Pattern Anal Applic (2004) 7: 243–254
DOI 10.1007/s10044-004-0222-5

THEORETICAL ADVANCES

**Peter J. Dickinson · Horst Bunke · Arek Dadej**
**Miro Kraetzl**

# Matching graphs with unique node labels

Received: 11 February 2004 / Accepted: 26 May 2004 / Published online: 6 October 2004


**Abstract** A special class of graphs is introduced in this paper. The graphs belonging to this class are characterised by the existence of unique node labels. A number of matching algorithms for graphs with unique node labels are developed. It is shown that problems such as graph isomorphism, subgraph isomorphism, maximum common subgraph (MCS) and graph edit distance (GED) have a computational complexity that is only quadratic in the number of nodes. Moreover, computing the median of a set of graphs is only linear in the cardinality of the set. In a series of experiments, it is demonstrated that the proposed algorithms run very fast in practice. The considered class makes the matching of large graphs, consisting of thousands of nodes, computationally tractable. We also discuss an application of the considered class of graphs and related matching algorithms to the classification and detection of abnormal events in computer networks.

**Keywords** Graph matching · Graph isomorphism · Maximum common subgraph · Graph edit distance · Median graph · Unique node label

P. J. Dickinson · M. Kraetzl
ISR Division, DSTO, PO Box 1500,
Edinburgh, SA, 5111, Australia
E-mail: peter.dickinson@dsto.defence.gov.au
E-mail: miro.kraetzl@dsto.defence.gov.au

H. Bunke (✉)
Institut für Informatik und angewandte Mathematik,
Universität Bern, Neubrückstrasse 10,
CH-3012 Bern, Switzerland
E-mail: bunke@iam.unibe.ch

A. Dadej
ITR, University of South Australia,
Mawson Lakes, SA, 5095, Australia
E-mail: arek.dadej@unisa.edu.au

## 1 Introduction

Graph matching has become a very active field of research [1–3]. In its most general form, graph matching refers to the problem of finding a mapping $f$ from the nodes of one given graph $g_1$ to the nodes of another given graph $g_2$, that satisfies some constraints or optimality criteria. For example, in graph isomorphism detection [4], mapping $f$ is a bijection that preserves all edges and labels. In subgraph isomorphism detection [5], mapping $f$ has to be injective such that all edges of $g_1$ are included in $g_2$ and all labels are preserved. Other graph matching problems that require the construction of a mapping $f$ with particular properties are maximum common subgraph (MCS) detection [6, 7] and graph edit distance (GED) computation [8, 9].

The main problem with graph matching is its high computational complexity, which arises from the fact that it is usually very costly to find mapping $f$ for a pair of given graphs. It is a known fact that the detection of a subgraph isomorphism or a maximum common subgraph (MCS), as well as the computation of GED are NP-complete problems. If the graphs in the application are small, optimal algorithms can be used. These algorithms are usually based on an exhaustive enumeration of all possible mappings $f$ between two graphs. Sometimes, application-dependent heuristics can be found that allow us to eliminate significant portions of the search space (i.e. the space of all possible functions $f$), but still guarantee the correct, or optimal, solution being found. Such heuristics can be used in conjunction with look-ahead techniques and constraint satisfaction [5, 10, 11]. For the matching of large graphs, one needs to resort to suboptimal matching strategies. Methods of this type are characterised by a (often low-order) polynomial time complexity, but they are no longer guaranteed to find the optimal solution for a given problem. A large variety of such suboptimal approaches have been proposed in the literature, based on a multitude of different computational paradigms. Examples include probabi-

listic relaxation [12, 13], genetic algorithms [14, 15], expectation maximisation [16], eigenspace methods [17, 18] and quadratic programming [19].

Another possibility to overcome the problem arising from the exponential complexity of graph matching is to focus on classes of graphs with an inherently lower computational complexity of the matching task. Some examples of such classes are given in [20–22]. Most recently, in the field of pattern recognition and computer vision, the class of trees has received considerable attention [23, 24].

In this paper, another special class of graphs will be introduced. The graphs belonging to this class are characterised by the existence of unique node labels, which means that each node in a graph possesses a node label that is different from all other node labels in that graph. This condition implies that, whenever two graphs are being matched with each other, each node has at most one candidate for possible assignment under function $f$ in the other graph. This candidate is uniquely defined through its node label. Consequently, the most costly step in graph matching, which is the exploration of all possible mappings between the nodes of the two graphs under consideration, is no longer needed. Moreover, we introduce matching algorithms for this special class of graphs and analyse their computational complexity. Particular attention is directed to the computation of graph isomorphism, subgraph isomorphism, MCS, GED and median graph computation.

If constraints are imposed on any class of graphs, we usually lose some representational power. The class of graphs considered in this paper is restricted by the requirement of each node label being unique. Despite this restriction, there exist some interesting applications for this class of graphs. From the general point of view, graphs with unique node labels seem to be appropriate whenever the objects from the problem domain, which are modelled through nodes, possess properties that can be used to uniquely identify them. We review one particular application of this class of graphs in the domain of computer network monitoring. Another application of graphs with unique node labels is Web document analysis. In this case, each unique term (word) that occurs in a document is represented by a node. Multiple occurrences of a term are represented by the same node. In this application, the considered class of graphs has been used for the tasks of classification and clustering of Web pages, and has shown superior performance over traditional vector-based approaches. For further details, see [25–27].

The remainder of this paper is organised as follows. In Sect. 2, we introduce our basic concepts and terminology. Graphs with unique node labels and related matching strategies are presented in Sect. 3. Potential applications of this class of graphs are discussed in Sect. 4. In Sect. 5, we present the results of an experimental study where the run time of some of the proposed algorithms was measured. Finally, conclusions from this work are drawn in Sect. 6. An earlier version of this paper appeared in [28]. The present paper has been significantly extended in both theory and experimental evaluation.

## 2 Basic concepts and notation

In this section, the basic concepts and terminology used throughout the paper will be introduced. We consider directed graphs with labelled vertices (nodes) and edges (links). Let $L_V$ and $L_E$ denote the sets of node and edge labels, respectively. A *graph* $g=(V, E, \alpha, \beta)$ is a 4-tuple where $V$ is the finite set of *vertices*, $E \subseteq V \times V$ is the set of *edges*, $\alpha$: $V \rightarrow L_V$ is a function assigning labels to the nodes and $\beta$: $E \rightarrow L_E$ is a function assigning labels to edges. Edge $(x, y) \in E$ originates at node $x \in V$ and terminates at node $y \in V$. An undirected graph is obtained as a special case if there exists an edge $(y, x) \in E$ for every edge $(x, y) \in E$ with $\beta(x, y) = \beta(y, x)$.

Let $g=(V, E, \alpha, \beta)$ and $g'=(V', E', \alpha', \beta')$ be graphs; $g'$ is a *subgraph* of $g$, $g' \subseteq g$ if $V' \subseteq V$, $E' \subseteq E$, $\alpha(x) = \alpha'(x)$ for all $x \in V'$ and $\beta(x, y) = \beta'(x, y)$ for all $(x, y) \in E'$. Let $g \subseteq g'$ and $g \subseteq g''$. Then, $g$ is called a *common subgraph* of $g'$ and $g''$. Furthermore, $g$ is called a *maximum common subgraph* (notation: MCS) of $g'$ and $g''$ if there exists no other common subgraph of $g'$ and $g''$ that has more nodes and, for a given number of nodes, more edges than $g$.

For graphs $g$ and $g'$, a *graph isomorphism* is any bijection $f$: $V \rightarrow V'$ such that:

1. $\alpha(x) = \alpha'(x)$ for all $x \in V$; and
2. For any edge $(x, y) \in E$, there exists $(f(x), f(y)) \in E'$ with $\beta(x, y) = \beta'(f(x), f(y))$, and for any edge $(x', y') \in E'$, there exists an edge $(f^{-1}(x'), f^{-1}(y')) \in E$ with $\beta'(x', y') = \beta(f^{-1}(x'), f^{-1}(y'))$.

If $f$: $V \rightarrow V'$ is a graph isomorphism between graphs $g$ and $g'$, and $g'$ is a subgraph of another graph $g''$, i.e. $g' \subseteq g''$, then $f$ is called a *subgraph isomorphism* from $g$ to $g''$.

