively in strings, to the graphs and present a simple three-step approach to solving this problem completely and accurately (without resorting to heuristics). We handle the natural combinatorial explosion due to isomorphism inherent in the problem by the use of "compact location lists". In other words, instead of enumerating $k$ elements out of $n$, we use the $\binom{n}{k}$ form in an implicit manner, this drastically reduces the size of the output without any loss of information. The algorithm we present is linear in terms of the size of the output encoded as compact lists.

**Keywords:** Subgraph isomorphism, graph isomorphism, topological motifs, pattern discovery, motif discovery, data mining

## 1 Introduction

Understanding large volumes of data is a key problem in a large number areas such as the world wide web, bioinformatics and so on. Some of data in these areas cannot be represented as linear strings which have been studied extensively with a repertoire of sophisticated and efficient algorithms. The inherent structure in these data sets is best represented as graphs. This is particularly important in areas such as bioinformatics or chemistry since it might lead to understanding of biological systems from indirect evidences in the data. Thus automated discovery of "phenomenon" is a promising path to take as is evidenced by the use of motif (substring) discovery in DNA and protein sequences.

A protein network is a graph that encodes primarily protein-protein interactions and this is important in understanding the *computations* that happen within a cell [HETC00, SBH$^{+}$01]. A recurring topology or motif in such a setting has been interpreted to be to act as robust filters in the transcriptional network of *Escherichia coli* [MSOI$^{+}$02, SOMMA02]. It has been observed that the conservation of proteins in distinct topological motifs correlates with the interconnectedness and function of that motif and also depends on the structure of the topology of all the interactions indicating that motifs may represent evolutionary conserved topological units of cellular networks in accordance with specific biological functions they perform [SOB03, LMF03]. This observation is strikingly similar to the hypothesis in dealing with DNA and protein primary structures.

Topological motifs are also being studied in the context of structural units in RNA [HPS03] and for structural multiple alignments of proteins [DBNW03]. For yet another application consider a typical chemical data set: a chemical is modelled as a graph with attributes on the vertices and the edges. A vertex represents an atom and the attribute encodes the atom type; an edge models the bond between the atoms it connects and its attribute encodes the bond type. In such a database, very frequent common topologies could suggest the relationship to the characteristic of the database. For instance, in a toxicology related database, the common topologies may indicate carcinogenicity or any other toxicity.

In the field of machine learning, methods have been proposed to search for subgraph patterns which are considered characteristic and appear frequently: this uses an a priori-based algorithm with generalizations from association discovery [IWM03]. In massive data mining where the data is extremely large: it is of the order of tens of gigabytes. These include the world wide web, internet traffic and telephone call detail. The common topologies here are used to discover social networks and web communities among other characteristics [Mur03].

In biological data the size of the database is not as large, yet unsuitable for enumeration schemes. When this scheme was applied researchers had to restrict their motifs to small sizes such as three or four [MSOI$^{+}$02]. The problem of finding common trees in a forest is discussed in [Zak02], which is a special case of the general graph problem discussed below. We take a combinatorial approach to the problem and introduce a compact notation to handle the combinatorial explosion arising from isomorphisms. The problem is abstracted as follows: Given a graph $G(V, E)$ with labelled vertices and edges, the task is to discover at least $k(> 1)$ subgraphs that are topologically identical in $G$. Such subgraphs are termed topological motifs. It is very closely related to the the classical *subgraph isomorphism* problem defined as follows [GJ79]: Given graphs $G = (V_1, E_1)$ and $H = (V_2, E_2)$. Does $G$ contain a subgraph isomorphism to $H$ i.e., a subset $V \subseteq V_1$ and a subset $E \subseteq E_1$ such that $|V| = |V_2|$, $|E| = |E_2|$ and there exists a one-to-one function $f : V_2 \rightarrow V$ satisfying $\{v_1, v_2\} \in E_2$ if and only if $\{f(v_1), f(v_2)\} \in E$? Two closely related problems are as follows [GJ79]. (1) *Largest common subgraph problem*: Given graphs $G = (V_1, E_1)$ and $H = (V_2, E_2)$, positive integer $K$. Do there exist subsets $E_1' \subseteq E_1$ and $E_2' \subseteq E_2$ with $|E_1'| = |E_2'| \geq K$ such that the two subgraphs $G' = (V_1, E_1')$ and $H' = (V_2, E_2')$ are isomorphic? (2) *Maximum subgraph matching problem*: Given directed graphs $G = (V_1, E_1)$ and $H = (V_2, E_2)$, positive integer $K$. Is there a subset $R \subseteq V_1 \times V_2$

with $|R| \geq K$ such that for all $< u, u' >, < v, v' > \in R, (u, v) \in A$, if and only if $(u', v') \in A_2$? All the three problems are NP-Complete: each can be transformed from the CLIQUE problem. The problem addressed in this paper is similar to the latter two problems. However our interest is in finding at least $K$ isomorphs and all possible such isomorphs.

A naive enumeration to discover all the topological motifs is stymied primarily by the combinatorial explosion. When a common topology represented by $G'$ occurs at least $k$ times in a graph, so do all the subgraphs of $G'$. Also, when a vertex $v$ has $n$ identical neighbors and only $k$ need to be chosen then there are $\binom{n}{k}$ ways of choosing $v$'s neighbors leading to a combinatorial explosion: this is a result of self-isomorphism in the subgraphs.

A discovery process must handle these issues appropriately to make the task of topological motif detection a practical and an useful process. At the same time it is vital that there is no loss of information. This requires some natural restrictions like maximality and we extend the ideas from one-dimensional strings [Par00, AP04] to our problem definition in the following discussion in Section 2. The compact lists handle the self-isomorphism related issues and in Section 3 we present an efficient algorithm to compute the motifs in time proportional to the size of the results represented in the compact form.

## 2 The Discovery Problem

In the remainder of the discussion we deal with undirected graphs with vertex attributes. In other words all the edges have the same attribute. We show that any general graph, directed or undirected and with arbitrary attributes defined on the vertices and the edges, can be mapped onto such a special case and the results from this can be mapped back to the original general graph. This mapping is straightforward and we demonstrate this in the Appendix.

Consider a graph $G(V, E)$ with $|V| = n$. Let $\mathcal{A}_V$ be a finite set of elements (referred to as attributes) with $A : v \to \mathcal{A}_V$, a mapping sending $v \mapsto a_v$.

**Definition 1.** A topological motif is a connected graph $M(V_m, E_m)$ where $V_m = \{u_0, u_1, u_2, \ldots, u_p\}$, $p > 1$ and is said to **occur** on the set of vertices $O_i = \{v_{i0}, v_{i1}, v_{i2}, \ldots, v_{ip}\} \subseteq V$ of graph $G$, $1 \leq i \leq K$, if and only if (1) for each $i, 1 \leq i \leq K$, there is a one-to-one onto mapping $F_i : V_m \to O_i$, such that, $A(u_j) = A(F_i(u_j))$ (inverse is denoted by $F_i^{-1}()$), and (2) $(u_{j_1}, u_{j_2}) \in E_m \Rightarrow (F_i(u_{j_1}), F_i(u_{j_2})) \in E$ holds. If there is no $O$ distinct from each of $O_i$ such that motif $M$ occurs on $O$, then $O_1, O_2, \ldots O_k$ is a complete occurrence list. The ordered set of vertices $O_i$'s are aligned if for each $j$, $F_i^{-1}(v_{ij}) = v_j \in V_m$, for all $i$.

A topological motif must be connected, hence $|E_m| > 0$. Notice that a singleton vertex is not considered a topological motif by the definition: this excludes the trivial motifs of a vertex with a fixed attribute.

For example, consider the graph shown in Figure 1(a). The occurrence of a topological motif $M$ at $O_1, O_2, \ldots O_{10}$ is shown in Figure 1. Here the motif shown in Figure 1(b) is $M(V_m, E_m)$ where $V_m = \{u1, u2, u3, u4\}$. For the occurrence (numbered 1) shown in Figure 1(1), aligned $O_1 = \{v4, v5, v9, v10\}$. Further $F_1(u1) = v4$, $F_1(u2) = v5$, $F_1(u3) = v9$ and $F_1(u4) = v10$ and $A(u1) = A(v4) = \Box$ and so on.

