12

Matching Hierarchical Graphs

12.1 Introduction

In general, the computation of graph similarity is a very costly task. In the context of this book, however, we focus on a special class of graphs that allow for low-order polynomial-time matching algorithms. The considered class of graphs is characterized by the constraint that each node has a unique node label. This constraint is met in all computer network monitoring and abnormal event detection applications considered in this book.

Future applications of graph matching may require one to deal with graphs consisting of tens or even hundreds of thousands of nodes. For these applications low-order polynomial matching algorithms, such as those considered in previous chapters, may be still too slow. In this chapter we introduce a hierarchical graph representation scheme that is suitable for reducing the size of the graphs under consideration. Other reduction schemes have been proposed in [109], for example. There are also some conceptual similarities with hierarchical quadtree, or pyramid, representations in image processing [4]. The basic idea underlying the proposed hierarchical representation scheme is to contract some nodes of the given graph and represent them as a single node at a higher level of abstraction. There are no particular assumptions about the criteria that control the selection of nodes to be contracted into a single node at a higher abstraction level. For the contraction process, any algorithm that clusters nodes of a graph, including heuristic selection strategies or the algorithms discussed in Chapter 7, may be chosen. Properties of the nodes that are contracted are stored as attributes with the corresponding node at the higher level of abstraction. This process can be carried out in a hierarchical, iterative fashion, which will allow us to eventually contract any arbitrarily large set of nodes into a single node.

Because of the reduced number of nodes, computing the similarity of two graphs at a higher level of abstraction can be expected to be much faster than the corresponding computation on the original graphs. It is, however, desirable that the graph contraction procedure, as well as the chosen graph distance measure, have some monotonicity properties. That is, if graph g_1 is more similar to g_2 than to g_3 at the original, full graph resolution level, then this property should be maintained for the representation at any

higher level of abstraction. In this chapter we study several of these properties. While the general monotonicity property, as stated above, can't be guaranteed, we will derive upper and lower bounds of the graph similarity measure at higher levels of abstraction. It will be shown that under certain conditions these bounds are tight, i.e., they are identical to the real similarity value.

In the next section, the proposed graph abstraction scheme is presented. Then in Section 12.3, our new graph similarity measures will be defined and upper and lower bounds for graph distance at higher levels of abstraction derived. Next, potential applications of the proposed graph contraction scheme and the similarity measures in the domain of computer network monitoring will be discussed. In Section 12.5 the results of an experimental study will be presented. Finally, a summary and conclusions will be provided in Section 12.6.

12.2 Hierarchical Graph Abstraction

In this chapter we consider graphs $g = (V, E, \alpha, \beta)$ with unique node labels, and use the following graph edit distance:

$$d(g_i, g_j) = |V_i| + |V_j| - 2|V_i \cap V_j| + |E_i| + |E_j| - 2|E_i \cap E_j| \,. \tag{12.1}$$

This edit distance is identical to the edit distance introduced in Chapter 4 for the case that we neglect edge weight and are just interested in whether an edge is present between a given pair of nodes.

We start our graph abstraction process by partitioning the set of nodes V into a set of subsets, or clusters, $C = \{c_1, \ldots, c_n\}$, where $c_i \subseteq V$, $c_i \cap c_j = \emptyset$, $\bigcup_{i=1}^{n} c_i = V$ for $i \neq j$; $i, j = 1, \ldots, n$.

Definition 12.1. Given a graph g and an arbitrary partitioning C, a *hierarchical abstraction* of g is the graph $\bar{g} = (\bar{V}, \bar{E}, \bar{\alpha}, \bar{\beta})$ where:

(i) $\bar{V} = C$, i.e., each node in $\bar{g}$ represents a cluster of nodes in g (hence $\bar{V} = \{c_1, \ldots, c_n\}$);
(ii) $\bar{E} = \bar{V} \times \bar{V}$, i.e., $\bar{g}$ is fully connected;
(iii) $\bar{\alpha}(v) = (nodes(v), edges(v))$ for each $v \in \bar{V}$, such that
- $nodes(v) = |c|$, where v represents c
- $edges(v) = |\{e \mid e = (x, y) \in E \wedge x \in c \wedge y \in c\}|$, where v represents c.

That is, each node in $\bar{g}$ gets two attributes, $nodes(v)$ and $edges(v)$, assigned to it. The attribute $nodes(v)$ is equal to the number of nodes in graph g that belong to the cluster represented through v, while $edges(v)$ is equal to the number of edges in that cluster in graph g; and

(iv) $\bar{\beta}(e) = |\{(x, y) \mid (x, y) \in E \wedge x \in c_i \wedge y \in c_j \wedge e = (c_i, c_j)\}|$ for each $e \in \bar{E}$. That is, if e is an edge in $\bar{g}$ originating at the node representing cluster c_i and terminating at the node representing cluster c_j, then we count the number of edges in g that lead from a node in c_i to a node on c_j.