Next, we introduce the concept of GED, which is based on graph edit operations. We consider six types of edit operations: substitution of a node label, substitution of an edge label, insertion of a node, insertion of an edge, deletion of a node and deletion of an edge. A cost (i.e. a non-negative real number) is assigned to each edit operation. Let $e$ be an edit operation and $c(e)$ its cost. The *cost of a sequence of edit operations*, $s = e_1, \ldots, e_n$, is given by the sum of all its individual costs, i.e. $c(s) = \sum_{i=1}^{n} c(e_i)$. The *edit distance* $d(g_1, g_2)$ of two graphs $g_1$ and $g_2$ is equal to the *minimum cost*, taken over all sequences of edit operations, that transform $g_1$ into $g_2$. Procedures for GED computation are discussed in [8].

Finally, we introduce the median of a set of graphs [29]. Let $G = \{g_1, \ldots, g_N\}$ be a set of graphs and $U$ be the set of all graphs with labels from $L_V$ and $L_E$. The *median* $\bar{g}$ of $G$ is a graph that satisfies the condition:

$$\sum_{i=1}^{N} d(\bar{g}, g_i) = \min \left\{ \sum_{i=1}^{N} d(g, g_i) | g \in U \right\}$$

It follows that the median is a graph which has the minimum average edit distance to the graphs in set $G$. It is a useful concept to represent a set of graphs by a single prototype. Intuitively, the median can be understood as an optimal representative of a set of graphs since it has, in the universe of all graphs, the smallest average distance to the given graphs. In many instances, the median of a given set $G$ is not unique, neither is it always a member of $G$. The use of the term "median graph" seems appropriate because of its analogy to the median of a set of numbers; both minimise the average absolute difference to the members in the set.

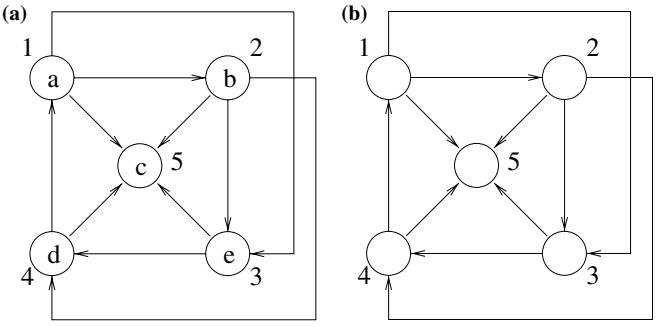


**Fig. 1a, b** Example graph $g$ and its label representation. **a** Example graph $g$. **b** Label representation of $g$

## 3 Graphs with unique node labels

In this section, we introduce a special class of graphs that are characterised by the existence of unique node labels. Formally, we require that, for any graph $g$ and any pair $x, y \in V$, the condition $\alpha(x) \neq \alpha(y)$ holds if $x \neq y$. Furthermore, we assume that the underlying alphabet of node labels is an ordered set, for example, the integers, i.e. $L_V = \{1, 2, 3, \ldots\}$. Throughout the rest of this paper, we consider graphs from this class only, unless otherwise mentioned.

*Definition 1* Let $g = (V, E, \alpha, \beta)$ be a graph. The label representation $\rho(g)$ of $g$ is given by $\rho(g) = (L, C, \lambda)$, where:

1. $L = \{\alpha(x) | x \in V\}$
2. $C = \{(\alpha(x), \alpha(y)) | (x, y) \in E\}$; and
3. $\lambda$: $C \rightarrow L_E$ with $\lambda(\alpha(x), \alpha(y)) = \beta(x, y)$ for all $(x, y) \in E$.

According to this definition, the label representation of a graph $g$ is obtained by representing each node of $g$ by its (unique) label and dropping set $V$. From the formal point of view, $\rho(g)$ defines the equivalence class of all graphs that are isomorphic to $g$. The individual members of this class are obtained by assigning an arbitrary node, or, more precisely, an arbitrary node name, to each unique node label, i.e. to each element from $L$.

*Example 1* Let $L_V = \{1, 2, 3, 4, 5\}$ and $g = (V, E, \alpha, \beta)$ where $V = \{a, b, c, d, e\}$, $E = \{(a, b), (b, e), (e, d), (d, a), (a, c), (b, c), (d, c), (e, c), (a, e), (b, d)\}$, $\alpha$: $a \mapsto 1$, $b \mapsto 2$, $c \mapsto 5$, $d \mapsto 4$, $e \mapsto 3$, $\beta$: $(x,y) \mapsto 1$ for all $(x, y) \in E$. A graphical illustration of $g$ is shown in Fig. 1a, where the node names (i.e. the elements of $V$) appear inside the nodes and the corresponding labels appear outside. Because all edge labels are identical, they have been omitted. The label representation $\rho(g)$ of $g$ is then given by the following quantities: $L = \{1, 2, 3, 4, 5\}$, $C = \{(1, 2), (2, 3), (3, 4), (4, 1), (1, 5), (2, 5), (4, 5), (3, 5), (1, 3), (2, 4)\}$, $\lambda : (i, j) \mapsto 1$ for all $(i, j) \in C$.

Intuitively, we can interpret the label representation $\rho(g)$ of any graph $g$, as a graph identical to $g$ up to the fact that all node names are left unspecified. Hence, $\rho(g)$ can be conveniently graphically represented in the same way as $g$ is represented. For example, a graphical representation of $\rho(g)$, where $g$ is shown in Fig. 1a, is given in Fig. 1b.

*Lemma 1* Let $g_1 = (V_1, E_1, \alpha_1, \beta_1)$, $g_2 = (V_2, E_2, \alpha_2, \beta_2)$ be two graphs and $\rho(g_1) = (L_1, C_1, \lambda_1)$, $\rho(g_2) = (L_2, C_2, \lambda_2)$ be their label representations. Graph $g_1$ is isomorphic to graph $g_2$ if and only if $\rho(g_1) = \rho(g_2)$ (i.e. $L_1 = L_2$, $C_1 = C_2$ and $\lambda_1 = \lambda_2$).

*Proof* Assume that there exists a graph isomorphism $f$: $V_1 \rightarrow V_2$. Then, $\alpha_1(x) = \alpha_2(f(x))$ for all $x \in V_1$. As $f$ is bijective, it follows that $L_1 = L_2$. Furthermore, because of the conditions on the edges that are imposed by graph isomorphism $f$, we conclude that $C_1 = C_2$ and $\lambda_1 = \lambda_2$. Conversely, assume that $\rho(g_1) = \rho(g_2)$. Construct now the mapping $f$: $V_1 \rightarrow V_2$ such that $f(x) = y$ if and only if $\alpha_1(x) = \alpha_2(y)$. Because $L_1 = L_2$ and the node labels in both $g_1$ and $g_2$ are unique, this mapping is a bijection that satisfies the conditions of graph isomorphism imposed on the edges and edge labels in $g_1$ and $g_2$.

Based on this lemma, we can examine two graphs for isomorphism by simply generating their label representations and checking the conditions $L_1 = L_2$, $C_1 = C_2$ and $\lambda_1 = \lambda_2$. Assume $n = \max\{|V_1|, |V_2|\}$. Then, $|L_1| = |L_2| = \mathcal{O}(n)$, $|E_1| = |C_1| = \mathcal{O}(n^2)$ and $|E_2| = |C_2| = \mathcal{O}(n^2)$. Testing two ordered sets for identity is an operation that is linear in the number of elements. Hence, the computational complexity of testing two graphs with unique node labels for isomorphism amounts to $\mathcal{O}(n^2)$.

*Lemma 2* Let $g_1$, $g_2$, $\rho(g_1)$ and $\rho(g_2)$ be the same as in Lemma 1. Then, $g_1$ is a subgraph isomorphic to $g_2$ if and only if $L_1 \subseteq L_2$, $C_1 \subseteq C_2$ and $\lambda_1(i, j) = \lambda_2(i, j)$ for all $(i, j) \in C_1$.

*Proof* Firstly, we assume that there exists a subgraph isomorphism $f$: $V_1 \rightarrow V_2$. Then, $\alpha_1(x) = \alpha_2(f(x))$ for all $x \in V_1$. As $f$ is injective, it follows that $L_1 \subseteq L_2$. Similarly to the proof of Lemma 1, we conclude $C_1 \subseteq C_2$ and $\lambda_1(i, j) = \lambda_2(i, j)$ for all $(i, j) \in C_1$. Conversely, assume

$L_1 \subseteq L_2$, $C_1 \subseteq C_2$ and $\lambda_1(i, j) = \lambda_2(i, j)$ for all $(i, j) \in C_1$. Then, we can construct an injective mapping $f$: $V_1 \rightarrow V_2$ such that $f(x) = y$ if and only if $\alpha_1(x) = \alpha_2(y)$. Similarly to the proof of Lemma 1, it follows that this mapping is a subgraph isomorphism from $g_1$ to $g_2$.