**Definition 2.** Given a graph $G(V, E)$, the vertices in $V_1 \subseteq V$ are isomorphic with respect to $V_2 \subseteq V$ if $A(v \in V_1) = a_1$ and $A(v \in V_2) = a_2$ for some attributes $a_1$ and $a_2$ and there is an edge $v_i v_j \in E$, for each $v_i \in V_1$ and $V_j \in V_2$. If there exists no $V_1' \supset V_1$ and $V_2' \supset V_2$ such that $V_1'$ and $V_2'$ are isomorphic, then $V_1$ is maximal with respect to $V_2$. [To avoid clutter, $V_1$ will be called a set of (maximal) isomorphic vertices and existence of some non-empty $V_2$ will be assumed to be implicit]

In Figure 1(2), $U_1 = \{u1, u2\}$ is (maximally) isomorphic with respect to $U_2 = \{u3, u4\}$ and vice-versa. In the rest of the paper, a set such as $U_1$ (and also $U_2$) will be called a maximal set of isomorphic vertices. **Maximality**: We next define a maximal motif, which is a natural extension of the maximality on linear strings [Par00].

**Definition 3.** Let $M(V_m, E_m)$ be a topological motif with its complete occurrence list $O_1, O_2, \ldots, O_l$. The motif is maximal if both edge-maximality and vertex-maximality hold. *Edge-maximal:* For all $v_{j_1}, v_{j_2} \in V_m$ if $(F_i(v_{j_1}, F_i(v_{j_2})) \in E$ for all $i$, $1 \leq i \leq l$, then $(v_{j_1}, v_{j_2}) \in E_m$. *Vertex-maximal:* There does not exist vertices $v_1, v_1', v_2, v_2', \ldots, v_l, v_l' \in V$ such that the following hold: (1) $v_i \notin O_i$, $v_i' \in O_i$, $1 \leq i \leq l$ (2) $A(v_i) = A(v_j)$, $1 \leq i, j \leq l$ (3) $F_i^{-1}(v_i') = F_j^{-1}(v_j')$, $1 \leq i, j \leq l$

In other words, edge-maximality ensures that no more edges can be added to the motif and vertex-maximality ensures that no more vertices can be added to the motif. Notice that Figure 1(b) is not a maximal motif since at least another vertex can be added to the motif and Figure 2 gives an example of a maximal motif.

**Definition 4.** Let $O_1, O_2, \ldots O_K \subseteq V$ be a complete and aligned occurrence list of topological motif $M(V_m, E_m)$, and let $U = \{u_1, \ldots, u_d\} \subseteq V_m$ be a set of maximal isomorphic vertices of $M(V_m, E_m)$. The location list of $U$ is denoted as $U\text{-}\mathcal{L}_m = \{\{F_i(u_1), \ldots, F_i(u_d)\} 1 \leq i \leq K\}$. $M$ has a quorum $K$, if there exists at least one set of maximal isomorphic vertices $U$ with $|U\text{-}\mathcal{L}_m| \geq K$.

Thus the occurrence list gives the entire footprint of the motif at each occurrence and the location list tracks the mapping of each vertex (or a set of isomorphic vertices) in the motif. Also notice that by definition the location list is *complete* since it corresponds to a complete occurrence list. In Figure 1, the (complete) occurrence list of the two isomorphic vertices $U = \{u1, u2\}$ are given as $U\text{-}\mathcal{L}_m = \{\{v4, v5\}, \{v1, v2\}, \{v2, v3\}, \{v1, v3\}\}$.
**Compact list**: To ease handling the combinatorial explosion due to isomorphisms, we introduce the compact list notation for the location lists of topological motifs. It is often possible to represent $U\text{-}\mathcal{L}_m$ (Definition 4) in a much

more compact way taking into account the fact the the vertices in $U$ are isomorphic. For instance, in the example in Figure 1(b), $U = \{u_1, u_2\}$ is set of isomorphic vertices and $U\text{-}\mathcal{L}_m = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}, \{v_4, v_5\}\}$. However, a more compact way to represent $U\text{-}\mathcal{L}_m$ is by the set, $U\text{-}\mathcal{L}^c_m = \{\{v_1, v_2, v_3\},\{v_4, v_5\}\}$ and one recovers $U\text{-}\mathcal{L}_m$ from $U\text{-}\mathcal{L}^c_m$ by taking all two (the smallest cardinality of the sets in $U\text{-}\mathcal{L}_m$) element subsets of the sets in $U\text{-}\mathcal{L}^c_m$. We call $U\text{-}\mathcal{L}_m$ the expansion of the set $U\text{-}\mathcal{L}^c_m$, and $U\text{-}\mathcal{L}^c_m$ a compact form of $U\text{-}\mathcal{L}_m$. More precisely, given a set $\mathcal{L} = \{L_1, \ldots, L_\ell\} \subset 2^V$, we define the expansion of $\mathcal{L}$, $E(\mathcal{L}) \subset 2^V$ as follows:

**Definition 5.** Let $d = \min_{i=1}^{\ell} |L_i|$. If $|L' \in \mathcal{L}| = d$ then $L'$ is called a *discriminant* of $\mathcal{L}$. $Flat(\mathcal{L}) = \cup_i L_i \in \mathcal{L}$ and $|\mathcal{L}| = |E(\mathcal{L})|$ where the expansion $E(\mathcal{L})$ is given as

$$E(\mathcal{L}) = \{L \in 2^V | \exists L_i \in \mathcal{L} \mid L \subset L_i, |L| = d\}.$$

## 3 The Discovery Algorithm

### 3.1 Main Idea

The method is based on a simple observation that given a graph a topological motif can be represented either as a motif (graph) or as a collection of location lists of the vertices of the motif. Our approach to discover the multiply occurring motifs is to work in the space of the location lists. There are two aspects that lend themselves to efficient discovery: (1) The motifs are maximal so a relatively mall number of possibilities are to be explored. For instance if a graph has exactly three red vertices with 5, 8 and 10 blue immediate neighbors (along with other edges and vertices with other colors), then any maximal motif with a red vertex can have exactly 5 blue neighbors or exactly 8 blue neighbors, although a subgraph could have from 0 to 10 blue neighbors. The compact location list captures this succinctly. (2) A single conservative "intersection"-like operation (eg conjunct in Step 1 and join in Step 2) can efficiently handle all the potential candidates with a high degree of efficiency.

**Method**. The discovery process consists of mainly two steps. In the first step we compute an exhaustive list of potential location lists of vertices of these motifs, which we store in a compact form, as compact location lists. In the second step we enlarge the collection of compact location lists computed in the first step by including all the non-empty intersections (termed *join* in the paper) amongst the location lists computed in the first step. The collection of compact location lists so obtained has the nice property that every maximal motif has a vertex (or a group of vertices) whose location list appear in compact form in this collection. Conversely, every compact location list appearing in this collection is the location list of some vertex (or isomorphism class of vertices) of some maximal motif. The intersection operations at different stages are carried out in an output-sensitive manner: this is possible since we are computing maximal intersections. This makes the overall algorithm very efficient. We could stop at this point, since often the location lists are all that is required. However, for the sake of completeness we compute in the third step of our algorithm an actual list of maximal topological motifs. In fact the previous steps store the neighborhood information of the location lists, thus the maximal motifs are obtained by traversing this neighborhood structure.

