Design Pattern Mining for GIS Application Using Graph Matching Techniques

Akshara Pande  
DST- Centre for Interdisciplinary Mathematical Sciences, Banaras Hindu University, Varanasi-221005, India  
pandeakshara@gmail.com  

Manjari Gupta  
DST- Centre for Interdisciplinary Mathematical Sciences, Department of Computer Science, Banaras Hindu University, Varanasi-221005, India  
manjari@bhu.ac.in  

A.K.Tripathi  
Department of Computer Engineering, Institute of Technology, Banaras Hindu University, Varanasi-221005, India  
aktripathi.cse@itbhu.ac.in

Abstract—Design Pattern Detection is a part of many solutions to Software Engineering difficulties. It is a part of reengineering process and thus gives important information to the designer. Design Pattern existence improve the program understanding and software maintenance. With the help of these patterns specific design problem can be solved and object oriented design become more flexible and reusable. Hence a reliable design pattern mining is required. A GIS is an information system designed to work with data referenced by spatial / geographical coordinates. Here we are detecting design patterns so that it can be used as a conceptual tool to cope with recurrent problems appearing in the GIS domain. In this way, GIS applications can evolve smoothly, because maintenance is achieved by focusing on different concerns at different times.

Keywords-design pattern, graph distance, UML, matrix, template matching, normalized crossed correlation, subgraph isomorphism

I. INTRODUCTION

Geographic Information Systems deal with many characteristics such as data acquisition, accuracy, representation of spatial relationships, topological features and interface design. The designers must consider them when developing geographic applications. In simple words, GIS is used for database applications that store and analyze georeferenced data both. The relational model upon which most Current GIS software systems are built has been acknowledged as an insufficient model for applications that deal with spatial data [14], [15]. The use of object oriented technology was proposed for design of Geographical Information System and now it is becoming a growing trend in the GIS applications [12], [13]. In GIS domain there are many recurrent problems involving the use of spatial information such as object locations, coordinate manipulation, computation of geographic functions, and so on.

In many object oriented software, there are recurring patterns of classes. Design Patterns are defined as explanation of corresponding classes that forms a common solution to frequent design problem. To reuse expert design experiences, the design patterns [1] have been extensively used by software industry. A software design pattern, filter and refine, which is widely used in spatial database design to improve the efficiency of spatial retrieval. When patterns are implemented in a system, the pattern-related information is generally no longer available. It is tough to trace out such design information. To understand the systems and to modifications in them it is necessary to recover pattern instances. There are number of pattern detection techniques [2], [3], [4]. In this paper we firstly draw the equivalent UML diagram for GIS application and then try to find out whether a particular design pattern exists in that application or not by applying graph matching techniques [5], [6], [8], and [17]. In this paper we propose methods (i.e. Graph Distance Approach, Normalized Crossed Correlation and Subgraph Isomorphism Detection) for design pattern detection. The detailed methods are given in below sections.

II. UML DIAGRAM IN GIS DOMAIN

Consider a problem in which the area is selected by GIS system which consists of buildings (like schools, hospitals, residents, offices etc), empty lots and blocks.

III. GRAPH MATCHING TECHNIQUES

Graph Matching techniques are important and very general form of pattern matching that finds realistic use in areas such as image processing, pattern recognition and
computer vision, graph grammars, graph transformation, bio
computing, search operation in chemical structural formulae
database, etc.

A. Graph Distance

Let \( g_1 = (V_1, E_1, \alpha_1, \beta_1) \) and \( g_2 = (V_2, E_2, \alpha_2, \beta_2) \) be
graphs. A common subgraph of \( g_1 \) and \( g_2 \), \( CS(g_1, g_2) \), is a
graph \( g \) \((V, E, \alpha, \beta)\). We call \( g \) a minimum common
subgraph of \( g_1 \) and \( g_2 \), \( MCS(g_1, g_2) \) [16]. The graph
distance is given as [17]:

\[
\delta(g_1, g_2) = 1 - \frac{\text{MCS}(g_1, g_2)}{\max(|g_1|, |g_2|)} \tag{1}
\]