Example 12.2. A graph g_i and its hierarchical abstraction $\bar{g}_i$ are shown in Figure 12.1. For these graphs we observe that $V_i = \{1, 2, 3, 4, 5\}$ and $E_i = \{(1, 2), (1, 4), (2, 1), (2, 4), (2, 5), (3, 1), (4, 3)\}$.

We assume that V_i is partitioned into $C = \{\{1, 2\}, \{3, 4\}, \{5\}\}$, i.e., $c_1 = \{1, 2\}$, $c_2 = \{3, 4\}$, $c_3 = \{5\}$. The hierarchical abstraction $\bar{g}_i = (\bar{V}, \bar{E}, \bar{\alpha}, \bar{\beta})$ is then given by

$$\begin{aligned}
&\bar{V}_i = \{\{1, 2\}, \{3, 4\}, \{5\}\} = \{c_1, c_2, c_3\},\\
&\bar{E}_i = \{(c_1, c_2), (c_1, c_3), (c_2, c_1)\},\\
&\text{nodes}_i : c_1 \to 2, c_2 \to 2, c_3 \to 1,\\
&\text{edges}_i : c_1 \to 2, c_2 \to 1, c_3 \to 0,\\
&\bar{\beta}_i : (c_1, c_2) \to 2, (c_1, c_3) \to 1, (c_2, c_1) \to 1.
\end{aligned}$$

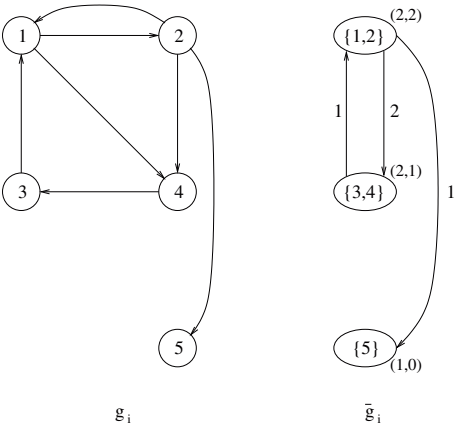

Fig. 12.1. A graph g_i and its hierarchical abstraction $\bar{g}_i$.

All edges e that have an attribute value $\beta(e) = 0$ are not included in Figure 12.1. In the graphical representation of $\bar{g}_i$ in Figure 12.1, the pairs (x, y) displayed next to the nodes correspond to the node attributes, i.e., $x = \text{nodes}(v)$, $y = \text{edges}(v)$. Similarly, the numbers next to the edges correspond to the edge attributes.

12.3 Distance Measures for Hierarchical Graph Abstraction

In this section we introduce two distance measures for the hierarchical graph abstraction introduced in Section 12.2, and discuss relationships with the measure defined in equation (12.1). Throughout this section we assume that $g_i = (V_i, E_i, \alpha_i, \beta_i)$ and $g_j = (V_j, E_j, \alpha_j, \beta_j)$ are two given graphs. The nodes and edges of both graphs come from (possibly larger) sets V and E, respectively, i.e., $V_i \cup V_j \subseteq V$, $E_i \cup E_j \subseteq E$,

and $C = \{c_1, \ldots, c_n\}$ is a partition of V. The graphs $\bar{g}_i = (\bar{V}_i, \bar{E}_i, \bar{\alpha}_i, \bar{\beta}_i)$ and $\bar{g}_j = (\bar{V}_j, \bar{E}_j, \bar{\alpha}_j, \bar{\beta}_j)$ are the hierarchical abstractions of g_i and g_j, respectively, both based on the partition C. In Section 12.4, we will consider not only pairs, but whole sets of graphs $G = \{g_1, \ldots, g_m\}$, and compute the distance of various pairs of graphs from set G. For reasons of efficiency, it is advantageous to consider one global partitioning C for all graphs from G. Otherwise, if individual partitionings are applied, not all pairs $\bar{g}_i$ and $\bar{g}_j$ will be comparable under the considered distance measures.

The first distance measure is defined as follows:

$$D_l(\bar{g}_i, \bar{g}_j) = \sum_{v \in \bar{V}} |\text{nodes}_i(v) - \text{nodes}_j(v)| + \sum_{v \in \bar{V}} |\text{edges}_i(v) - \text{edges}_j(v)| + \sum_{e \in \bar{E}} |\bar{\beta}_i(e) - \bar{\beta}_j(e)| \,. \tag{12.2}$$

Example 12.3. A graph g_j and its hierarchical abstraction $\bar{g}_j$ are shown in Figure 12.2. We assume that $V = V_i \cup V_j$, $E = E_i \cup E_j$ and $C = \{\{1, 2\}, \{3, 4\}, \{5, 6\}\}$. It is easy to verify that $d(g_i, g_j) = 13$ and $D_l(\bar{g}_i, \bar{g}_j) = 9$. The distance $D_l(\bar{g}_i, \bar{g}_j)$ is obtained by summing the absolute differences of all pairs of corresponding attribute values. For *nodes* we get the value two, for *edges* the value three, and for $\bar{\beta}(e)$ the value four.