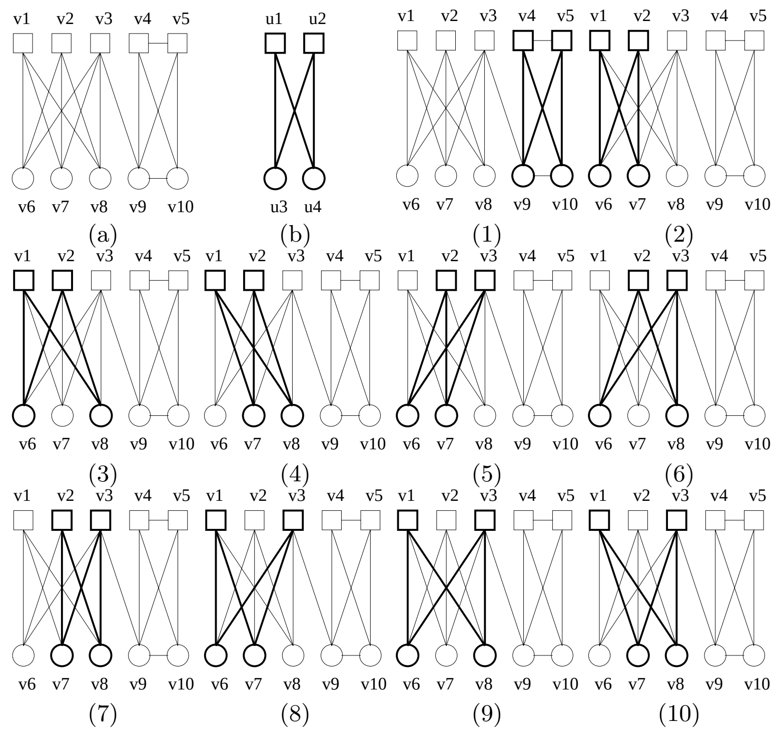

**Fig. 1.** (a) The input graph is given by $G$ with nodes numbered from $v1$ to $v10$. The nodes represented by squares have the attribute $s$, the circular nodes have attribute $r$. (b) A topological motif $M(V_m, E_m)$ on this graph: $u1$ and $u2$ are isomorphic and so are $u3$ and $u4$. (1)-(10) The ten occurrences of the topological motif are shown in bold on the original graph. Let $U_\square = \{u_1, u_2\}$, $U_\bigcirc = \{u_3, u_4\}$. All the 10 occurrences are represented in the following compact notation of the location lists corresponding to $U_\square$ and $U_\bigcirc$: $U_\square$-$\mathcal{L}_m = \{\{v4, v5\}, \{v1, v2, v3\}\}$ and $U_\bigcirc$-$\mathcal{L}_m = \{\{v9, v10\}, \{v6, v7, v8\}\}$.

**Input**: Given a graph $G(V, E)$ with at most $D$ distinct attributes associated with each vertex $v_i$. For the sake of convenience, let $B$ be a two-dimensional array of dimension $|V| \times D$ which encodes the graph as follows: $B[i][j]$ is the set of vertices adjacent to $v_i$ having the attribute $a_j$, $1 \leq i \leq |V|$, $1 \leq j \leq D$.

## 3.2 Step 1-Computing the Conjuncts

Let $L$ be the set of all possible complete compact lists $\mathcal{L} = \{L_1, \ldots, L_\ell\}$ with the following properties. (1) For each $v \in L_i, 1 \leq i \leq \ell$, $A(v) = a_c$ holds, and, (2)

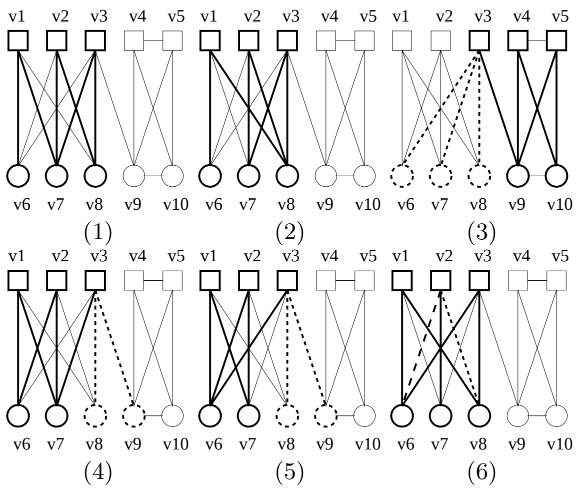

**Fig. 2.** The occurrences of a maximal topological motif (see (1) & (2) above) of the graph shown in Figure 1(a) are shown in bold. The dashed-bold indicate multiple occurrences shown in the same figure: the occurrence of a maximal motif is to be interpreted as all the solid vertices and edges and one of the dashed edges and the connected dashed vertex.

There exists at least one attribute $a_j$ such that there are $h > 0$ common neighbors $v_{ij}$ of all vertices $v \in L_i$ such that $A(v_{ij}) = a_j$, for all $i$. $\mathcal{L}$ is complete if there exists no $L \in 2^V$, $|L| \geq d$, $L \neq L_i, 1 \leq i \leq \ell$, satisfying properties 1 and 2 above. The *signature* $S(\mathcal{L}) = ((a_c, d), P)$ where $a_c$ is the attribute of property 1 and $P$ is the non-empty set of pairs $(a_j, h)$ that satisfy property 2 and $d$ is the size of the discriminant of $\mathcal{L}$. Given $\mathcal{L}$, for each attribute $a_j$ (of property 2), the *conjugate* compact location list is given as $\overline{\mathcal{L}}^j = \{L_k \in 2^V |\text{for each } v \in L_k, A(v) = a_j \text{ and } (uv) \in E, u \in L_i \in \mathcal{L}\}$. $L_k \in \overline{\mathcal{L}}^j$ is the conjugate of $L_i \in \mathcal{L}$ given as $c_j(L_i) = L_k$.

$\mathbf{L}_b \subseteq L$ is called the conjuncts and is defined as follows: $\mathcal{L} \in \mathbf{L}_b \Leftrightarrow \forall \mathcal{L}' \in L, (\mathcal{L}' \supset \mathcal{L}) \Rightarrow S(\mathcal{L}') \neq S(\mathcal{L})$. The output of Step 1 is $\mathbf{L} = \mathbf{L}_b \cup \mathbf{L}_c$ where $\mathbf{L}_c$ is the set of all conjugates of each $\mathcal{L} \in \mathbf{L}_b$. Observe that any compact location list $U$-$\mathcal{L}$, where $U$ has the usual meaning, is such that $U$-$\mathcal{L} \subseteq \mathcal{L}' \in \mathbf{L}$ for some $\mathcal{L}'$.

Due to space constraints, a running example is shown in the Appendix (Figure 4) for the referees. Given $\mathbf{C}$, $\mathcal{C}$, $\mathcal{P}$ where $\mathbf{C}$ is a subset of the location lists $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n$, $\mathcal{C}$ is a collection of vertices and $\mathcal{P}$ is a collection of attributes (indices), CREATE-SET($\mathbf{C}, \mathcal{C}, \mathcal{P}$) creates a data structure $\mathcal{D}$ so that searching can be done efficiently. Thus a query of the form if a subset $\mathcal{C} \in \mathcal{D}$ (EXIST($\mathbf{C}$) in the routine shown below) returns a True/False in time $O(\log |V|)$. In the routine below, $K$ is the quorum (see Definition 4).

```
PREPROCESS(K)
For each $j$, $1 \leq j \leq D$
  $\mathbf{C}_j \leftarrow \phi$
  For each $k$, $1 \leq k \leq D$
    For each $l$, $|B[i][k]|=l$ for some $i$
      $\mathcal{C}_j^{k(l)} = \{v_i \mid A(v_i)=a_j, |B[i][k]| \geq l\}$
      If $(|\mathcal{C}_j^{k(l)}| \geq K)$ $\mathbf{C}_j \leftarrow \mathbf{C}_j \cup \{\mathcal{C}_j^{k(l)}\}$

For each attribute $a_j$
   CONJUNCTS($\mathbf{C}_j, |V|, K$)
```

```
CONJUNCTS ($\mathbf{C}_j, h, K$)
{
    (1) If $(h \leq 0)$ Then Exit
    (2) Let $\mathbf{C}'_j = \{\mathcal{C}_j^{i(l)} \mid v_h \in \mathcal{C}_j^{i(l)} \in \mathbf{C}_j\}$
        Let $\mathcal{C}'_j = \{v_h\}$
        Let $\mathcal{P}'_j = \{(i,l) | \{\mathcal{C}_j^{i(l)}\} \in \mathbf{C}'_j\}$
    (3) If $((|\mathbf{C}'_j| \geq K)$ and not EXIST($\mathbf{C}'_j$))
          (4) CREATE-SET($\mathbf{C}'_j, \mathcal{C}'_j, \mathcal{P}'_j$)
          (5) CONJUNCTS($\mathbf{C}'_j, h-1, K$)
    (6) If EXIST($\mathbf{C}'_j$) $\mathcal{C}'_j \leftarrow \mathcal{C}'_j \cup \{v_h\}$
    (7) CONJUNCTS($\mathbf{C}_j, h-1, K$)
}
```

**Postprocessing**: New compact lists and their conjugates are generated in this step. The collection of compact lists $\mathbf{L}_b$ is constructed as follows: For each $\mathcal{C}'_j$, construct $\mathcal{L}_0^{jx}$ (where $Flat(\mathcal{L}_0^{jx}) = \mathcal{C}'_j$ and $x$ is an indexing number) as follows. If $v_{i_1}, v_{i_2} \in \mathcal{C}'_j$ and for each $(k, -) \in \mathcal{P}'$ $B[i_1][k] \subseteq B[i_2][k]$ without loss of generality then $v_{i_1}, v_{i_2} \in L_i \in \mathcal{L}_0^{jx}$. Notice that for each $v \in Flat(\mathcal{L}_0^{jx})$, $A(v) = a_j$.