Using Lemma 2, testing two graphs for subgraph isomorphism reduces to examining the corresponding label representations for the three conditions $L_1 \subseteq L_2$, $C_1 \subseteq C_2$ and $\lambda_1(i, j) = \lambda_2(i, j)$ for all $(i, j) \in C_1$. The third condition can be checked in $\mathcal{O}(|C_1|) = \mathcal{O}(n^2)$ time. Checking whether an ordered set is a subset of another ordered set is linear in the size of the larger of the two sets. Hence, the computational complexity of subgraph isomorphism of graphs with unique node labels equals $\mathcal{O}(n^2)$.

*Lemma 3* Let $g_1$, $g_2$, $\rho(g_1)$ and $\rho(g_2)$ be the same as in Lemma 1. Let $g$ be a graph with $\rho(g) = (L, C, \lambda)$ such that $L = L_1 \cap L_2$, $C = \{(i, j) | (i, j) \in C_1 \cap C_2$ and $\lambda_1(i, j) = \lambda_2(i, j)\}$ and $\lambda(i, j) = \lambda_1(i, j)$ for all $(i, j) \in C$. Then, $g$ is an MCS of $g_1$ and $g_2$.

*Proof* First, we note that $L \subseteq L_1$ and $L \subseteq L_2$. Hence, $V \subseteq V_1$ and $V \subseteq V_2$. for any graph $g$ with label representation $\rho(g) = (L, C, \lambda)$. Similarly, because $C$ includes a pair $(i, j)$ if and only if a corresponding edge with identical label exists in both $g_1$ and $g_2$, we observe $E \subseteq E_1$ and $E \subseteq E_2$. for any such graph $g$. Thirdly, the labels of edges $(x, y)$ occurring in both $g_1$ and $g_2$ are preserved under $\lambda$. Hence, $g$ is a subgraph of both $g_1$ and $g_2$. Now assume that $g$ is not an MCS. In this case, there must exist another subgraph $g'$ of both $g_1$ and $g_2$ with either more nodes than $g$, or the same number of nodes, but with more edges. The first case contradicts the way set $L$ is constructed; if $g'$ has more nodes than $g$, then $L \neq L_1 \cap L_2$. The second case is in conflict with the construction of $C$ and $\lambda$, i.e. if $g'$ has the same number of nodes as $g$, but more edges, then $C$ and $\lambda$ must be different from their values stated in Lemma 3. Hence, $g$ must be indeed an MCS of $g_1$ and $g_2$. The proof follows.

Possible computational procedures implied by Lemma 3 are again based on the intersection of two ordered sets. Hence, the complexity of computing the MCS of two graphs with unique node labels is $\mathcal{O}(n^2)$.

In [30], a detailed analysis was provided showing how GED depends on the costs associated with the individual edit operations. A set of edit operations together with their cost is also called a *cost function*. In this paper, we focus our attention on the following cost function: $c_{nd}(x) = c_{ni}(x) = 1$, $c_{ns}(x) = \infty$, $c_{ed}(x, y) = c_{ei}(x, y) = c_{es}(x, y) = 1$, where $c_{nd}(x)$, $c_{ni}(x)$ and $c_{ns}(x)$ denote the cost associated with the deletion, insertion and substitution of node $x$, while $c_{ed}(x, y)$, $c_{ei}(x, y)$ and $c_{es}(x, y)$ denote the cost associated with the deletion, insertion and substitution of edge $(x, y)$, respectively. This cost function is simple in the sense that each edit operation has a cost equal to 1, except for node substitutions, which have infinite cost. It is easy to see that, for any two graphs, $g_1$ and $g_2$, there always exists a sequence of edit operations that transforms $g_1$ into $g_2$ with a finite total cost (for example, a sequence that deletes all nodes and edges from $g_1$, and inserts all nodes and edges in $g_2$). Hence, edit operations with infinite cost will never be applied in the computation of any actual GED. This means that node substitutions will never be applied and may be considered non-admissible under the cost function introduced above, while all other edit operations can be applied and have the same cost. The exclusion of node substitutions for graphs with unique node labels makes sense since node label substitutions may generate graphs with non-unique node labels, i.e. graphs that do not belong to the class of graphs under consideration.

*Lemma 4* Let $g_1$, $g_2$, $\rho(g_1)$ and $\rho(g_2)$ be the same as in Lemma 1. Furthermore, let $C_0 = \{(i, j) | (i, j) \in C_1 \cap C_2\}$ and $C_0' = \{(i, j) | (i, j) \in C_1 \cap C_2\}$ and $\lambda_1(i, j) \neq \lambda_2(i, j)$. Then $d(g_1, g_2) = |L_1| + |L_2| - 2|L_1 \cap L_2| + |C_1| + |C_2| - 2|C_0| + |C_0'|$.

*Proof* Because node substitutions can be regarded non-admissible, the minimum cost sequence of edit operations transforming $g_1$ into $g_2$ assigns each node $x \in V_1$ with label $\alpha_1(x)$ to node $y \in V_2$ with $\alpha_1(x) = \alpha_2(y)$. If no node $y \in V_2$ exists with this property, node $x$ is deleted from $g_1$. Similarly, all nodes $y \in V_2$ for which no node $x \in V_1$ exists with $\alpha_1(x) = \alpha_2(y)$ will be inserted in $g_2$. This leads to $|L_1| - |L_1 \cap L_2|$ node deletions in graph $g_1$ and $|L_2| - |L_1 \cap L_2|$ node insertions in graph $g_2$, each having a cost equal to 1. Hence, the total cost arising from edit operations on the nodes of $g_1$ and $g_2$ amounts to $|L_1| - |L_1 \cap L_2| + |L_2| - |L_1 \cap L_2| = |L_1| + |L_2| - 2|L_1 \cap L_2|$.

We now consider the edges. There exist $|C_1| - |C_0|$ edges in $g_1$ that do not occur in $g_2$, and need to be deleted. Similarly, there exist $|C_2| - |C_0|$ edges in $g_2$ that do not have a counterpart in $g_1$, and need to be inserted. Furthermore, there are two types of edges corresponding to the set $|C_1 \cap C_2|$. The first type are edges $(i, j) \in C_0$, for which $\lambda_1(i, j) = \lambda_2(i, j)$. No edit operations are needed for edges of this kind. The second type are edges $(i, j) \in C_0'$, for which $\lambda_1(i, j) \neq \lambda_2(i, j)$. An edge substitution with a cost of 1 is needed for each such edge. Hence, the total cost of the edit operations on the edges of $g_1$ and $g_2$ is equal to $|C_1| - |C_0| + |C_2| - |C_0| + |C_0'| = |C_1| + |C_2| - 2|C_0| + |C_0'|$. This concludes the proof.

Possible computational procedures for GED computation implied by Lemma 4 are based again on the intersection of two ordered sets. Hence, similarly to all other graph matching procedures considered before, the complexity of edit distance computation of graphs with unique node labels is $\mathcal{O}(n^2)$.

Finally, we turn to the problem of computing a graph $g$ that is the median of a set of graphs $G=\{g_1,..., g_N\}$ with unique node labels. In the remainder of this section, we assume, for the purpose of notational convenience, and without restricting generality, that all graphs under consideration are complete. That is, there is an edge $(x, y)\in E$ between any pair of nodes $x, y\in V$ for any considered graph $g$. "Real" edges can be easily distinguished from "virtual" edges by including a special *null* symbol in the edge label alphabet $L_E$ and defining $\beta(x, y)=null$ for any virtual edge. The benefit we get from considering complete graphs is that the only necessary edit operations on the edges are substitutions. In other words, any edge deletion or insertion now becomes a substitution that involves the *null* label. No conflicts will arise from this simplification because the cost of edge substitutions, deletions and insertions are the same.