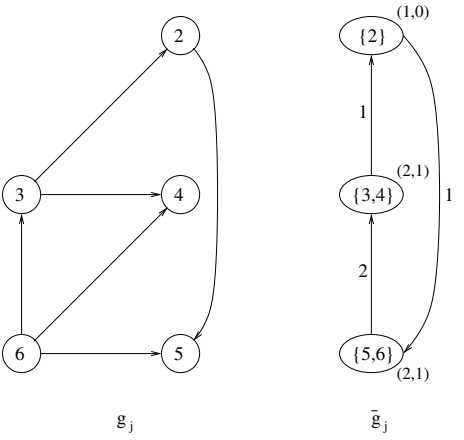

Fig. 12.2. Another graph g_j, and its hierarchical abstraction $\bar{g}_j$.

The fact that $D_l(\bar{g}_i, \bar{g}_j) \leq d(g_i, g_j)$ is not a coincidence. It can be easily proven that $D_l(\bar{g}_i, \bar{g}_j)$ is a lower bound of $d(g_i, g_j)$ for any pair of graphs g_i and g_j, and any partitioning C.

Lemma 12.4. *Let g_i, g_j, $\bar{g}_i$ and $\bar{g}_j$ be graphs as introduced above. Then*

$$D_l(\bar{g}_i, \bar{g}_j) \leq d(g_i, g_j).$$

Proof. The proof is based on the observation that the term $|V_i| + |V_j| - 2|V_i \cap V_j|$ in equation (12.1) is equal to the number of nodes that are in either g_i or g_j, but not in both. Similarly, $|E_i| + |E_j| - 2|E_i \cap E_j|$ is equal to the number of edges either in g_i or g_j, but not in both. In equation (12.2), node v (corresponding to one of the clusters c_k) includes exactly $nodes_i(v)$ nodes from g_i and $nodes_j(v)$ nodes from g_j. Hence there must be at least $|nodes_i(v) - nodes_j(v)|$ nodes that are not in both g_i and g_j. Summing up over all nodes $v \in \bar{V}$ (i.e., clusters $c_k \in C$) yields a lower bound of the expression $|V_i| + |V_j| - 2|V_i \cap V_j|$. Similarly, the sum of the second and the third terms in equation (12.2) yields a lower bound of $|E_i| + |E_j| - 2|E_i \cap E_j|$.

It can be shown that under certain conditions the lower bound given by equation (12.2) is exact.

Lemma 12.5. *Let g_i, g_j, $\bar{g}_i$, and $\bar{g}_j$ be the same as in Lemma 12.4. Furthermore, let $V_i \subseteq V_j$ and $E_i \subseteq E_j$. Then*

$$D_l(\bar{g}_i, \bar{g}_j) = d(g_i, g_j).$$

Proof. From our assumptions it follows that $|V_i \cap V_j| = |V_i|$ and $|E_i \cap E_j| = |E_i|$. Hence $|V_i| + |V_j| - 2|V_i \cap V_j| = |V_j| - |V_i|$, $|E_i| + |E_j| - 2|E_i \cap E_j| = |E_j| - |E_i|$, and $d(g_i, g_j) = |V_j| - |V_i| + |E_j| - |E_i|$. Obviously, the right-hand side of this equation is identical to the right-hand side of equation (12.2) under the assumption $|V_i| \subseteq |V_j|$ and $|E_i| \subseteq |E_j|$.

The second graph distance measure is defined as follows:

$$\begin{aligned} D_u(\bar{g}_i, \bar{g}_j) = &\sum_{v \in \bar{V}} \text{NODES}(v) + \sum_{v \in \bar{V}} \text{INTRACLUSTER-EDGES}(v) \\ &+ \sum_{e \in \bar{E}} \text{INTERCLUSTER-EDGES}(e)\,, \end{aligned} \tag{12.3}$$

where

$$\text{NODES}(v) = \begin{cases} nodes_i(v) + nodes_j(v), & \text{if } nodes_i(v) \\ & \quad + nodes_j(v) < |c|, \\ 2|c| - nodes_i(v) - nodes_j(v), & \text{otherwise}\,, \end{cases}$$