We next introduce a process of *maximalization* of the compact lists. This is best explained through a simple example. Consider a compact list $\mathcal{L}' = \{ \{v_1, v_2\}, \{v_3, v_4, v_5\}, \{v_6, v_7, v_8, v_9\}\}$ where each vertex has the attribute say $a_1$. Then the corresponding maximal motif has at least two vertices $U = \{u_1, u_2\}$ with attribute $a_1$ and $\mathcal{L}' = U$-$\mathcal{L}$. However by our definition of maximal motifs, there will also exist a maximal motif with at least three vertices $U' = \{u_1, u_2, u_3\}$ with $U'$-$\mathcal{L} = \{\{v_3, v_4, v_5\}, \{v_6, v_7, v_8, v_9\}\}$. This can always be implictly handled but for clarity in the presentation of the algorithm, we will explicitly compute and store $U'$-$\mathcal{L}$ as described below.

For each $\mathcal{L}$, MAXIMALIZE($\mathcal{L}$) as follows: Let $d$ be the size of the discriminant of $\mathcal{L}$ and let $m = \max_i(|L_i \in \mathcal{L}|)$. Then for each p, $d < p \leq m$, generate $\mathcal{L}_p = \{L_i \in \mathcal{L} \mid |L_i| \geq p\}$. Update $\mathbf{L}_b$ with the new locations lists generated in MAXIMALIZE($\mathcal{L}$).

Conjugate location lists of $\mathcal{L}_0^{jx}$ ($\mathbf{L}_c$) for each $(i,l) \in \mathcal{P}'_j$ is constructed as $\mathcal{L}_{i(l)}^{jx} = \{B[h][i] \mid v_h \in \mathcal{C}'_j, (i,l) \in \mathcal{P}'_j\}$. Notice that for each $v \in Flat(\mathcal{L}_{i(l)}^{jx})$, $A(v) = a_i$. For simplicity, we collect all the compact location lists from $\mathcal{L}_i \in \mathbf{L}_b \cup \mathbf{L}_c$ such that for each $v \in Flat(\mathcal{L}_i)$, $A(v) = a_j$. Let the number of such lists be $J$ denoted as $\mathcal{L}_1^j, \mathcal{L}_2^j, \ldots, \mathcal{L}_J^j$. These sets are the key players in Step 2.

### 3.3 Step 2-Computing the Joins

**Definition 6.** Given $p$ compact location lists $\mathcal{L}_i$, $1 \leq i \leq p$, some $\mathcal{L}'_1 \subseteq \mathcal{L}_1$ and a family of one-to-one mappings $M_i : (L_1 \in \mathcal{L}'_1) \rightarrow (L_i \in \mathcal{L}_i)$, $2 \leq i \leq p$, we call $\mathcal{L}'_M = \{L_1 \cap (\bigcap_{i=2}^{p} M_i(L_1)) \mid L_1 \in \mathcal{L}_1\}$ a join of $\mathcal{L}_i$, $1 \leq i \leq p$. If $\mathcal{L}'_1 = \mathcal{L}_1$ and the mappings $M_i$ are onto, then we call $\mathcal{L}'_M$ a *complete-join* of $\mathcal{L}_i$.

Clearly, the join of two sets is not necessarily unique [1]. For example, let $\mathcal{L}_1 =$

[1] Hence we use the term *join* instead of intersection.

$\{L_{11} = \{v_0, v_1, v_2\}, L_{12} = \{v_0, v_1, v_3\}\}$ and $\mathcal{L}_2 = \{L_{21} = \{v_1, v_2, v_4\}, L_{22} = \{v_1, v_2, v_3, v_4\}\}$. Then $\mathcal{L}_3 = \{L_{11} \cap L_{21}, L_{12} \cap L_{22}\} = \{\{v_1, v_2\}, \{v_1, v_3\}\}$ and $\mathcal{L}_4 = \{L_{11} \cap L_{22}, L_{12} \cap L_{21}\} = \{\{v_1\}, \{v_1, v_2\}\}$ are both joins of $\mathcal{L}_1$ and $\mathcal{L}_2$. Although it appears that the algorithm has to explore a very large space of mappings ($M_i$'s), since our interest is in "maximal" intersections, we can design an efficient algorithm to compute all the joins. In the last example both $\mathcal{L}_3$ and $\mathcal{L}_4$ are complete-joins. If $\mathcal{L}_5 = \{\{v_1, v_2\}, \{v_3, v_4\}\}$ and $\mathcal{L}_6 = \{\{v_2, v_1\}, \{v_5, v_6\}\}$, then $\mathcal{L}_7 = \{\{v_1, v_2\}\}$ is a join of $\mathcal{L}_5$ and $\mathcal{L}_6$ but not a complete-join.

Again due to space constraints, the running example is continued in the Appendix (Figure 6) for the referees.

For each attribute $a_j$, $\mathcal{L}_1^j, \mathcal{L}_2^j, \ldots, \mathcal{L}_J^j$ were computed at the end of last step. Let $n_i = |\mathcal{L}_i^j|$ and let $L_{ik} \in \mathcal{L}_i^j, 1 \le k \le n_i$, $1 \le i \le J$. XTION routine is along the lines of CONJUNCTS: CREATE-SET() stores the compact location lists in an appropriate data structure (say a tree) and the query in line (3) of the routine can be computed in $\log |V|$ time. At the top level, $\mathcal{L} = \{L_{ik} \mid 1 \le i \le J, 1 \le k \le n_i\}$, $h$ is the number of vertices with the attribute $a_j$ and $K = 1$.

```
XTION($\mathcal{L}, h, K$)
{
    (1) If ($h \le 0$) Then Exit
    (2) Let $\mathcal{L}'' = \{L_{ik} | v_h \in L_{ik} \in \mathcal{L}\}$
        Let $\mathcal{P} = \{(i,k) | L_{ik} \in \mathcal{L}''\}$
        Let $\mathcal{L}' = \{v_h\}$
    (3) If $|\mathcal{L}''| \ge K$ and not EXIST($\mathcal{L}''$)
            (4) CREATE-SET($\mathcal{L}'', \mathcal{L}', \mathcal{P}$)
            (5) XTION($\mathcal{L}'', h-1, K$)
    (6) If EXIST($\mathcal{L}''$) add $v_h$ to $\mathcal{L}'$
    (7) XTION($\mathcal{L}, h-1, K$)
}
```

**Postprocessing**: (1) Computing the new compact lists: This involves (a) computing the joins and (b) maximalization. (a) *Computing Joins*: At this phase new compact location lists are constructed from the results produced by XTION. A compact list is a collection of sets $L$ and $XTION$ first extracts all the $L$'s from the compact lists and then computes all the possible intersection of these $L$. In this step we reconstruct the collection of sets (as compact lists) back again as follows: Let $\mathcal{L}_l'$ and $\mathcal{P}_l$, $1 \le l \le \ell$ be the sets produced by the routine (line 2). For each $1 \le l \le \ell$, let $D_l = \{i \mid (i, -) \in \mathcal{P}_l\}$. In other words, each $D_l$ is a collection of the input compact list numbers. For example, let $P_1 = \{(1,10),(2,20),(3,30)\}$, then $D_l = \{1,2,3\}$. Notice that the element $(1,10) \in P_1$ denotes the element labeled 10 in compact list numbered 1; $(2,20) \in P_1$ denotes the element labeled 20 in compact list numbered 2 and so on. $D_1$ captures the non-empty intersection of the three compact lists numbered 1, 2 and 3 and the one-to-one mappings $M_i$ of Definition 6 sends element labeled 10 of compact list 1 to element labeled 20 of compact list 2 and to element labeled 30 of compact list 3.

On the other hand, if some $P_2 = \{(1,10),(1,15),(2,20),(3,30)\}$, then $D_2 = \{1,2,3\}$ and the two families of one-to-one mappings, $M^1$ and $M^2$ satisfy: (1) $M_2^1(10) \mapsto (20)$, $M_3^1(10) \mapsto (30)$, and, (2) $M_2^2(15) \mapsto (20)$, $M_3^2(15) \mapsto (30)$.