Let $\rho(g_1),..., \rho(g_N)$ be the label representations of $g_1,..., g_N$. Define $L_U=\bigcup_{i=n}^{N} L_i$ and $C_U=\bigcup_{i=n}^{N} C_i$. Furthermore, let $\gamma(i)$ be the total number of occurrences of node label $i\in L_U$ in $L_1,..., L_N$. Note that $(1 \le \gamma(i) \le N)$. Formally, $\gamma(i)$ can be defined through the following procedure:

$\gamma(i)=0$;
for $k=1$ to $N$ do
  if $i \in L_k$ then $\gamma(i)=\gamma(i)+1$

Next, we define $\rho(g)=(L, C, \lambda)$ such that:

1. $L=\{i|i\in L_U$ and $\gamma(i)\ge N/2\}$;
2. $C=\{(i, j)|i, j\in L\}$; and
3. $\lambda(i, j)=max\_label(i, j)$,where the function *max_label*$(I,j)$ returns the label $\lambda_k(i, j)\in L_E$ that has the maximum number of occurrences on edge $(i, j)$ in $C_1,..., C_N$. In case of a tie, any of the competing labels $\lambda_k(i, j)$ may be returned.

*Lemma 5* Let $G$ and $\rho(g)$ be as above. Then, any graph $g$ with a label representation $\rho(g)$ is a median graph of $G$.

*Proof* The smallest potential median graph candidate is the graph with an empty set of nodes, while the largest potential candidate corresponds to the case $L=L_U$. The second observation is easy to verify because any graph $g^*$ that includes more node labels will have at least one label $k^*$ that does not occur in any of the $L_i$'s. Hence, the node with label $k^*$ will be deleted in all of the distance computations for $d(g^*, g)$, $i=1,..., N$. Therefore, dropping the node with label $k^*$ from $g^*$ will produce a graph with a smaller average edit distance to the members of $G$. It follows that, for any median graph $g$ with node label representation $\rho(g)$, the set $L$ must be necessarily a subset of $L_U$.

If we substitute the expression derived in Lemma 4 into the definition of a median graph given in Sect. 2, we recognise that any median graph $g$ with node label representation $\rho(g)$ must minimise the following expression:

$$\begin{aligned}\Delta = {} & |L| + |L_1| - 2|L \cap L_1| + |C| + |C_1| - 2|C_{01}| + \left|C'_{01}\right| \\ & + \ldots + |L| + |L_N| - 2|L \cap L_N| \\ & + |C| + |C_N| - 2|C_{0N}| + \left|C'_{0N}\right|\end{aligned}$$

where we use the following notation (see Lemma 4):

$$C_{0k}=\{(i,j)|(i,j)\in C\cap C_k\} \text{ and } \lambda(i,j)=\lambda_k(i,j)$$
$$C'_{0k}=\{(i,j)|(i,j)\in C\cap C_k\} \text{ and } \lambda(i,j)\neq\lambda_k(i,j)$$

Clearly, $\Delta$ can be rewritten as:

$$\begin{aligned}\Delta = {} & N|L| + \sum_{i=1}^{N}|L_i| - 2\sum_{i=1}^{N}|L\cap L_i| + N|C| + \sum_{i=1}^{N}|C_i| \\ & - 2\sum_{i=1}^{N}|C_{0i}| + \sum_{i=1}^{N}\left|C'_i\right|\end{aligned}$$

Note that all quantities are non-negative integers. As all $L_i$'s and $C_i$'s are given, the minimisation of $\Delta$ is equivalent to minimising:

$$N|L| - 2\sum_{i=1}^{N}|L\cap L_i| + N|C| - 2\sum_{i=1}^{N}|C_{0i}| + \sum_{i=1}^{N}\left|C'_{0i}\right|$$

First, we analyse the term $\Delta_1=N|L|-2\sum_{i=1}^{N}|L\cap L_i|$. It is obvious that $N|L|$ will become smaller if we include fewer nodes in the median graph. On the other hand, this will also make the term $2\sum_{i=1}^{N}|L\cap L_i|$ smaller, which leads to an increase of $\Delta_1$. To find the optimal number of nodes to be included in the median graph, we consider each element of $L$ individually and decide whether it must be included in the median graph or not. From the definition of $\Delta_1$, it follows that, if a node with label $i$ is included in the median graph, its contribution to $\Delta_1$ will be $N-2\gamma(i)$. Conversely, if that node is not included, its contribution will be zero. Hence, in order to minimise $\Delta_1$, we include a node with label $i$ in the median graph if $N-2\gamma(i)\le 0$, which is equivalent to $\gamma(i)\ge N/2$.

Now consider the term $\Delta_2=N|C| -2\sum_{i=1}^{N}|C_{0i}|+\sum_{i=1}^{N}\left|C'_{0i}\right|$. Assume for the moment that $|C|$ is a constant that is defined through the choice of $L$. Then, we have to minimise $-2\sum_{i=1}^{N}|C_{0i}|+\sum_{i=1}^{N}\left|C'_{0i}\right|$. As $|C_{0i}|+\left|C'_{0i}\right|=|C\cap C_0|$, this is equivalent to maximising $|C_{0i}|$. However, such a maximisation is exactly what is accomplished by the function *max_label*. This function chooses, for edge $(i, j)$, the label that most often occurs on edge $(i, j)$ in all the given graphs.

So far, we have treated the terms $\Delta_1$ and $\Delta_2$ independently of each other. In fact, they are not independent because the exclusion of a node with label $i$ from the median graph implies exclusion of any of its incident edges $(i, j)$ or $(j, i)$. Therefore, the question arises whether this dependency can lead to an inconsistency in the minimization of $\Delta=\Delta_1+\Delta_2$ in the sense that decreasing $\Delta_1$ leads to an increase of $\Delta_2$ by a larger amount, and vice versa. It is easy to see that such an inconsistency can never happen. First of all, exclusion of an edge $(i, j)$ for the sake of minimizing $\Delta_2$ does not imply any constraints

on inclusion or exclusion of any of the incident nodes $i$ and $j$. Second, if node $i$ is not included because $\gamma(i) < N/2$, the function *max_label* will surely return the *null* label for any edge $(i, j)$ or $(j, i)$. This is equivalent to not including $(i, j)$ or $(j, i)$ in the median graph. In other words, if a node $i$ is not included in the median graph because $\gamma(I) < N/2$, the dependency between $\Delta_1$ and $\Delta_2$ leads to also not including all incident edges, which is exactly what is required to minimize $\Delta_2$. This concludes the proof of Lemma 5.

In order to derive a practical computational procedure for the computation of a median of a set of graphs with unique node labels, we need to implement functions $\gamma(i)$ and *max_label*$(i, j)$. It is easy to verify that the complexity of these two functions is $\mathcal{O}(nN)$ and $\mathcal{O}(n^2N)$, respectively. It follows that the median graph computation problem can be solved in $\mathcal{O}(n^2N)$ time for graphs with unique node labels.

So far, we have assumed that there are $\mathcal{O}(n^2)$ edges in a graph with $n$ nodes. There are, however, applications where the graphs are of bounded degree, i.e. the maximum number of edges incident to a node is bounded by a constant $\kappa$. In this case, all of the expressions $\mathcal{O}(n^2)$ reduce to $\mathcal{O}(n)$.

The following theorem summarises all of the results derived in this section.

*Theorem 1* For the class of graphs with unique node labels, there exist computational procedures that solve the following problems in quadratic time with respect to the number of nodes in the underlying graph:

1. Graph isomorphism
2. Subgraph isomorphism
3. Maximum common subgraph
4. Graph edit distance under the cost function introduced earlier in this section

The median graph computation problem can be solved in $\mathcal{O}(n^2N)$ time where $n$ is the number of nodes in the largest graph and $N$ is the number of given graphs.

## 4 Application to computer network monitoring

The performance management of computer networks is becoming increasingly important as computer networks grow in size and complexity. Not surprisingly, this growth has resulted in an increase in frequency, type and severity of network problems [31, 32]. In addition, the dynamic behaviour exhibited by computer networks, in terms of their connectivity and traffic distributions, has significantly complicated the role of network performance management. Unusual activity patterns of network users is often a major contributor to the dynamic behaviour of these networks. Techniques are required to improve network monitoring and provide proactive detection of network anomalies so that problems can be corrected before they result in a disruption to services.