Equation (1) measures the graph distance. For any three
graphs the following relations hold [17]:

- \( 0 \leq \delta(g_1, g_2) \leq 1 \)
- \( \delta(g_1, g_2) = 0 \iff g_1 = g_2 \)
- \( \delta(g_1, g_2) = \delta(g_2, g_1) \)
- \( \delta(g_1, g_3) \leq \delta(g_1, g_2) + \delta(g_2, g_3) \)

B. Template Matching by Normalized Crossed Correlation

Normalized cross correlation (NCC) has been used
extensively for many machine vision applications. Normalized
cross correlation (NCC) has been commonly
used as a metric to evaluate the degree of similarity
(or dissimilarity) between two compared images [6]. Here we
are taking two graphs one corresponding to system and
another for design pattern (template graph), then we are
applying NCC to find match between template \((i, j)\) of size
\( m \times n \) and system matrix \( f(x, y) \) of size \( M \times N \). NCC is
defined as

\[
\tilde{d}(x, y) = \frac{\sum_{i,j=0}^{m-1} \sum_{j=0}^{n-1} f(x+i, y+j) r(i, j) - mn \mu_f \mu_r}{\sqrt{\left(\sum_{i,j=0}^{m-1} \sum_{j=0}^{n-1} f^2(x+i, y+j) - mn \mu_f^2\right) \left(\sum_{i,j=0}^{m-1} \sum_{j=0}^{n-1} r^2(i, j) - mn \mu_r^2\right)}} \tag{2}
\]

For all \((x, y) \in M \times N\). Where,

- \( \mu_f(x, y) = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} f(x+i, y+j) \)
- \( \mu_r(x, y) = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} r(i, j) \)

C. Subgraph Isomorphism Detection

The subgraph isomorphism is an important generalization
of graph isomorphism. The subgraph isomorphism problem
[7] is to determine whether a graph is isomorphic to a
subgraph of another graph. Subgraph isomorphism is NP-
complete.

Let [5] \( G_1 \) \((V_1, E_1)\) and \( G_2 \) \((V_2, E_2)\) be two graphs,
where \( V_1, V_2 \) are the set of vertices and \( E_1, E_2 \) are the set of
edges. Let \( M_1 \) and \( M_2 \) be the adjacency matrices
corresponding to \( G_1 \) and \( G_2 \) respectively. A permutation
matrix is a square \((0,1)\)-matrix that has exactly one entry 1
in each row and each column and 0's elsewhere. Two graphs
\( G_1 \) \((M_1, L_v, L_e)\) and \( G_2 \) \((M_2, L_v, L_e)\) are said to be
isomorphic [5] if there exist a permutation matrix \( P \) such that

\[
M_2 = P M_1 P^T \tag{3}
\]

Given an \( n \times n \) matrix \( M = (m_{ij}) \), let \( S_{k,m}(M) \) denote the \( k \times m \)
matrix that is obtained from \( M \) by deleting rows \( k+1, \ldots, n \) and columns \( m+1, \ldots, n \), where \( k, m < n \). A
subgraph \( S \) of a graph \( G, S \subseteq G \), is a graph \( S = (M', L_v, L_e) \)
where \( M' = S_{k,m}(P M P^T) \) is an \( m \times m \) adjacency matrix for
some permutation matrix \( P \). The concept of subgraph
isomorphism can now be described as follows.

Let \( G_1 \) and \( G_2 \) be graphs with adjacency matrices
\( M_1 \) and \( M_2 \) of dimensions \( m \times m \) and \( n \times n \) respectively,
where \( m < n \). There is a subgraph isomorphism [5] from \( G_1 \)
to \( G_2 \) if there exists an \( n \times n \) permutation matrix \( P \) such that

\[
M_1 = S_{k,m}(P M_2 P^T) \tag{4}
\]

We take \( M_2 \) matrix as a system design matrix and we
guess nondeterministically \( M_1 \) as a design pattern matrix.
And then try to find out whether \( M_1 \) is subisomorphic to \( M_2 \)
or not or it can be easily said whether there exist design
pattern in the system graph or not.