Obtain all the intersections of the $D's$ constructed in the last paragraph using the CONJUNCT routine: Let $\mathbf{D}$ be the collection of all $D$'s of the last step, then CONJUNCT($\mathbf{D}$, $N$,2) computes all the distinct intersections. For each distinct intersection $I = \bigcap_{i=l_1}^{l_m} D_i$, construct the compact location list $\{\{\mathcal{L}'_l\} \mid l = l_1, l_2, \ldots, l_m\}$. This new compact list is a join of the input compact lists (numbers) contained in $I$. For instance if $I = \{1, 2, 3\}$, then the new compact list $\mathcal{L}'$ is a join of input compact lists numbered 1, 2 and 3 say $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ respectively. If $|E(\mathcal{L}')| = |E(\mathcal{L}_i)|$, $i = 1, 2, 3$, then $\mathcal{L}'$ is a complete-join.

(b) *Maximalize*: For each new compact list $\mathcal{L}$ generated in the last phase, MAXIMALIZE($\mathcal{L}$) (see Section 3.2) and update the set of compact lists.

(2) Updating Conjugates: For each $\mathcal{L}_k \in \mathbf{L}$, construct for each conjugate $\overline{\mathcal{L}}_k^j$ and add it to $\mathbf{L}$. Thus as each location list loses elements due to the joins, its conjugate list (with a different attribute) also loses the same corresponding elements to create possibly new location lists.
Step 2 is iterated until no more new location lists are formed.

Due to space constraints the proof of correctness of the algorithm has been omitted and is presented in the appendix for the referees.

### 3.4 Step 3-Maximal motifs from the compact lists

By the end of Step 2 all the information regarding the maximal motifs has been gathered namely the location lists in the compact form. This step is more about organizing the results of the first two steps.

Let $\mathbf{L}$ be the collection of all compact lists at the end of Step 2. Consider a graph $G(V, E)$ where $(v_i \in V)$ corresponds to the location list $(\mathcal{L}_i \in \mathbf{L})$ and $(v_i v_j) \in E$ if (1) $\mathcal{L}_i$ is a conjugate of $\mathcal{L}_j$ or (2) $\mathcal{L}_i$ is a complete-join of $\mathcal{L}_j$. Conjugate denotes immediate neighbors of vertices and complete-join denotes the extension of the neighborhood information of the vertices in a compact list. Thus, it is easy to see that the connected components of the graph correspond to maximal motifs and they can be deduced from $G$. Continuing the concrete example, the maximal motif shown in Figure 2 is obtained from (1) $\mathcal{L}_0^{\square 5}$, (2) $\mathcal{L}_{\bigcirc(2)}^{\square 5}$, (3) $\mathcal{L}_0^{\bigcirc 3}$ and (4) $\mathcal{L}_{\square(3)}^{\bigcirc 2}$. (1) and (2) are conjugates, (2) and (3) are complete-joins and (3) and (4) are conjugates.

### 3.5 Time Complexity

We give the time complexity in terms of the size of the output $S$ which is considered to be the sum total of all the location lists of all the maximal motifs. Notice that every compact list that is generated at each step corresponds to a location list of some maximal motif.

Consider Step 1 (Section 3.2). It takes time $O(|E|)$ to initialize $B$ since each edge is read at most twice during the process. Next, in the routine CONJUNCTS, all the resulting sets, which are $N$ in number are at the leaf node of tree implicitly computed by it. Also, the number of internal nodes can not exceed the number of leaf nodes, $N$. Thus the total number of internal nodes of this implicit tree is $O(N)$. The cost of the query at each node is $O(\log |V|)$ (line (3) of CONJUNCTS). The size of the input data is $O(|V|D)$ and each data item is read exactly once in the algorithm (line (2) of CONJUNCTS) Hence the algorithm takes $O(N \log |V| + |V|D)$ time. The postprocessing again takes time $O(|E|)$ since each edge is represented at most twice in $B$. Thus this takes $O(|E| + N \log |V|)$ where $N$ is the number of distinct location lists at this step. The postprocessing takes time linear in the size of the output at this step.

Consider Step 2 (Section 3.3). The routine XTION is along the line of CONJUCT and using similar arguments the step takes $O(N_2 \log |N_2| + N_1)$ where $N_1$ is the size of the input and $N_2$ is the size of the output at this step. The distinct intersection computation takes $O(N_3 \log |N_3| + N_2)$ where $N_3$ is the size of the output. Again the postprocessing takes time linear in the size of the output at this stage.

Notice that at each step $N, N_1, N_2, N_3 \leq S$ where $S$ is the size of the output in terms of its location lists. Let the number of times Step 2 is iterated be $m$, then $m$ is bounded by $\log M$ where $M$ is the maximum number of vertices in any maximal motif. The total time taken by the algorithm is $O(|E| + S \log S \log M)$ as Step 3 is just a linear time operation.

## 4 Discussion

Here we present a general framework for discovering common recurring structures or motifs in data. The framework has been developed for graphs. The next natural variation is to allow for "wild card" nodes, i.e. nodes whose attributes can be ignored in the motif. Again for such variations, (1) the *conjunct* operation needs to be redefined to allow for wild card nodes and (2) the order in which the join operations are undertaken. It is clear that similar other notions of approximate topological motifs can be developed along these lines.

If we restrict the graphs to represent strings, does the algorithm specialize to finding motifs in strings? We claim that it does and in fact gives rise to the algorithm presented in [CP04] for finding extensible patterns in strings. Again, this simply involves specializing *conjunct* and *join* operations appropriately.

## 5 Conclusion & Ongoing Work

We have presented a method for discovering all common subgraphs that appear at least $K$ times in a graph or multiple graphs. The usual problem due to isomorphism is handled effectively by the use of a compact notation that abbreviates the output substantially without any loss of information. It also presents an alternative way of presenting the results as compact location lists and opens the possibility of evaluating significance of the motifs in a specific domain directly on these compact lists. We are currently looking into using this technique for discovering motifs in metabolic pathways and biological network data.

## References

[AP04] A. Apostolico and L. Parida. Incremental paradigms for motif discovery. *Journal of Computational Biology*, 11(4):15–25, 2004.

[CP04] A. Chattaraj and L. Parida. An inexact suffix tree based algorithm for extensible pattern discovery. *Theoretical Computer Science*, 2004.

[DBNW03] O. Dror, H. Benyamini, R. Nussinov, and H. J. Wolfson. Multiple structural alignment by secondary structures: Algorithm and applications. *Protein Science*, 12(11):2492–2507, 2003.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco, 1979.

[HETC00] J. D. Hughes, P. W. Estep, S. Tavazoie, and G. M. Church. Computational identification of cis-regulatory elements associated with groups of functinally related genes in Saccharomyces cerevisiae. *J Molec. Bio.*, 296:1205–1214, 2000.

[HPS03] H. H.Gan, S. Pasquali, and T. Schlick. Exploring the repertoire of RNA secondary motifs using graph theory: implications for RNA design. *Nucleic Acids Research*, 31(11):2926–2943, 2003.

[IWM03] A. Inokuchi, T. Washio, and H. Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, 50(3):321–354, 2003.

[LMF03] A. V. Lukashin, M.E.Lakashev, and R. Fuchs. Topology of gene expression networks as revealed by data mining and modeling. *Bioinformatics*, 19(15):1909–1916, 2003.

[MSOI$^+$02] R Milo, S Shen-Orr, S Itzkovitz, N Kashtan, D Chklovskii, and U Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.

[Mur03] T. Murata. Graph mining approaches for the discovery of web communities. *Proceedings of the International Workshop on Mining Graphs, Trees and Sequences*, pages 79–82, 2003.

[Par00] Laxmi Parida. Some results on flexible-pattern matching. In *Proc. of the Eleventh Symp. on Comp. Pattern Matching*, volume 1848 of *Lecture Notes in Computer Science*, pages 33–45. Springer-Verlag, 2000.

[SBH$^+$01] I. Simon, J. Barnett, N. Hannett, C. T. Harbison, N. J. Rinaldi, T. L. Volkert, J. J. Wyrick, J. Zeitlinger, D. K. Gifford, T. S. Jaakkola, and R. A. Young. Serial regulation of transcriptional regulators in the yeast cell cycle. *Cell*, 106:697–708, 2001.