In [33, 34], graph similarity measures for network monitoring and abnormal change detection were proposed. The basic idea is to represent a computer network by a graph where the nodes represent clients or servers, and the edges represent physical connections between clients or servers. A time series of graphs $g_1$, $g_2$,...,$g_t$, $g_{t+1}$,..., $g_n$ is obtained by measuring the state of a network at regular time intervals and representing each state by a graph. Measures of network difference that use graph matching algorithms, such as MCS and GED, were introduced in [35]. These algorithms are applied to pairs of consecutive graphs $g_t$ and $g_{t+1}$ to measure network change. Change is classified as anomalous if the network change $d(g_t, g_{t+1})$ exceeds a given threshold $\alpha$. Another measure of network change can be achieved using the median graph. Similarly to classical time series analysis, it can be expected that using the median, computed over a time window of a certain length, rather than an individual graph, will lead to more robustness against outliers in the time series of graphs. In [33], four procedures, based on median graphs, were defined for abnormal change detection. Essentially, the median graph $\bar{g}_t$ is computed for a subsequence of graphs $(g_{t-L+1},..., g_t)$ of length $L$. Then, $d(\bar{g}_t, g_{t+1})$ can be used to measure the anomalous network change. We classify the change between $\bar{g}_t$ and $g_{t+1}$ as anomalous if $d(\bar{g}_t, g_{t+1})$ is larger than a given threshold. The average deviation $\phi$ occurring within the median window is used to assign a more robust threshold (i.e. $d(\bar{g}_t, g_{t+1}) \geqslant \alpha\varphi$).

The class of graphs considered in this paper have the requirement that each node label must be unique. This constraint does not pose a problem when dealing with graphs constructed from data collected from computer networks. It is common in these networks that each node, such as a client, server or router, be uniquely identified. For example, in an intranet employing ethernet technology on a local area network (LAN) segment, either the media access control (MAC) or the internet protocol (IP) address could be used to uniquely identify nodes on the local segment. As a consequence, the efficient graph matching algorithms described in Theorem 1 can be applied to computer networks to assist in network management functions.

Using Definition 1, the time series of graphs given above has label representation $\rho(g_1)$, $\rho(g_2)$,..., $\rho(g_t)$, $\rho(g_{t+1})$,..., $\rho(g_n)$. In order to gain a significant saving in the computation time of MCS and GED, we use the label representations $\rho(g_t)$ and $\rho(g_{t+1})$, and apply the matching algorithms defined in Lemmas 3 and 4. If the threshold is exceeded, it can be concluded that a significant network change has occurred at time $t+1$. A network administrator can then use other network management tools to determine whether the change represents an abnormal event. The implementation of the median graph detection strategies, using the label representation, is achieved using the algorithm defined in Lemma 4.

Given the diverse range of networks that must be managed by network administrators (e.g. LAN, wide area networks and the Internet) it is important that techniques

for measuring network change perform independently of the type of network topology. By inspection of the algorithms defined in Sect. 3, it can be deduced that, as long as the graphs have the same number of vertices and edges, the computation time for each method will be the same regardless of network topology. Thus, the techniques described are independent of network topology. This finding is verified in Sect. 5 and means that an accurate prediction of computation times can be determined for real network data based on measurements achieved for synthetic network data of equivalent size.

## 5 Experimental results

The aim of the experiments described in this section is to verify the low computational complexity for the theoretical results derived in Sect. 3. The time taken to compute isomorphism, subgraph isomorphism, MCS and GED are measured for graphs ranging in size from hundreds of nodes to tens of thousands of nodes, and with different edge densities. In addition, we validate the linear dependency of the time taken to compute a median graph to the number of graphs from which the median is derived. Computation times are measured for synthetic data sets and real network data. Real network data was acquired from a link in the core of a wide area computer network. A test for similarity of computation time measurements for real and synthetic data sets is made to verify that the results achieved for simulated networks can be repeated for real-world implementations. An experiment was conducted to verify that the time taken to compute algorithms in this paper are independent of network topology. Two graph generators were used to produce synthetic data sets having different network topologies. The real network data set was used as a third sample having different topology. The graphs in each data set had to be equivalent in the number of nodes and links for this test.

The hardware platform used to measure computation times was a SUN Fire V880 with 4×750 MHz UltraSparc3 processors and 8 GB of RAM. The specific hardware platform used to perform the experiments is not important and has been provided for completeness only. Only relative computation times with respect to graph dimensions are important.

### 5.1 Synthetic network data

Synthetic data sets are used to validate the computational complexity of the procedures defined in Sect. 3. These data sets are also used to verify that the procedures are independent of network topology.

Two data sets have been produced using normally distributed random edges with edge densities of 2.5% and 10%, respectively. An edge density of 2.5% was used so that graphs with 20,000 nodes could be synthesised without exceeding the computer memory of the computer platform used for the experiments. The data set with 10% edge density was chosen to mimic the characteristics of the real data network. The maximum number of nodes possible for graphs in this data set was 10,000.

An additional single synthetic data set having edge density of 2.5% was created using a Fan Chung algorithm [36]. This graph generator produced graphs having vertex degrees with a power-law distribution. The resultant topology of graphs produced using this method is quite different to those of graphs having normally distributed random edges. In fact, graphs having degree distribution that are power-laws are characteristic of large networks, such as the Internet [37].

For each synthetic data set, we first obtain a series $S$ of graphs, comprising 100, 1,000, 3,000, 5,000, 7,000 and 10,000 nodes. For data sets with an edge density of 2.5%, we obtain an additional graph in the series that has 20,000 nodes. The resulting graphs have directed edges with Poisson distributed edge weights. A second series $S'$ was produced as a counterpart, using the same procedure, for measurements of computation times for MCS and GED.

A further set of graphs was created to verify the linear increase in computation time with an increase in edge density, for a fixed number of nodes. The graph generator assigned edges using a normal distribution. For this data set, graphs had 5,000 vertices and edge densities ranging from 1% to 10% in steps of 1%. A counterpart was created for each graph to be used for MCS and GED computations.

To compare the computation times of algorithms measured for synthetic data against real data sets, we created two randomly distributed graphs having the same number of vertices and edges as each of the real data sets in Sect. 5.2.

Finally, for the validation of computation times for median graphs, we created a series of 100 graphs using randomly distributed edges. In this series, the average number of vertices and the edge density is matched to our business domain network data set (i.e. comprises graphs having on average 70 vertices with edge density of 10%) as described in Sect. 5.2.

### 5.2 Real network data

Real network data was acquired from a core link in a large enterprise data network using network performance monitoring tools. The data network employs static IP addresses, hence, its suitability for representation by the class of graphs defined in this paper. Graphs were produced from traffic traversing the link at intervals of one day. This resulted in a time series of 100 graphs representing 100 days of network traffic.

Two levels of abstraction have been used to produce the time series of real network data. Both have quite different characteristics. The first data set has graph vertices that represent IP addresses, whilst the second

has vertices that represent business domains. In both data sets, edges represent logical links, and edge weights represent the total number of bytes communicated between vertices in one day. The business domain abstraction is created by coalescing IP addresses belonging to a predefined business entity into a single node. This resulted in graphs that comprises on average 70 nodes with edge densities of 10%. The IP network abstraction has graphs that have on average 9,000 nodes with an edge density of 0.04%. The low edge density is a result of the near bipartite nature of the graphs arising from data collected at a single point in the core of the enterprise data network. The business domain and IP network abstractions are of interest to network administrators as they provide both coarse and fine network performance data, respectively.

Two consecutive graphs were chosen from each of the real network data set abstractions to be used in comparisons of computation times of algorithms with times measured for synthetic data. The two graphs chosen for the business domain abstraction comprised approximately 90 vertices with an edge density of 10%, whilst the graphs chosen from the IP abstraction comprised 9,000 vertices with an edge density of 0.04%.

To verify median graph computation times, the whole 100-day time series of graphs of business domain data was used.

### 5.3 Verification of $\mathcal{O}(n^2)$ theoretical computational complexity for isomorphism, subgraph isomorphism, MCS and GED

To measure the time taken to compute a test for graph isomorphism, we select the first graph $g_1$ from $S$, comprising 100 unique nodes, and make an exact copy $g_2$. The fact that $g_2 = g_1$ guarantees that the graphs tested are in fact isomorphic to each other. The computation time measurement does not include the time taken to derive the label representations $\rho(g_1)$ and $\rho(g_2)$ for graphs $g_1$ and $g_2$. This is true for all computation times measured for each algorithm. For the measurement of computation time for the subgraph isomorphism test, we use the same graph $g_1$ together with graph $g_3$, obtained by removing 20% of the edges from $g_1$. The graph $g_3$ is obviously a subgraph of $g_1$. The measurements of time to compute both MCS and GED required both graph series $S$ and $S'$. To measure the time taken to execute these algorithms, we again use $g_1$ from $S$ and select the equivalent size graph from $S'$. The procedures outlined above were repeated for all three synthetic data sets for graph sizes 1,000, 3,000, 5,000, 7,000, 10,000 and 20,000 (where present).