$$\text{INTRACLUSTER-EDGES}(v) = \begin{cases} edges_i(v) + edges_j(v), & \text{if } edges_i(v) + edges_j(v) \\ & \quad < |\text{EDGES}(v)|, \\ 2|\text{EDGES}(v)| \text{-} edges_i(v) - edges_j(v), & \text{otherwise}\,, \end{cases}$$

and

$$\text{INTERCLUSTER-EDGES}(e) = \begin{cases} \bar{\beta}_i(e) + \bar{\beta}_j(e), & \text{if } \bar{\beta}_i(e) + \bar{\beta}_j(e) < |\text{EDGES}(e)|, \\ 2|\text{EDGES}(e)| - \bar{\beta}_i(e) - \bar{\beta}_j(e), & \text{otherwise}\,. \end{cases}$$

In this definition, c denotes the cluster that corresponds to node v, EDGES(v) is the set of all edges in set E that belong to cluster c, and EDGES(e) is the set of all edges in E that start and end at the same cluster as edge e. Formally,

$$\text{EDGES}(v) = \{e \mid e = (x, y) \in E \wedge x \in c \wedge y \in c\},$$

and

$$\text{EDGES}(e) = \{(x, y) \mid (x, y) \in E \wedge x \in c_i \wedge y \in c_j \wedge e = (c_i, c_j)\}.$$

Example 12.6. For the graphs shown in Figures 12.1 and 12.2, we obtain $D_u(\bar{g}_i, \bar{g}_j) = 13$. Note that in all of the quantities NODES(v), INTRACLUSTER-EDGES(v), and INTERCLUSTER-EDGES(e) the second condition always evaluates to true. The first term in equation (12.3) evaluates to two, while values five and six are obtained for the second and third terms, respectively.

Next we show that the measure $D_u(\bar{g}_i, \bar{g}_j)$ is an upper bound on $d(g_i, g_j)$.

Lemma 12.7. *Let g_i, g_j, $\bar{g}_i$ and $\bar{g}_j$ be graphs as introduced above. Then*

$$d(g_i, g_j) \leq D_u(\bar{g}_i, \bar{g}_j).$$

Proof. If the number of nodes of V that belong to cluster c is greater than the number of nodes of g_i in cluster c plus the number of nodes of g_j in cluster c, then the intersection of nodes of g_i and g_j is possibly empty and the expression $|V_i| + |V_j| - 2|V_i \cap V_j|$ in equation (12.1) is bounded from above by $nodes_i(c) + nodes_j(c)$. Otherwise, some nodes from g_i and g_j must be the same, i.e., some nodes must occur in both g_i and g_j. The number of these nodes is equal to $nodes_i(c)+nodes_j(c)-|c|$. Hence the expression $|V_i| + |V_j| - 2|V_i \cap V_j|$ becomes equal to $|nodes_i(c) + nodes_j(c) - 2(nodes_i(c) + nodes_j(c) - |c|)| = 2|c| - nodes_i(c) - nodes_j(c)$. A similar argument holds for the edges, i.e., for the the attributes $edges_i(c)$, $edges_j(c)$, $\bar{\beta}_i(e)$, and $\bar{\beta}_j(e)$. Summing over all clusters c and all edges in E provides an upper bound of $d(g_i, g_j)$.

In Example 12.6 we note that $D_u(\bar{g}_i, \bar{g}_j) = d(g_i, g_j)$. This is no coincidence because the proof of Lemma 12.7 implies that the upper bound $D_u(\bar{g}_i, \bar{g}_j)$ is equal to the actual distance $d(g_i, g_j)$ if $|V_i| + |V_j| \geq |V|$ and $|E_i| + |E_j| \geq |E|$. This is summarized in the following lemma.

Lemma 12.8. *Let g_i, g_j $\bar{g}_i$, and $\bar{g}_j$ be defined as in Lemma 12.7 and let $|V_i|+|V_j| \geq |V|$ and $|E_i| + |E_j| \geq |E|$. Then*

$$D_u(\bar{g}_i, \bar{g}_j) = d(g_i, g_j).$$

A consequence of this lemma is that for any two graphs g_i, g_j and their hierarchical abstractions $\bar{g}_i$, $\bar{g}_j$, the quantity $D_u(\bar{g}_i, \bar{g}_j)$ is always equal to $d(g_i, g_j)$ if we set $V = V_i \cup V_j$ and $E = E_i \cup E_j$.