[SOB03] S.Wuchty, Z.N. Oltvai, and A-L Barabasi. Evolutionary conservation of motif constituents in the yeast protein interactin networks. *Nature Genetics*, 35(2):176–179, 2003.

[SOMMA02] S.S. Shen-Orr, R. Milo, S. Mangan, and U. Alon. Network motifs in the transcriptional regulation network of Escherichia coli. *Nature Genetics*, 31:64–68, 2002.

[Zak02] M. J. Zaki. Efficiently mining frequent trees in a forest. *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, July, 2002.

## Appendix A

Given a general graph $G$, we compute an undirected graph $G'$ that has only vertex attributes by the following steps.

1. Introduce suffixes to vertices and edges with identical attributes.
2. (a) (directed graph) For each incoming edge with attribute $x_i$, vertex with attribute $A_j$ and outgoing edge with attribute $y_k$, create a node with attribute $x_i A_j y_k$ in $G'$.
   (b) (undirected graph) For each incident edge with attribute $y_k$, vertex with attribute $A_j$ create a node with attribute $A_j y_k$ in $G'$.
3. In $G'$ for each pair of nodes with attributes
   (a) (directed graph) –$y_k$ and $y_k$–,
   (b) (undirected graph) -$y_k$ and -$y_k$,
   introduce an edge between these nodes.

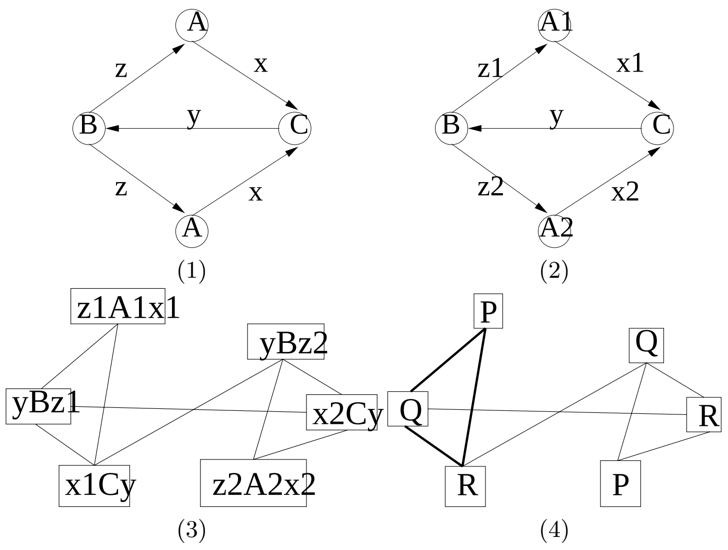

**Fig. 3.** (1) A directed graph with vertex and edge attributes. (2) The annotated directed graph. (3) Construction of an undirected graph from Step 2. (4) The undirected graph with no edge attributes and a motif is shown in bold on this graph. Here $P = zAx, Q = yBz, R = xCy$.

## Appendix B

**Proof of Correctness of the algorithm**: At the end of Step 2, we have a collection of compact lists that have the property discussed below. We first make the observation: $|\mathcal{L}^{k+1}| \leq |\mathcal{L}^{x_i}|$, $1 \leq i \leq p$, where $\mathcal{L}^{k+1}$ is generated at iteration $(k+1)$ from $\mathcal{L}^{x_i}$, generated at iteration $(x_i \leq k)$, and, $1 \leq i \leq p$.

$\mathcal{L}^{k+1}$ can be generated in the two ways: (1) $\mathcal{L}^{k+1}$ is the join of $\mathcal{L}^{x_i}$, $1 \leq i \leq p$ (then $|\mathcal{L}^{k+1}| \leq |\mathcal{L}^{x_i}|$ for each $i$), and (2) $\mathcal{L}^{k+1}$ is the conjugate of $\mathcal{L}^{x_i}$ for some $x_i \leq k$. Since the mapping is one-to-one, $|\mathcal{L}^{k+1}| \leq |\mathcal{L}^{x}|$.

Let $\mathbf{L}$ be the set of all compact lists computed at the end of Step 2. Let $M(V_m, E_m)$ be a maximal motif and let $U$-$\mathcal{L}$ be the compact location list where $U \subseteq V_m$ is a set of maximal isomorphic vertices in $M$. Let $\mathbf{L}'$ be the set of all such compact lists of all maximal motifs.

**Theorem 7.** $\mathbf{L} = \mathbf{L}'$.

We begin with a few lemmas.

**Lemma 8.** *Let $M(V_m, E_m)$ be a maximal motif with a set of maximal isomorphic vertices $U_1 \subseteq V_m$ and $\mathcal{L} = U$-$\mathcal{L}$. Let $\overline{\mathcal{L}}^j$ be one of the conjugates of $\mathcal{L}$. Then there exists at least one maximal motif $M'(V'_m, E'_m)$ with isomorphic vertices $U_1, U_2 \subseteq V'_m$ with $U_1$-$\mathcal{L} = \mathcal{L}$ and $U_2$-$\mathcal{L} = \overline{\mathcal{L}}^j$.*

If $M$ is maximal then $U$-$\mathcal{L} = \mathcal{L}$ is called a *maximal* location list. Also, let $\mathcal{M}(\mathcal{L})$ denote the set of maximal motifs which have a set of isomorphic vertices $U$ with $U$-$\mathcal{L} = \mathcal{L}$.

**Lemma 9.** *For $i = 1, 2$, let $M_i(V_{m_i}, E_{m_i})$ be maximal motifs with a set of isomorphic vertices, $U_i \subseteq V_{m_i}$. If $U_1$-$\mathcal{L}_1 \subset U_2$-$\mathcal{L}_2$, then $M_2$ is a proper subgraph of $M_1$. The converse is also true.*

Since $\mathcal{L}_1 \subset \mathcal{L}_2$, $M_2$ must occur in all the locations of $\mathcal{L}_1$ and so does $M_1$. By Lemma 8, $M_2$ is a subgraph of $M_1$. Conversely, if $M_2$ is a subgraph of $M_1$, then $M_2$ occurs at all the locations in $\mathcal{L}_1$ and $\mathcal{L}_2$, hence $\mathcal{L}1 \subset \mathcal{L}_2$.

**Lemma 10.** *Let $M_i(V_{m_i}, E_{m_i})$, $1 \leq i \leq p$, be maximal motifs on $G(V, E)$, let $M$ be the smallest graph (also on $G$) such that each of $M_i$ is a subgraph of $M(V_m, E_m)$, then $M(V_m, E_m)$ also must be a maximal motif on $G(V, E)$.*
*Let $U$ be a set of isomorphic vertices in $V_m, V_{m_i}$, $1 \leq i \leq p$ such that they are isomorphic to each other. Then $U$-$\mathcal{L}$ is the join of $U$-$\mathcal{L}_i$, $1 \leq i \leq p$.*

**Lemma 11.** $\mathbf{L}$ *is closed under join and closed under conjugation.*

Notice that Steps (1-b) and (2) ensure these conditions.
**Back to the main theorem.** We first show that $\mathbf{L}'$ is closed under conjugation and join. Let $\overline{\mathcal{L}}^j$ be the conjugate of some $\mathcal{L} \in \mathbf{L}'$. By Lemma 8, $\overline{\mathcal{L}}^j \in \mathbf{L}'$. Hence $\mathbf{L}'$ is closed under conjugation. Consider ($\mathcal{L}_0$ is the join of ($\mathcal{L}_i \in \mathbf{L}'$), $1 \leq i \leq p$, $\mathcal{L}_0 \notin \mathbf{L}'$. Since $\mathcal{L}_i \supseteq \mathcal{L}_0$, $1 \leq i \leq p$, by Lemma 9, for each $M_i \in \mathcal{M}(\mathcal{L}_i)$ and

for each $M' \in \mathcal{M}(\mathcal{L}_0)$, $M_i$ is a subgraph of $M'$. Since each $M_i$ is maximal. By Lemma 10, $M'$ must be maximal. So $\mathcal{L}_0 \in \mathbf{L}'$. Hence $\mathbf{L}'_b$ is closed under join. (I)

For any $\mathcal{L} \in \mathbf{L}$ there exists a $\mathcal{L}' \in \mathbf{L}_b$ such that $\mathcal{L}' \supseteq \mathcal{L}$. Also, clearly $\mathbf{L}_b \subseteq \mathbf{L}'$. From Lemma 11 and (I), both $\mathbf{L}$ and $\mathbf{L}'$ are closed under join and conjugation; hence $\mathbf{L} \subseteq \mathbf{L}'$. (II)