The results of all computation time measurements are shown in Tables 1, 2, 3 and 4. As expected, the measured computational complexity of all matching algorithms is $\mathcal{O}(n^2)$. Figure 2 graphically illustrates the quadratic dependence of computation time on the number of nodes for GED; the $x$-axis corresponds to the number of nodes in a graph and the $y$-axis represents the time, in seconds, to compute the GED algorithm. Greater computation times can be observed for larger edge densities. This result was anticipated due to the dependency on graph elements. Computation times to test for graph isomorphism were the longest. Testing for subgraph isomorphism required the least time to compute. This was a consequence of removing 20% of the edges from $g_1$ to produce a subgraph $g_2$. The smaller the size of $g_2$ with respect to $g_1$, the shorter the time taken to compute the subgraph isomorphism. The computation times for both MCS and GED, as observed in Figs. 3 and 4, are almost indistinguishable. This is not surprising since the computational steps, proposed in Lemmas 3 and 4, are nearly identical. In all cases, the computation times measured for both randomly distributed edges and those with power-law degree distributions, for an edge density of 2.5%, are nearly identical. The results would be identical if the number of edges in the graphs from both data sets were equal. Since the graph generator used to produce graphs with randomly distributed edges create, on average, graphs with a specified edge density, the actual number of edges can vary. The closeness of the

**Table 1** Computation times for isomorphism

| | Computation times(s) for graphs with $N$ vertices | | | | | | |
|---|---|---|---|---|---|---|---|
| | 100 | 1,000 | 3,000 | 5,000 | 7,000 | 10,000 | 20,000 |
| Fan Chung 2.5% | 0.01 | 0.55 | 5.48 | 17.41 | 36.77 | 78.97 | 343.77 |
| Random 2.5% | 0.02 | 0.5 | 5.33 | 17.74 | 37.94 | 81.63 | 383.80 |
| Random 10% | 0.02 | 2.00 | 25.16 | 77.75 | 163.22 | 357.72 | |

**Table 2** Computation times for subgraph isomorphism

| | Computation times(s) for graphs with $N$ vertices | | | | | | |
|---|---|---|---|---|---|---|---|
| | 100 | 1,000 | 3,000 | 5,000 | 7,000 | 10,000 | 20,000 |
| Fan Chung 2.5% | 0.01 | 0.38 | 3.04 | 10.04 | 21.33 | 45.60 | 197.32 |
| Random 2.5% | 0.01 | 0.33 | 2.82 | 9.51 | 20.71 | 45.03 | 206.90 |
| Random 10% | 0.01 | 1.17 | 13.92 | 43.23 | 90.23 | 195.68 | |

**Table 3** Computation times for MCS

| | Computation times(s) for graphs with $N$ vertices | | | | | | |
|---|---|---|---|---|---|---|---|
| | 100 | 1,000 | 3,000 | 5,000 | 7,000 | 10,000 | 20,000 |
| Fan Chung 2.5% | 0.01 | 0.38 | 3.44 | 11.21 | 24.28 | 52.36 | 230.63 |
| Random 2.5% | 0.01 | 0.40 | 3.18 | 10.74 | 23.63 | 51.69 | 237.16 |
| Random 10% | 0.01 | 1.28 | 16.21 | 49.40 | 102.27 | 221.55 | |

**Table 4** Computation times for GED

| | Computation times(s) for graphs with $N$ vertices | | | | | | |
|---|---|---|---|---|---|---|---|
| | 100 | 1,000 | 3,000 | 5,000 | 7,000 | 10,000 | 20,000 |
| Fan Chung 2.5% | 0.01 | 0.40 | 3.49 | 11.34 | 24.46 | 52.09 | 230.51 |
| Random 2.5% | 0.01 | 0.33 | 3.38 | 10.90 | 23.63 | 51.45 | 232.09 |
| Random 10% | 0.01 | 1.35 | 16.30 | 49.00 | 101.88 | 220.26 | |

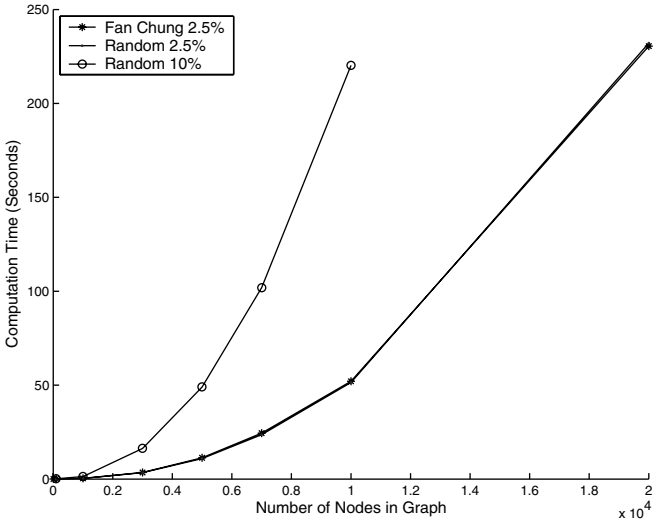


**Fig. 2** Computation times for GED

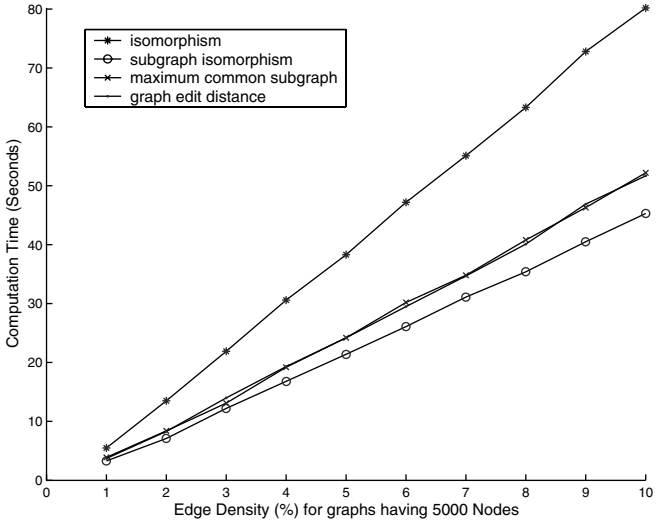


**Fig. 3** Plot of linear dependency of computation times with respect to edge density

results verifies the independence of the algorithms to network topology.

Further experimentation was performed to show the linear dependency of computational complexity in the number of edges in a graph with a fixed number of vertices. Figure 3 shows results for the four graph algorithms. Observation of these results reveals the linear relationship.

### 5.4 Comparison of computation times for real and synthetic data sets

In this section, measurements of computation times of isomorphism, subgraph isomorphism, MCS and GED on the two real network data sets (i.e. business domain and IP-level abstractions) and their synthetic counterparts were performed. The aim was to confirm that synthetic data measurements are consistent with those measured for real network data.

The label representation is first derived for the two graphs in each data set. Isomorphism and subgraph isomorphism computation requires only one of the graphs from each set. Both graphs are required for the MCS and GED computations. The results are given in Table 5. It can be seen that all measurements between the real and equivalent synthetic data sets agree. This infers that results obtained for synthetic data are consistent to those for real data.

### 5.5 Verification of theoretical computation times for the median graph

The computation time of the median graph algorithm described in Sect. 3 is measured for both real and synthetic data sets. Both comprise a time series of 100 graphs. The sizes of the graphs within each time series and between time series are similar. We wish to verify that the time taken to compute a median graph increases linearly as the number of graphs in the median computation increases.

Measurements commenced by taking the first 10 graphs in the time series (i.e. $\{g_1,..., g_{10}\}$) of each data set and computing the median graph. The procedure was repeated using the first 20, 30, 40,..., 90 and 100 graphs. The results are given in Table 6 and Fig. 4. Both real and synthetic data sets show a linear dependency of computation time with respect to the number of graphs. The plot of computation times for the real data set deviates from a straight line because the edge counts in this data set had a greater standard deviation than those of the synthetic data set. The number of vertices and edges in graphs belonging to real and synthetic data sets can been seen in Fig. 5.