In the remainder of this section we will investigate the problem of how the upper and lower bounds $D_u(\bar{g}_i, \bar{g}_j)$ and $D_l(\bar{g}_i, \bar{g}_j)$ depend on the way we partition the set V. Let $C = \{c_1, \ldots, c_n\}$ and $\bar{C} = \{\bar{c}_1, \ldots, \bar{c}_m\}$ be two different partitionings of set V. We call C *finer* than $\bar{C}$ if for each c_i there exists a $\bar{c}_j$ such that $c_i \subseteq \bar{c}_j$. Let g_i and g_j be two graphs, $\bar{g}_i$ and $\bar{g}_j$ their hierarchical abstractions based on partition C, and $\bar{G}_i$ and $\bar{G}_j$ their hierarchical abstractions based on partition $\bar{C}$, where C is finer than $\bar{C}$. Then we can prove that $D_u(\bar{g}_i, \bar{g}_j)$ and $D_l(\bar{g}_i, \bar{g}_j)$ are better approximations of $d(g_i, g_j)$ than $D_u(\bar{G}_i, \bar{G}_j)$ and $D_l(\bar{G}_i, \bar{G}_j)$, respectively.

Lemma 12.9. *Let $\bar{g}_i$, $\bar{g}_j$, $\bar{G}_i$, and $\bar{G}_j$ be as defined above. Then*

$$D_l(\bar{G}_i, \bar{G}_j) \leq D_l(\bar{g}_i, \bar{g}_j) .$$

Proof. Assume that the cluster $c \in \bar{C}$ is split into clusters $c_1, \ldots, c_k \in C$ when we refine the partition $\bar{C}$ to the partition C. Clearly, $|c| = \sum_{l=1}^{k} |c_l|$. The contribution of cluster c to the first term of $D_l(\bar{G}_i, \bar{G}_j)$ is equal to $|nodes_i(c) - nodes_j(c)|$, which can be rewritten as $|\sum_{l=1}^{k} nodes_i(c_l) - \sum_{l=1}^{k} nodes_j(c_l)|$; see equation (12.2). On the other hand, for clusters $c_1, \ldots, c_k$ we get a contribution equal to $\sum_{l=1}^{k} |nodes_i(c_l) - nodes_j(c_l)|$ to the first term in $D_l(\bar{g}_i, \bar{g}_j)$. Applying a similar argument to the second and third terms in equation (12.2) and using the well-known relation $|\sum_{l=1}^{k} a_l - \sum_{l=1}^{k} b_l| \leq \sum_{l=1}^{k} |a_l - b_l|$, which holds for any set of real numbers a_l, b_l, concludes the proof.

Lemma 12.10. *Let $\bar{g}_i$, $\bar{g}_j$, $\bar{G}_i$, and $\bar{G}_j$ be the same as in Lemma 12.9. Then*

$$D_u(\bar{g}_i, \bar{g}_j) \leq D_u(\bar{G}_i, \bar{G}_j) .$$

Proof. The proof is based on observing that in the computation of NODES(v) in equation (12.3), whenever the second case evaluates to true for a partition $\bar{C}$, it will also evaluate to true for any other partition C that is finer than $\bar{C}$. On the other hand, if the first case evaluates to true for $\bar{C}$, then either the first or the second case may evaluate to true for any of the clusters in C. Moreover, we observe that the value under the second condition is always less than or equal to the value obtained under the first condition. Applying a similar argument to INTRACLUSTER-EDGES(v) and INTERCLUSTER-EDGES(e) yields the proof.

Summarizing all results derived in this section, we obtain the following theorem:

Theorem 12.11. *Let all quantities be as introduced above. Then:*

(i) $D_l(\bar{G}_i, \bar{G}_j) \leq D_l(\bar{g}_i, \bar{g}_j) \leq d(g_i, g_j) \leq D_u(\bar{g}_i, \bar{g}_j) \leq D_u(\bar{G}_i, \bar{G}_j)$,
(ii) $d(g_i, g_j) = D_l(\bar{g}_i, \bar{g}_j) = D_l(\bar{G}_i, \bar{G}_j)$, *if* $V_i \subseteq V_j$ *and* $E_i \subseteq E_j$,
(iii) $d(g_i, g_j) = D_u(\bar{g}_i, \bar{g}_j) = D_u(\bar{G}_i, \bar{G}_j)$, *if* $V = V_i \cup V_j$, $E = E_i \cup E_j$.

We notice that whenever the condition in (ii) is satisfied, the condition in (iii) will also be satisfied. Hence in this case $D_l(\bar{g}_i, \bar{g}_j) = D_l(\bar{G}_i, \bar{G}_j) = d(g_i, g_j) = D_u(\bar{g}_i, \bar{g}_j) = D_u(\bar{G}_i, \bar{G}_j)$.