Hence the problem of finding a subgraph isomorphism
from graph \( G_1 \) to \( G_2 \) is equivalent to finding a permutation
matrix for which equation (4) holds. Thus, we generate
permutation adjacency matrix of a model graph (system
under study) one by one and check whether equation (4)
holds or not, when it holds we stop and declare that that
particular design pattern has been detected. It can be also
possible that there is no design pattern exists in system graph.
In this case we find no permutation matrix for which
equation (4) holds.

D. Examples

We have taken the system under consideration (i.e. the
UML diagram which is developed under GIS) as well as
design pattern. The generalization and aggregation graph of
system under study (i.e. corresponding to UML diagram
shown Fig. 1) is shown in Fig. 2.
Equation (1) can be applied for calculating the graph distance between two nodes of the graph. Firstly take the generalization graph from Fig. 2 (corresponding to system design) and Fig. 4 (corresponding to Prototype design pattern). The maximum common nodes between them are 2 and maximum number of nodes is 9 (for system graph). Here, $|\text{MCS}(g_1, g_2)| = 2$, max ($|g_1|$, $|g_2|$) = 9, so, on applying equation (1) we have $\delta(g_1, g_2)$ as $7/9$ i.e. $0.77$.

The limitation of above method is that it only calculates the distance between two vertices rather than whole graph. To remove this drawback another approaches are present there, called normalized crossed correlation and subgraph isomorphism detection. For applying these approaches we have to consider adjacency matrices corresponding to each relationship graphs of UML diagrams.

IV. MATRIX REPRESENTATION

Firstly we write the matrix corresponding to system design (i.e. the area which is selected by GIS system), here are two relationships for system graph, generalization and aggregation, so corresponding to them generalization matrix and aggregation matrix are shown in fig 5 and fig 6 respectively.

In the similar way, generalization and association matrices for prototype design pattern are shown in figure 7 and figure 8.

Now from fig 5 and fig 7 (i.e. matrices corresponding to generalization relationships of system design and prototype design pattern), if normalized cross correlation formula (i.e. equation 2) is applied, then the value of $\delta(0, 0)$ is approximately 0.65 (Here we are starting row from 0th row instead of saying 1st row). For $\delta(2, 2)$ the value is 1, which indicates the proper match means for generalization relationship the prototype design pattern exists in system design.
This is very time consuming to calculate each feature matrix (i.e. relationships present example generalization, direct association etc) separately. So we are trying to write it down in the form of overall matrix [3].

There is one or more design features are present in system and patterns, they can be represented in the form of matrices. For example there is generalization relationship, we have to create \(n \times n\) matrix for \(n\) number of classes. If \([3]\) the \(i^{th}\) and \(j^{th}\) classes have generalization relationship, the corresponding cell of the matrix should be 1, otherwise 0.

To reduce the number of manipulations, we try to combine different matrices (like generalization matrix, association matrix, dependency matrix, aggregation matrix etc) into a single matrix. So there will be only two overall matrices, one corresponding to system and one for design pattern. To combine different matrices into an overall matrix, certain root value of a prime number is given to a matrix (different feature matrices) and then combine them. The cell value [3] of each matrix is then changed to the value of its root to the power of the old cell value. Let us consider the UML diagram corresponding to system design (Fig. 1) and other UML diagrams corresponding to design pattern (i.e. Composite Design Pattern).

### A. Calculation of matrices for system design

Now firstly we calculate the overall matrix corresponding to model graph (i.e. system design Fig. 1), there are 2 corresponding matrices aggregation matrix and generalization matrix.

#### Generalization matrix
\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\]

#### Aggregation matrix
\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\]

### B. Calculation of matrices for design patterns

Let us consider Composite Design Pattern (Fig. 9). There are two matrices one matrix for relationship generalization and other for aggregation relationship.