## 6 Summary and conclusions

Graph matching is finding many applications in the fields of science and engineering. In this paper, we considered a special class of graphs, characterised by unique node labels. A label representation is given for graphs in this class. For a given graph, it comprises a set of unique

**Table 5** Comparison of computation times for real and synthetic network data

| | Real network data | | | |
|---|---|---|---|---|
| | Business domain | | IP domain | |
| | Real | Synthetic | Real | Synthetic |
| Isomorphism | 0.02 | 0.02 | 0.60 | 0.58 |
| Subgraph isomorphism | 0.01 | 0.01 | 0.35 | 0.32 |
| MCS | 0.01 | 0.01 | 0.38 | 0.36 |
| GED | 0.01 | 0.01 | 0.38 | 0.36 |

**Table 6** Computation times for median graph

| Number of graphs in median | Computation time(s) | |
|---|---|---|
| | Real data | Synthetic data |
| 10 | 2.21 | 3.36 |
| 20 | 2.75 | 4.72 |
| 30 | 3.17 | 5.37 |
| 40 | 3.53 | 6.00 |
| 50 | 3.83 | 6.39 |
| 60 | 4.08 | 6.76 |
| 70 | 4.27 | 7.16 |
| 80 | 4.16 | 7.47 |
| 90 | 4.48 | 7.79 |
| 100 | 4.93 | 8.15 |

vertex labels of the graph, an edge set based on vertex labels and a set of edge weights. A number of computationally efficient matching algorithms were derived for this class of graphs. The suitability of applying these matching algorithms to computer network monitoring was addressed.

The matching algorithms that have been derived for graphs having a label representation are detection of graph isomorphism and subgraph isomorphism, computation of MCS, GED and the median graph. The theoretical computational complexity of these algorithms, for graphs having $n$ nodes, is $\mathcal{O}(n^2)$. It was also shown that the time taken to compute a median graph increases linearly with the number of graphs in the set from which the median is computed. Theoretical results were verified using real and synthetic data sets.

It is possible to apply the derived matching algorithms to computer network monitoring as the constraint for unique node labels can be satisfied. In computer networks, nodes can be uniquely identified by means of the MAS or IP addresses. The matching algorithms proposed can be used to measure a change that occurs in a computer network over time. Measures of network change provide good indicators of when

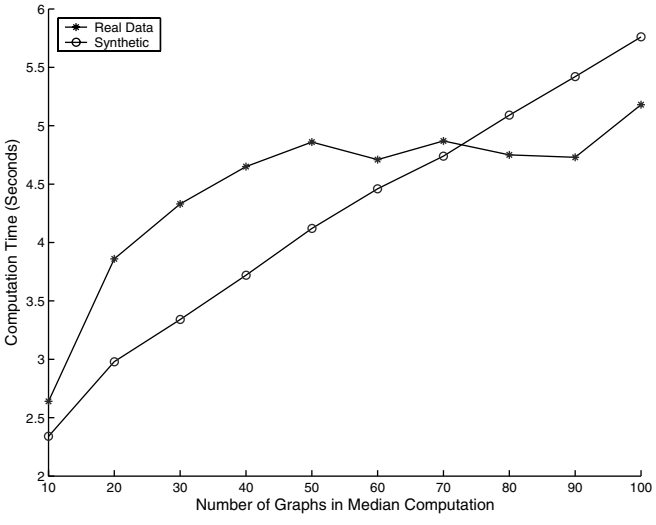


**Fig. 4** Computation times for median graph algorithm

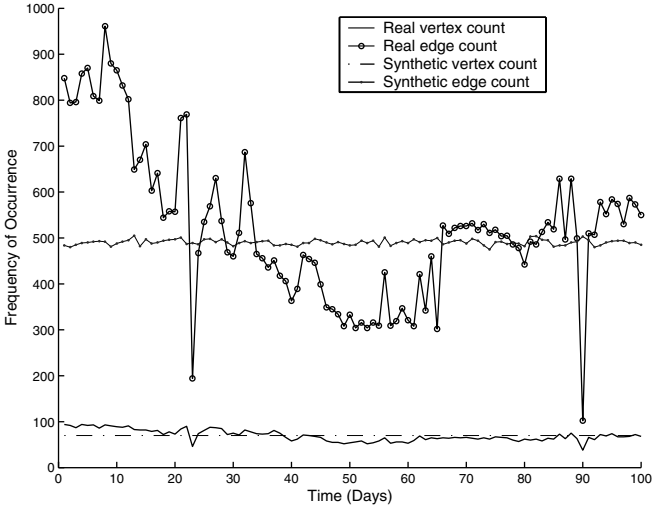


**Fig. 5** Vertex and edge counts for real and synthetic data sets

abnormal network events have occurred. Such techniques greatly enhance computer network management, especially in the field of performance monitoring.

The theoretical computational complexity of the matching algorithms were verified through experimentation using synthetic and real network data. Synthetic data sets of graphs with a specified number of nodes and edge densities were used for this purpose. In addition, synthetic data sets having different network topologies were used to show that the computation times for derived algorithms are independent of network topology. A comparison of results achieved for synthetic data sets with those obtained using data acquired from a large wide area computer network of equal dimension were shown to agree. This outcome, along with the knowledge that the algorithms are independent of network topology, means that the simulation of performance of the algorithms on synthetic data can be used to accurately predict the performance that will be achieved for real networks.

In conclusion, graph matching algorithms for uniquely labelled graphs having a label representation provide a significant computational saving compared to the generalised class of graphs where such matching algorithms have an exponential computational complexity. In this paper, we have shown that, for this class of graphs, we have been able to apply matching algorithms to graphs having many thousands of nodes. This is a significant improvement in the pattern recognition community, where graphs with a few hundred nodes is considered very large.

The application considered in this paper, i.e. computer network analysis, is not a traditional application of pattern recognition, such as detection or classification of objects in an image. Nevertheless, it surely belongs to the discipline of pattern recognition because the considered task, i.e. abnormal event detection, is cast as a classification problem, where the change occurring in the

network between time $t$ and time $t+1$ is to be categorised as normal or abnormal. It remains an open problem to identify more problems in pattern recognition and artificial intelligence, in addition to the ones described in [25–27], where graphs with unique node labels are suitable object representatives.

## 7 Originality and contribution

The problem of graph matching, such as the computation of GED edit distance, is NP complete. In this paper, we consider a class of graphs with an inherently lower computational cost for graph matching tasks. These graphs are characterised by the existence of unique node labels. The definition of a label representation is given for graphs in this class, and $\mathcal{O}(n^2)$ matching algorithms are devised for isomorphism, subgraph-isomorphism, MCS, GED and the median graph. Whilst graph matching is not in itself novel, the algorithmic framework developed in this paper for the special class of graphs is new and offers significant computational improvements, especially when dealing with very large graphs. The paper introduces some interesting applications for this class of graphs and verifies the theoretical low computational complexity with practical results.

We explore the application of this class of graphs to computer network monitoring. Nodes in the computer network under investigation have static address allocations and, hence, can be considered to have unique identities. The networks are very large in size and are dynamic in behaviour. This has lead to increasing difficulties in network performance management. The proactive detection of network anomalies is of specific importance. In this application, we use GED and median graph computation to detect abnormal changes in network behaviour. This is very much a useful contribution to the discipline of pattern recognition as computer network monitoring is a classification problem where we want to categorise network changes as either normal or abnormal.

## 8 Biographies of authors

**Arek Dadej** is an Associate Professor and leader of the Telecommunication Networks and Services research group at the Institute for Telecommunications Research (ITR), University of South Australia. His current research interests comprise protocols for mobility support in IP networks, QoS control in wireless/mobile network environment, routing and network configuration, and ad-hoc network service discovery and user access control. His teaching areas include many courses in telecommunication networks and computer systems engineering. Arek has led major industry-sponsored research projects in telecommunication networks and protocols, like high-capacity 802.11 WLAN design with guaranteed QoS, a study of multicasting and scheduling techniques in satellite broadcast systems and a study of ad-hoc networking technologies. He also led two projects within the Cooperative Research Centre for Satellite Systems, focussing on the network architectures, protocols and on-board processing for ATM and IP-based service delivery via networks of small satellites.