12.4 Application to Computer Network Monitoring

Through the hierarchical abstraction process described in Section 12.2, the number of nodes in a graph can be reduced. In fact, it can be made arbitrarily small. In the extreme case, a large graph will be represented by a single node only. Applying equations (12.2) and (12.3) to a pair of graphs $\bar{g}_i, \bar{g}_j$, which result from g_i and g_j through the proposed abstraction process, yields upper and lower bounds for $d(g_i, g_j)$. The closer $\bar{g}_i$ and $\bar{g}_j$ are to the original, full-resolution graphs g_i and g_j, i.e., the more details are included in the abstract graph representation, the closer will be the upper and lower bounds to the actual value $d(g_i, g_j)$. Note that only two numbers, the number of corresponding nodes and the number of corresponding edges from the original graph, need to be stored with a node at an abstract level. For edges at an abstract level, only one number is needed, representing the number of corresponding edges in the original graph.

If an abnormal event at time $t+1$ is defined by the condition $d(g_t, g_{t+1}) \geq \theta$, where θ is a threshold that depends on the considered application and the underlying network, then rather than considering $d(g_t, g_{t+1})$ one can compute $D_l(\bar{g}_t, \bar{g}_{t+1})$ and $D_u(\bar{g}_t, \bar{g}_{t+1})$, where $\bar{g}_t$ and $\bar{g}_{t+1}$ are obtained from g_t and g_{t+1} through the proposed graph abstraction procedure. Clearly, if $D_l(\bar{g}_t, \bar{g}_{t+1}) \geq \theta$ then we conclude that an abnormal event has occurred. Similarly, if $D_u(\bar{g}_t, \bar{g}_{t+1}) < \theta$ then we conclude that no abnormal event has occurred. In either case we need not compute $d(g_t, g_{t+1})$, and it can be expected that computing $D_l(\bar{g}_t, \bar{g}_{t+1})$ and $D_u(\bar{g}_t, \bar{g}_{t+1})$ is faster than the computation of $d(g_t, g_{t+1})$. On the other hand, if $D_l(\bar{g}_t, \bar{g}_{t+1}) < \theta$ and $D_u(\bar{g}_t, \bar{g}_{t+1}) \geq \theta$, then we need to calculate $d(g_t, g_{t+1})$. Alternatively we can compute $D_l(\tilde{g}_t, \tilde{g}_{t+1})$ and $D_u(\tilde{g}_t, \tilde{g}_{t+1})$ for graphs $\tilde{g}_t$ and $\tilde{g}_{t+1}$ that are closer to the original level of resolution than $\bar{g}_t$ and $\bar{g}_{t+1}$, expecting either $D_l(\tilde{g}_t, \tilde{g}_{t+1}) \geq \theta$ or $D_u(\tilde{g}_t, \tilde{g}_{t+1}) < \theta$.

In Chapter 7 (intra)graph clustering algorithms have been described. These algorithms are able to identify clusters of nodes within a graph such that nodes in the same cluster are similar to each other, while nodes in different clusters are dissimilar. For the graph abstraction process described in Section 12.2, no such clustering algorithm is needed. In fact, any partition of the underlying set of nodes can be used as the basis of graph abstraction. In the simplest case, one can just assign unique labels from 1 to N to all servers in the underlying network, and then partition the set $\{1, \ldots, N\}$ into a given number of disjoint subsets.

Computing $D_l(\bar{g}_t, \bar{g}_{t+1})$ and $D_u(\bar{g}_t, \bar{g}_{t+1})$, we get lower and upper bounds of $d(g_t, g_{t+1})$, respectively, as discussed before. Note that in the case that all nodes and edges of the entire network appear in the union g_t and g_{t+1}, the upper bound $D_u(\bar{g}_t, \bar{g}_{t+1})$ is the exact value. In this case two arbitrarily large graphs g_t and g_{t+1} can be contracted to a single node each, and the upper bound will still be the exact value, i.e., $D_u(\bar{g}_t, \bar{g}_{t+1}) = d(g_t, g_{t+1})$. As an example consider the graphs in Figures 12.1 and 12.2. If we contract g_i into a single node v, we get $nodes_i(v) = 5, edges_i(v) = 7$. Similarly, if g_j is contracted into a single node, then $nodes_j(v) = 5, edges_j(v) = 6$. Moreover, under the assumption that the union of g_i and g_j includes the entire set of nodes and edges of the network, we observe that $|c| = 6$ and $|\text{EDGES}(v)| = 12$. Hence $D_u(\bar{g}_i, \bar{g}_j) = (12 - 10) + (24 - 13) = 13 = d(g_i, g_j)$.