We will show that $\mathbf{L}' \setminus \mathbf{L} = \phi$. Let $\mathcal{L} \in \mathbf{L}' \setminus \mathbf{L}$. Then $\mathcal{L}$ must have the form that there exist some $p \geq i \geq 1$ location lists with $(\mathcal{L}_i \supset \mathcal{L})$. Let $\mathcal{L}'$ be the join of these $p$ location lists, hence $\mathcal{L}' \in \mathbf{L}$. Assume $\mathcal{L} \neq \mathcal{L}'$. Let $\mathcal{M}(\mathcal{L})$ denote the collection of maximal motifs such that $\mathcal{L}$ is a location list correpsonding to (possibly set) vertex in the motif. Consider $M(V_m, E_m) \in \mathcal{M}(\mathcal{L})$ and $M'(V'_m, E'_m) \in \mathcal{M}(\mathcal{L}')$. Without loss of generality $M'$ is a proper subgraph of $M$ (by Lemma 9). Hence there must exist a vertex $u_1 \in V_m$ with isomorphic vertices $u'_1 \in V'_m$ satisfying $A(u_1) = A(u'_1) = a_j$. Further, there is some $u_2 \in V_m, (u_1 u_2) \in E_m$, $A(u_2) = a_i$ and no vertex isomorphic to $u_2$ in $M'$. Consider some $\mathcal{L}_b \in \mathbf{L}_b$ with signature $S(\mathcal{L}_b) = ((a_j, \text{-}), P)$, where $(a_i, \text{-}) \in P$. Then clearly $\mathcal{L}_b \cap \mathcal{L}' = \mathcal{L}$, hence $\mathcal{L} \in \mathbf{L} \cap \mathbf{L}'$ contradicting the assumption and $\mathcal{L} = \mathcal{L}'$. Hence $\mathbf{L}' \subseteq \mathbf{L}$. (III)

From (II) and (III), $\mathbf{L} = \mathbf{L}'$. $\square$

Matrix B:

| $v_i\ (A(v_i))$ | $\square$ | $\bigcirc$ |
|---|---|---|
| $v_1(\square)$ | $\phi$ | $\{v_6, v_7, v_8\}$ |
| $v_2(\square)$ | $\phi$ | $\{v_6, v_7, v_8\}$ |
| $v_3(\square)$ | $\phi$ | $\{v_6, v_7, v_8, v_9\}$ |
| $v_4(\square)$ | $\{v_5\}$ | $\{v_9, v_{10}\}$ |
| $v_5(\square)$ | $\{v_4\}$ | $\{v_9, v_{10}\}$ |
| $v_6(\bigcirc)$ | $\{v_1, v_2, v_3\}$ | $\phi$ |
| $v_7(\bigcirc)$ | $\{v_1, v_2, v_3\}$ | $\phi$ |
| $v_8(\bigcirc)$ | $\{v_1, v_2, v_3\}$ | $\phi$ |
| $v_9(\bigcirc)$ | $\{v_3, v_4, v_5\}$ | $\{v_{10}\}$ |
| $v_{10}(\bigcirc)$ | $\{v_4, v_5\}$ | $\{v_9\}$ |

PREPROCESS produces the following:

$\mathcal{C}_{\square}^{\square(1)} = \{v_4, v_5\}$
$\mathcal{C}_{\square}^{\bigcirc(2)} = \{v_1, v_2, v_3, v_4, v_5\}$
$\mathcal{C}_{\square}^{\bigcirc(3)} = \{v_1, v_2, v_3\}$
$\mathcal{C}_{\square}^{\bigcirc(4)} = \{v_3\}$

$\mathcal{C}_{\bigcirc}^{\square(2)} = \{v_6, v_7, v_8, v_9, v_{10}\}$
$\mathcal{C}_{\bigcirc}^{\square(3)} = \{v_6, v_7, v_8, v_9\}$
$\mathcal{C}_{\bigcirc}^{\bigcirc(1)} = \{v_9, v_{10}\}$

**Fig. 4.** Consider the graph of Figure 1(a). The graph has 10 vertices and two kinds of vertex attributes (1) $\square$ and (2) $\bigcirc$. The correesponinng adjacency matrix is shown as $B$ along with the set produced in PREPROCESS of Step 1.

This article was processed using the LaTeX macro package with LLNCS style

CONJUNCTS
$(\{\mathcal{C}_{\square}^{\square(1)}, \mathcal{C}_{\square}^{\bigcirc(2)}, \mathcal{C}_{\square}^{\bigcirc(3)}\}, 10, 2)$

The successive values taken by $\mathcal{P}'_{\square}$ are:
$\mathcal{P}'_{\square}=\{(\square,1),(\bigcirc,2)\}$, $\mathcal{P}'_{\square}=\{(\bigcirc,2)\}$

The post-processing results are:

(1) From $\mathcal{P}'_{\square}=\{(\square,1),(\bigcirc,2)\}$

$$\begin{array}{rcll}
\mathcal{L}_{\square(1)}^{\square 1} & = & \{\{v_5\}, & \{v_4\}\} \\
 & & \updownarrow & \updownarrow \\
\mathcal{L}_{0}^{\square 1} & = & \{\{v_4\}, & \{v_5\}\} \\
 & & \updownarrow & \updownarrow \\
\mathcal{L}_{\bigcirc(2)}^{\square 1} & = & \{\{v_9, v_{10}\}\} &
\end{array}$$

(2) From $\mathcal{P}'_{\square}=\{(\bigcirc,2)\}$

$$\begin{array}{rclll}
\mathcal{L}_{0}^{\square 2} & = & \{\{v_1,v_2,v_3\}, & \{v_4,v_5\}, & \{v_3\}\} \\
 & & \updownarrow & \updownarrow & \updownarrow \\
\mathcal{L}_{\bigcirc(2)}^{\square 2} & = & \{\{v_6,v_7,v_8\}, & \{v_9,v_{10}\}, & \{v_6,v_7,v_8,v_9\}\}
\end{array}$$

(3) From (2) $(\mathcal{L}_{\bigcirc(2)}^{\square 2})$ (by MAXIMALIZE)

$$\begin{array}{rcll}
\mathcal{L}_{0}^{\square 3} & = & \{\{v_1,v_2,v_3\}, & \{v_3\}\} \\
 & & \updownarrow & \updownarrow \\
\mathcal{L}_{\bigcirc(3)}^{\square 3} & = & \{\{v_6,v_7,v_8\}, & \{v_6,v_7,v_8,v_9\}\}
\end{array}$$

(4) From (2) $(\mathcal{L}_{\bigcirc(2)}^{\square 2})$ (by MAXIMALIZE)

$$\begin{array}{rcl}
\mathcal{L}_{0}^{\square 4} & = & \{\{v_3\}\} \\
 & & \updownarrow \\
\mathcal{L}_{\bigcirc(4)}^{\square 4} & = & \{v_6,v_7,v_8,v_9\}\}
\end{array}$$

(5) From (2) $(\mathcal{L}_{0}^{\square 2})$ (by MAXIMALIZE)

$$\begin{array}{rcll}
\mathcal{L}_{0}^{\square 5} & = & \{\{v_1,v_2,v_3\}, & \{v_4,v_5\}\} \\
 & & \updownarrow & \updownarrow \\
\mathcal{L}_{\bigcirc(2)}^{\square 5} & = & \{\{v_6,v_7,v_8\}, & \{v_9,v_{10}\}\}
\end{array}$$

CONJUNCTS
$(\{\mathcal{C}_{\bigcirc}^{\square(2)}, \mathcal{C}_{\bigcirc}^{\square(3)}, \mathcal{C}_{\bigcirc}^{\bigcirc(1)}\}, 10, 2)$

The successive values taken by $\mathcal{P}'_{\bigcirc}$ are:
$\mathcal{P}'_{\bigcirc}=\{(\square,2),(\bigcirc,1)\}$, $\mathcal{P}'_{\bigcirc}=\{(\square,3)\}$,
$\mathcal{P}'_{\bigcirc}=\{(\square,3),(\bigcirc,1)\}$

The post-processing results are:

(1) From $\mathcal{P}'_{\bigcirc}=\{(\square,2),(\bigcirc,1)\}$

$$\begin{array}{rcll}
\mathcal{L}_{\bigcirc(1)}^{\bigcirc 1} & = & \{\{v_{10}\}, & \{v_9\}\} \\
 & & \updownarrow & \updownarrow \\
\mathcal{L}_{0}^{\bigcirc 1} & = & \{\{v_9\}, & \{v_{10}\}\} \\
 & & \updownarrow & \updownarrow \\
\mathcal{L}_{\square(2)}^{\bigcirc 1} & = & \{\{v_3,v_4,v_5\}, & \{v_4,v_5\}\}
\end{array}$$

(2) From $\mathcal{P}'_{\bigcirc}=\{(\square,2)\}$

$$\begin{array}{rclll}
\mathcal{L}_{0}^{\bigcirc 2} & = & \{\{v_6,v_7,v_8\}, & \{v_9,v_{10}\}, & \{v_9\}\} \\
 & & \updownarrow & \updownarrow & \updownarrow \\
\mathcal{L}_{\square(2)}^{\bigcirc 3} & = & \{\{v_1,v_2,v_3\}, & \{v_4,v_5\}, & \{v_3,v_4,v_5\}\}
\end{array}$$

(3) From (2) $(\mathcal{L}_{\square(2)}^{\bigcirc 3})$ (by MAXIMALIZE)

$$\begin{array}{rcll}
\mathcal{L}_{0}^{\bigcirc 3} & = & \{\{v_6,v_7,v_8\}, & \{v_9\}\} \\
 & & \updownarrow & \updownarrow \\
\mathcal{L}_{\square(3)}^{\bigcirc 2} & = & \{\{v_1,v_2,v_3\}, & \{v_3,v_4,v_5\}\}
\end{array}$$

(4) From $\mathcal{P}'_{\bigcirc}=\{(\square,3),(\bigcirc,1)\}$

$$\begin{array}{rcl}
\mathcal{L}_{\bigcirc(1)}^{\bigcirc 4} & = & \{\{v_{10}\}\} \\
 & & \updownarrow \\
\mathcal{L}_{0}^{\bigcirc 4} & = & \{\{v_9\}\} \\
 & & \updownarrow \\
\mathcal{L}_{\square(3)}^{\bigcirc 4} & = & \{\{v_3,v_4,v_5\}\}
\end{array}$$

**Fig. 5.** Continuing the example of Figure 4 (Step 1): Since the graph has two distinct attributes, the two CONJUNCTS calls are as shown. Note that $h = 10, K = 2$ for the calls. The two-way link between the conjugate elements is shown by $\updownarrow$.

Compact location lists with attribute $\square$:
$\mathcal{L}^{\square 1}_{\square(1)} = \mathcal{L}^{\square 1}_{0} = \{\{v_5\},\{v_4\}\}$
$\mathcal{L}^{\square 2}_{0} = \{\{v_1,v_2,v_3\},\{v_4,v_5\},\{v_3\}\}$
$\mathcal{L}^{\square 3}_{0} = \{\{v_1,v_2,v_3\},\{v_3\}\}$
$\mathcal{L}^{\bigcirc 1}_{\square(2)} = \{\{v_3,v_4,v_5\},\{v_4,v_5\}\}$
$\mathcal{L}^{\bigcirc 2}_{\square(3)} = \{\{v_1,v_2,v_3\},\{v_3,v_4,v_5\}\}$
$\mathcal{L}^{\bigcirc 3}_{\square(2)} = \{\{v_1,v_2,v_3\},\{v_4,v_5\},\{v_3,v_4,v_5\}\}$
$\mathcal{L}^{\square 4}_{0} = \{\{v_3\}\}$
$\mathcal{L}^{\bigcirc 4}_{\square(3)} = \{\{v_3,v_4,v_5\}\}$
$\mathcal{L}^{\square 5}_{0} = \{\{v_1,v_2,v_3\},\ \{v_4,v_5\}\}$

XTION
$(\{\mathcal{L}^{\square 1}_{\square(1)},\ \mathcal{L}^{\square 2}_{0},\ \mathcal{L}^{\square 3}_{0},\ \mathcal{L}^{\bigcirc 1}_{\square(2)},\ \mathcal{L}^{\bigcirc 2}_{\square(3)},\ \mathcal{L}^{\bigcirc 3}_{\square(2)},$
$\mathcal{L}^{\square 4}_{0},\ \mathcal{L}^{\bigcirc 4}_{\square(3)},\ \mathcal{L}^{\square 5}_{0}\ \},\ 5{,}2)$

A new list from the join is:
(1) $\mathcal{L}^{\square 1}_{new} = \{\{v_3\},\{v_4,v_5\}\}$
(Join of $\mathcal{L}^{\square 2}_{0}$, $\mathcal{L}^{\bigcirc 1}_{\square(2)}$, $\mathcal{L}^{\bigcirc 2}_{\square(3)}$, $\mathcal{L}^{\bigcirc 3}_{\square(2)}$,$\mathcal{L}^{\square 5}_{0}$ is $\mathcal{L}^{\square 1}_{new}$;
All, except with $\mathcal{L}^{\square 2}_{0}$, are *complete-joins*)

The new list from conjugation is:
(1) Conjugate of $\mathcal{L}^{\square 1}_{new}$ :
$\mathcal{L}^{\bigcirc 3}_{new} = \{\{v_6,v_7,v_8,v_9\},\{v_9,v_{10}\}\}$

Compact location lists with attribute $\bigcirc$:
$\mathcal{L}^{\bigcirc 1}_{\bigcirc(1)} = \mathcal{L}^{\bigcirc 1}_{0} = \{\{v_9\},\{v_{10}\}\}$
$\mathcal{L}^{\square 1}_{\bigcirc(2)} = \{\{v_9,v_{10}\}\}$
$\mathcal{L}^{\square 2}_{\bigcirc(2)} =$
$\{\{v_6,v_7,v_8\},\{v_9,v_{10}\},\{v_6,v_7,v_8,v_9\}\}$
$\mathcal{L}^{\square 3}_{\bigcirc(3)} = \{\{v_6,v_7,v_8\},\{v_6,v_7,v_8,v_9\}\}$
$\mathcal{L}^{\bigcirc 3}_{0} = \{\{v_6,v_7,v_8\},\{v_9\}\}$
$\mathcal{L}^{\bigcirc 2}_{0} = \{\{v_6,v_7,v_8\},\{v_9,v_{10}\},\{v_9\}\}$
$\mathcal{L}^{\square 4}_{\bigcirc(4)} = \{v_6,v_7,v_8,v_9\}\}$
$\mathcal{L}^{\bigcirc 4}_{\bigcirc(1)} = \{\{v_{10}\}\}$
$\mathcal{L}^{\bigcirc 4}_{0} = \{\{v_9\}\}$
$\mathcal{L}^{\square 5}_{\bigcirc(2)} = \{\{v_6,v_7,v_8\},\ \{v_9,v_{10}\}\}$

XTION $(\{\mathcal{L}^{\bigcirc 1}_{\bigcirc(1)},\ \mathcal{L}^{\square 1}_{\bigcirc(2)},\ \mathcal{L}^{\square 2}_{\bigcirc(2)},\ \mathcal{L}^{\square 3}_{\bigcirc(3)},$
$\mathcal{L}^{\bigcirc 2}_{0},\ \mathcal{L}^{\bigcirc 3}_{0},\ \mathcal{L}^{\square 4}_{\bigcirc(4)},\ \mathcal{L}^{\bigcirc 4}_{\bigcirc(1)},\ \mathcal{L}^{\bigcirc 4}_{0},\ \mathcal{L}^{\square 5}_{\bigcirc(2)}\},\ 5{,}2)$

A new list from the join is:
(1) $\mathcal{L}^{\bigcirc 1}_{new} = \{\{v_9\},\{v_9,v_{10}\}\}$
($\mathcal{L}^{\bigcirc 1}_{new}$ is the join of $\mathcal{L}^{\square 2}_{\bigcirc(2)}$ and $\mathcal{L}^{\bigcirc 2}_{0}$)

A new list from conjugation is:
(1) Conjugate of $\mathcal{L}^{\bigcirc 1}_{new}$ :
$\mathcal{L}^{\bigcirc 2}_{new} = \{\{v_{10}\},\{v_9,v_{10}\}\}$

**Fig. 6.** Continuing the example of Figure 4 (Step 2): The two groups of compact location lists generated in the last step are shown first. To avoid clutter, we have omitted the *join* labels on the connection between the compact lists. The iteration stops when no more new compact lists are generated.