#### Generalization matrix (root = 2)
\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
6 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

#### Aggregation matrix (root = 3)
\[
\begin{bmatrix}
2 & 2 & 2 \\
2 & 2 & 2 \\
2 & 2 & 2 \\
2 & 2 & 2 \\
2 & 2 & 2 \\
2 & 2 & 2 \\
2 & 2 & 2 \\
2 & 2 & 2 \\
2 & 2 & 2 \\
2 & 2 & 2 \\
\end{bmatrix}
\]

#### Overall matrix
\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]
V. DESIGN PATTERN DETECTION USING SUBGRAPH ISOMORPHISM

The main objective of this project is to detect design pattern in given system. We have UML diagrams corresponding to system design and design pattern and then we can calculate overall matrices corresponding to them (by above given procedure). Thus one matrix for system design and another for design pattern. First we non-deterministically guess any design pattern (or subgraph) of system design pattern and then by applying equation (3) it can be find out whether they really are sub-isomorphic or not or design pattern exists or not in a particular system design.

Let us consider the overall matrix corresponding to system design (Fig. 10) i.e. matrix S, and guess any subgraph of it.

A. Design Pattern Detection as Composite Design Pattern

First we guess that Composite Design Pattern (Fig. 9) exists in system design (or can say that there is subgraph isomorphism is in between them). The overall matrix is (Fig. 11)

\[
P = \begin{bmatrix}
1 & 1 & 1 \\
2 & 1 & 1 \\
6 & 1 & 1
\end{bmatrix}
\]

Let there is a permutation matrix \( P \) of order of system design i.e. 9x9,

\[
P = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
P^T = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

Calculation of \( PS^T \) is shown below.

\[
PS^T = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
6 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

After eliminating entries from 4th row and 4th column we have reduced matrix as (because Composite Design Pattern is of 3 x 3 orders)

\[
PSP^T = \begin{bmatrix}
1 & 1 & 1 \\
2 & 1 & 1 \\
2 & 1 & 1
\end{bmatrix}
\]

This is not the same as Composite design pattern’s overall matrix. Now if we take permutation matrix as shown below

\[
P = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
P^T = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
PSP^T = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
6 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

After eliminating entries from 4th row and 4th column we have reduced matrix as shown below.
This is the same as of Composite Design Pattern. So subpart of system design has a carbon copy as a composite design pattern in system design or can be said that Composite design pattern is detected in system design.

B. Particular Design Pattern may or may not exist

From above, we observe the example of design pattern existence but it can be possible that a particular design pattern does not exist in system design. In this case there will be no permutation matrix for which we can find out (after row and column elimination) a matrix which is equivalent to design pattern matrix.

VI. RELATED WORK

The first effort towards automatically detect design pattern was achieved by Brown [9]. In this work, Smalltalk code was reverse-engineered to facilitate the detection of four well-known patterns from the catalog by Gamma et al. [1].

Antoniol et al. [10] gave a technique to identify structural patterns in a system with the purpose to observe how useful a design pattern recovery tool could be in program understanding and maintenance.

Nikolaos Tsantalis [2], proposed a methodology for design pattern detection using similarity scoring. But the limitation of similarity algorithm is that it only calculates the similarity between two vertices, not the similarity between two graphs. To solve this Jing Dong [3] gave another approach called template matching, which calculates the similarity between subgraphs of two graphs instead of vertices.

S. Wenzel [11] gave the difference calculation method works on UML models. The advantage of difference calculation method on other design pattern detecting technique is that it detects the incomplete pattern instances also.

VII. CONCLUSIONS

Three approaches for design pattern detection using graph matching have been discussed here in GIS domain. In this we took the model graph and a data graph (corresponding to design pattern), and tried to find out whether design pattern exists or not in model graph. There are 23 GoF (Fang of Four) [1] design patterns. The Graph Distance approach determined how much similar the two nodes are. But the drawback of this method is that it only concerned about node similarity not the whole graph. By applying normalized crossed correlation it can be determined the particular template (design pattern) exactly matches or partially matches to subpart of system design or not. Subgraph isomorphism identification technique, using overall matrix, reduces the complexity and tried to find out whether a particular design pattern exists in system design.