12.5 Experimental Results

The aim of the experiments described in this section is to verify the theoretical results derived in Section 12.3, to measure the tightness of upper and lower bounds, and to quantitatively evaluate the computational savings that can be achieved through the proposed graph abstraction scheme. In the experiments described in this section, we first generate a graph g_1, with 10,000 nodes. Each node is connected, on average, to 1,000 other nodes via an undirected, unlabeled edge. Edges are randomly distributed in g_1. The set of integers $\{1, \ldots, 10{,}000\}$ is used to label the nodes and each node has a unique label. A second graph g_2 is obtained from g_1 by randomly deleting $n\%$ of the nodes of g_1, together with their incident edges. Additionally $n'\%$ of the remaining edges are deleted. Next, hierarchical abstractions of both g_1 and g_2 are generated, consisting of 1,000, 100, 10, and 1 node. These hierarchical abstractions are all based on the same partition of the nodes of g_1. For example, to generate a hierarchical abstraction with 1,000 nodes, i.e., with a cluster size of ten nodes each, the first cluster is given by the nodes with a label from $\{1, \ldots, 10\}$, the second cluster by the nodes with a label from $\{11, \ldots, 20\}$, and so on. From the way g_1 and g_2 are generated, it is obvious that the conditions of Lemmas 12.5 and 12.8 are fulfilled. Hence, we expect that $D_l(\bar{g}_1, \bar{g}_2) = d(g_1, g_2) = D_u(\bar{g}_1, \bar{g}_2)$ for any of the considered hierarchical abstractions $\bar{g}_i$. As a matter of fact, this expectation is confirmed in Figure 12.3, where the x-axis corresponds to the different levels of abstraction (i.e., number of clusters, which is 1,000, 100, 10, 1), and the y-axis represents the distances $d(g_1, g_2)$, $D_l(\bar{g}_1, \bar{g}_2)$, $D_u(\bar{g}_1, \bar{g}_2)$. All three distances coincide for any considered level of abstraction, which confirms that both upper and lower bounds are identical to the real graph distance. In Figure 12.3, values $n = 10$ and $n' = 5$ are used. In Figure 12.4, the results of four similar experimental runs are shown for values $(n = 20, n' = 10)$, $(n = 30, n' = 15)$, $(n = 40, n' = 20)$, and $(n = 50, n' = 25)$. In each case the values of $d(g_1, g_2)$, $D_l(\bar{g}_1, \bar{g}_2)$, and $D_u(\bar{g}_1, \bar{g}_2)$ coincide. Hence only four straight lines are observed in this figure. Clearly, with an increasing number of nodes and edges being deleted from g_1, the distance between g_1 and g_2 increases. This effect can be clearly observed in Figure 12.4. The point to be stressed about Figure 12.4 is that, similarly to Figure 12.3, all three measures $D_l(\bar{g}_1, \bar{g}_2)$, $d(g_1, g_2)$, and $D_u(\bar{g}_1, \bar{g}_2)$ are identical, as stated in Lemmas 12.5 and 12.8.

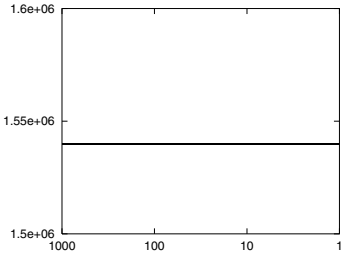

Fig. 12.3. Experimental data illustrating Lemmas 12.5 and 12.8.

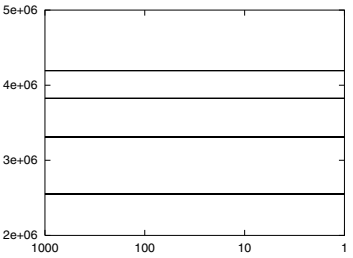

Fig. 12.4. Further illustration of Lemmas 12.5 and 12.8.

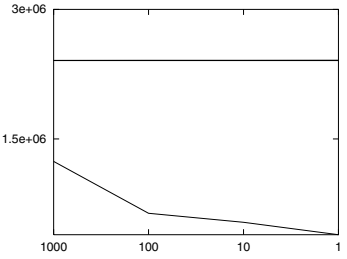

Fig. 12.5. Experimental data illustrating the upper and lower bound ($m = 60$).

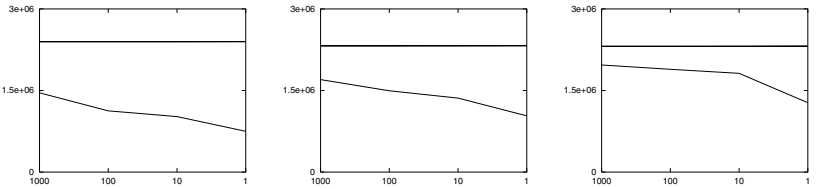

Fig. 12.6. Further illustration of upper and lower bound: (a) $m = 70$, (b) $m = 80$, (c) $m = 90$.