**Horst Bunke** received his MSc and PhD degrees in Computer Science from the University of Erlangen, Germany. In 1984, he joined the University of Bern, Switzerland, where he is a professor in the Computer Science Department. He was Department Chairman from 1992 to 1996 and Dean of the Faculty of Science from 1997 to 1998. His research interests comprise computer vision, image analysis, pattern recognition and artificial intelligence. From 1998 to 2000, Horst Bunke was the first Vice-President of the International Association for Pattern Recognition (IAPR). He is a fellow of the IAPR, former editor-in-charge of the International Journal of Pattern Recognition and Artificial Intelligence, editor-in-chief of Electronic Letters of Computer Vision and Image Analysis journal, editor-in-chief of the book series on Machine Perception and Artificial Intelligence by World Scientific Publ. Co., associate editor of Acta Cybernetica, the International Journal of Document Analysis and Recognition and Pattern Analysis and Applications. Horst Bunke was on the program and organisation committee of many conferences and served as a referee for numerous journals and scientific organisations. He has more than 450 publications, including 28 books and special editions of journals.

**Dr. Miro Kraetzl** is a Principal Research Scientist in the Communications Analysis Group of the Defence Science and Technology Organisation (DSTO). He received BSc and MSc degrees in pure mathematics from the University of Zagreb (Croatia) and a PhD in graph theory from Curtin University of Technology in Perth (Western Australia). His current research interests comprise analysis of telecommunication, information and social network dynamics, parallel computer interconnection architectures and information geometry. Miro holds adjunct positions at the School of Applied Mathematics at the University of Adelaide and at the Institute for Telecommunications Research at the University of South Australia. He is a member of AFCEA.

**Peter Dickinson** is a Senior Engineer in the Communications Analysis Group of the Defence Science and Technology Organisation (DSTO). He received BE in Electronic Engineering from the South Australian Institute of Technology in Adelaide. His research interests comprise telecommunication and computer network analysis. Peter is currently working towards a PhD in performance monitoring of large dynamic intranets with the Telecommunication Networks and Services research group at the Institute for Telecommunications Research (ITR), University of South Australia.

## References


1. (2001) Special section on graph algorithms and computer vision. IEEE Trans PAMI 23(10)
2. (2003) Special issue on graph-based representations in pattern recognition. Pattern Recognit Lett 24(8)
3. (2004) Special issue on graph matching in pattern recognition and machine vision. Int J Pattern Recognit Artif Intell 18(3)
4. McKay B (1981) Practical graph isomorphism. Congressus Numerantium 30:45–87
5. Ullman JR (1976) An algorithm for subgraph isomorphism. J ACM 23(1):31–42
6. Levi G (1972) A note on the derivation of maximal common subgraphs of two directed or undirected graphs. Calcolo 9:341–354
7. McGregor J (1982) Backtrack search algorithms and the maximal common subgraph problem. Software Pract Experience 12(1):23–34
8. Messmer BT, Bunke H (1998) A new algorithm for error-tolerant subgraph isomorphism detection. IEEE Trans Pattern Anal Machine Intell 20:493–504
9. Sanfeliu A, Fu KS (1983) A distance measure between attributed relational graphs for pattern recognition. IEEE Trans Syst Man Cybern 13(3):353–362
10. Cordella LP, Foggia P, Sansone C, Vento M (2001) An improved algorithm for matching large graphs. In: Proceedings of the 3rd IAPR-TC15 workshop on graph based representations in pattern recognition, Naples, Italy, May 2001, pp 149–159
11. Larrosa J, Valiente G (2002) Constraint satisfaction algorithms for graph pattern matching. Math Struct Comput Sci 12:403–422
12. Christmas WJ, Kittler J, Petrou M (1995) Structural matching in computer vision using probabilistic relaxation. IEEE Trans PAMI 8:749–764
13. Wilson RC, Hancock E (1997) Structural matching by discrete relaxation. IEEE Trans PAMI 19:634–648
14. Cross A, Wilson R, Hancock E (1997) Inexact graph matching with genetic search. Pattern Recognit 30:953–970
15. Wang I, Fan K-C, Horng J-T (1997) Genetic-based search for error-correcting graph isomorphism. IEEE Trans SMC 27:588–597
16. Luo B, Hancock E (2001) Structural graph matching using the EM algorithm and singular value decomposition. IEEE Trans PAMI 23:1120–1136
17. Kosinov S, Caelli T (2002) Inexact multisubgraph matching using graph eigenspace and clustering models. In: Caelli T, Amin A, Duin R, Kamel M, de Ridder D (eds) Structural, syntactic, and statistical pattern recognition, LNCS 2396. Springer, Berlin Heidelberg New York, pp 133–142
18. Luo B, Wilson R, Hancock E (2002) Spectral feature vectors for graph clustering. In: Caelli T, Amin A, Duin R, Kamel M, de Ridder D (eds) Structural, syntactic, and statistical pattern recognition, LNCS 2396. Springer, Berlin Heidelberg New York, pp 83–93
19. Pelillo M, Jagota A (1995) Feasible and infeasible maxima in a quadratic program for maximum clique. J Art Neural Netw 2(4):411–420
20. Hopcroft JE, Wong JK (1974) Linear time algorithm for isomorphism of planar graphs. In: Proceedings of the 6th annual ACM symposium on theory of computing, Seattle, Washington, April/May 1974, pp 172–184
21. Jiang X, Bunke H (1996) Including geometry in graph representations: a quadratic-time graph isomorphism algorithm and its application. In: Perner P, Wang P, Rosenfeld A (eds) Advances in structural and syntactic pattern recognition, LNCS 1121. Springer, Berlin Heidelberg New York, pp 110–119
22. Luks EM (1982) Isomorphism of graphs of bounded valence can be tested in polynomial time. J Comput Syst Sci 25:42–65
23. Pelillo M (2002) Matching free trees, maximal cliques, and monotone game dynamics. IEEE Trans PAMI 24(11):1535–1541
24. Shokonfandeh A, Dickinson S (2001) A unified framework for indexing and matching hierarchical shape structures. In: Arcelli C, Cordella L, Sanniti di Baja G (eds) Visual form 2001, LNCS 2059. Springer, Berlin Heidelberg New York, pp 67–84
25. Schenker A, Last M, Bunke H, Kandel A (2003) Clustering of web documents using a graph model. In: A Antonacopoulos, H Jianying (eds) Web document analysis: challenges and opportunities. World Scientific, River Edge, New Jersey
26. Schenker A, Last M, Bunke H, Kandel A (2003) Classification of web documents using a graph model. In: Proceedings of the 7th international conference on document analysis and recognition, Edinburgh, Scotland, August 2003, pp 472–476
27. Schenker A, Last M, Bunke H, Kandel A (2004) Classification of web documents using graph matching. Pattern Recognit Artif Intell 18(3):475–496
28. Dickinson P, Bunke H, Dadej A, Kraetzl M (2003) On graphs with unique node labels. In: Hancock E, Vento M (eds) Proceedings of the 4th IAPR international workshop on graph based representations in pattern recognition (GbRPR 2003), York, UK, June/July 2003. Springer, Berlin Heidelberg New York, pp 13–23
29. Jiang X, Munger A, Bunke H (2001) On median graphs: properties, algorithms, and applications. PAMI 23(10):1144–1151
30. Bunke H (1999) Error correcting graph matching: on the influence of the underlying cost function. IEEE Trans PAMI 21:917–922
31. Huberman BA, Lukose RM (1997) Social dilemmas and internet congestion. Science 277(5325):535–537
32. Snow AP, Weiss MBH (1997) Empirical evidence of reliability growth in large-scale networks. Netw Syst Manag 5(2):197–213
33. Dickinson P, Bunke H, Dadej A, Kraetzl M (2001) Application of median graphs in detection of anomalous change in communication networks. In: Proceedings of the 5th world multiconference on systemics, cybernetics and informatics (SCI 2001) vol 5, Orlando, Florida, July 2001
34. Shoubridge PJ, Kraetzl M, Wallis WD, Bunke H (2002) Detection of abnormal change in a time series of graphs. J Interconnection Netw 3:85–101
35. Bunke H, Kraetzl M, Shoubridge PJ, Wallis WD (2002) Measuring change in large enterprise data networks. In: Proceedings of the conference on information, decision and control (IDC 2002), Adelaide, South Australia, February 2002, pp 53–58
36. Chung FRK, Lu L (2002) Connected components in random graphs with given expected degree sequences. Ann Comb (6):125–145
37. Tangmunarunkit H, Govindan R, Jamin S, Shenker S, Willinger W (2002) Network topology generators: degree-based vs structural. In: Proceedings of the ACM SIGCOMM 2002 conference on applications, technologies, architectures, and protocols for computer communication, Pittsburgh, Pennsylvania, August 2002