The aim of the next set of experiments is to analyze the behavior of the upper and lower bounds in case the conditions of Lemmas 12.5 and 12.8 are no longer satisfied. For this purpose, we start again with a graph g_1 that is generated in exactly the same way as described in the previous paragraph. Next we randomly delete 50% of the nodes of g_1 together with their incident edges. The resulting graph is referred to as g_3. Next, graph g_4 is generated by randomly deleting $m\%$ of all nodes together with their incident edges

Table 12.1. Data corresponding to Figure 12.5

	1000	100	10	1	m
D_u	2,409,516	2,410,042	2,410,088	2,411,390	
D_l	1,239,038	638,308	534,130	390,094	60
d			2,407,580		

Table 12.2. Data corresponding to Figure 12.6.

	1000	100	10	1	m
D_u	2,397,782	2,398,060	2,398,146	2,400,294	
D_l	1,457,062	1,126,264	1,020,270	749,790	70
d			2,396,262		
D_u	2,322,436	2,322,510	2,322,814	2,325,836	
D_l	1,700,350	1,495,504	1,360,116	1,034,514	80
d			2,321,440		
D_u	2,313,272	2,313,318	2,313,494	2,317,408	
D_l	1,969,420	1,890,110	1,815,788	1,275,640	90
d			2,312,800		

from g_1 ($m = 60, 70, 80, 90$). Clearly, when we match graphs g_3 and g_4, the conditions of Lemmas 12.5 and 12.8 are not necessarily satisfied any longer. Similarly to the first set of experiments, hierarchical abstractions of g_3 and g_4 were generated consisting of 1,000, 100, 10, and 1 node. In Figure 12.5, the distances $d(g_3, g_4)$, $D_l(\bar{g}_3, \bar{g}_4)$, and $D_u(\bar{g}_3, \bar{g}_4)$ are shown for $m = 60$. While the lower bound is significantly smaller than the real distance, the upper bound is quite tight. As a matter of fact, $d(g_3, g_4)$ visually coincides with $D_u(\bar{g}_3, \bar{g}_4)$ in Figure 12.5. To see that $d(g_3, g_4)$ is not identical to $D_u(\bar{g}_3, \bar{g}_4)$, the information provided in Figure 12.5 is shown in tabular form in Table 12.1. In Figure 12.6 and Table 12.2 the corresponding values are given for $m = 70$, 80, and 90. As m increases, graphs g_3 and g_4 become more similar to each other. In any case, the upper bound is very close to the real distance even for the maximum degree of compression, where both graphs are represented through a single node only. We also observe that both upper and lower bounds become tighter as the distance $d(g_3, g_4)$ decreases.

The motivation of the third set of experiments is to measure the computational savings that can be achieved by means of the proposed hierarchical graph abstraction scheme. We assume that the sensors, or devices, that yield the graph data not only provide us with the graphs at the full level of resolution, but also with hierarchical abstractions. Hence the time needed to generate hierarchical abstractions from a graph at the full resolution level is not taken into account in the experiments described in the following. To analyze the computational efficiency of the proposed graph similarity measures, we select graphs g_3 and g_4 (with $m = 60$) and their hierarchical abstractions, as described in the last paragraph, and measure the time needed to compute $d(g_3, g_4)$, $D_l(\bar{g}_3, \bar{g}_4)$, and $D_u(\bar{g}_3, \bar{g}_4)$. The results are shown in Table 12.3. The computation of $d(g_3, g_4)$ is performed on the original graphs g_3 and g_4, and is independent of the cluster

size in the hierarchical abstraction. It turns out that the computation of both $D_l(\bar{g}_3, \bar{g}_4)$ and $D_u(\bar{g}_3, \bar{g}_4)$ is extremely fast when compared to $d(g_3, g_4)$. From this observation we can conclude that distance measure D_u provides an excellent compromise between speed and precision. On one hand, it is extremely fast to compute, and on the other, it returns values very close to the real graph distance. As a matter of fact, a speedup on the order of 10^8 can be observed over the computation of $d(g_3, g_4)$ for the case of maximum graph compression, while the precision of the upper bound is still within a tolerance of 0.2%.

Table 12.3. Computational time of the distance measures in msec.

	1000	100	10	1	m
Time D_u	61.6	0.5632	0.00568	0.000076	
Time D_l	19.52	0.1552	0.001304	0.0000552	60
Time d			36,000		

12.6 Conclusions

In this chapter we have described a hierarchical graph abstraction procedure that contracts clusters, or groups, of nodes into single nodes. On this hierarchical representation, graph similarity can be computed more efficiently than on the original graphs. Two distance measures for contracted graphs are introduced, and it is shown that they provide lower and upper bounds, respectively, for the distance of graphs at the original level of resolution. The proposed methods can be used to very significantly speed up the computation of graph similarity in the context of computer network monitoring and abnormal change detection. It can be proven that under special conditions, upper and/or lower bounds are exact.