International Journal of Engineering Science Invention (IJESI)
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org ||Volume 7 Issue 9 Ver II || Sep 2018 || PP 61-67

Graph Theory Based Software Clustering Algorithm

M. Amudhan[1], K. Sangavai[2]
[1](*Department of Computer Science and Engineering, PSG College of Technology, India)*
[2](*Assistant Professor, Department of Mathematics, PSG College of Technology, India)*
Corresponding Author: M. Amudhan

Abstract : *Modern software systems are scaling up manifold in terms of size as well as complexity. The ever demanding field of software engineering makes software project development even more challenging. The implementation of such software associates an extensive amount of classes or subsystems. Clustering is a worthwhile method to group data entities. Graph theory can aid us in the understanding of the abstract layer of software. In this paper, a novel graph theory based software clustering algorithm is proposed.*
Keywords –*Graph theory, Algorithms, Software Clustering, Degree preserving spanning tree*

Date of Submission: 07-09-2018 Date of acceptance: 24-09-2018

I. INTRODUCTION

Software engineering is the systematic application of engineering to the development of software. It was introduced to address the issues of low-quality software projects. Software design is a process to transform user requirements into some suitable form, which helps the programmer in coding and implementation of the software. Software design is the first step in the Software Development Life Cycle (SDLC). Object–Oriented Analysis (OOA) is the procedure of identifying requirements and developing specifications in terms of a software system's object model, which contains interacting objects where the object is an element in an object–oriented environment that may have a physical or a conceptual existence [8, 10].

1.1. Software Design and Class Diagram

A class represents a collection of objects having the same characteristic properties and exhibit common behavior. Any complex system is better understood by representing them in the form of diagrams. Class diagrams basically represent the object-oriented view of a system, which is static in nature. It shows relationships between classes, objects, attributes, and operations. Unified Modeling Language (UML) is a standardized modeling language used to construct and visualize commodities of a software system. Such UML diagrams help us understand the system in a straightforward and improved way. UML defines diverse diagrams to cover most of the aspects of a system. Class diagrams can be flawlessly mapped to graphs where vertices represent classes, while edges correspond to a specific kind of relationship like association, generalization or composition [3, 4].

1.2. Graph theory and Spanning trees

A graph G = (V, E) is a mathematical structure consisting of two finite sets V and E. The elements of V are called vertices (or nodes), and the elements of E are called edges. Each edge has a set of one or two vertices associated to it, which are called its endpoints. Graphs are visually represented by dots (vertices) connected by lines (edges). Graph theory is the study of graphs which are used to model pairwise relations between objects. A colossal amount of social, technological and biological networks can also be represented as graphs. A spanning tree of a graph G is a connected acyclic subgraph which includes all the vertices of G. Spanning trees play an important role in the area of research under graphs and networks due to their nature of minimally connecting subgraph. There are many kinds of spanning trees available in the literature of graph theory. One such special kind of spanning trees called Vertex Subset Degree Preserving Spanning Trees was introduced and extensively studied by Anitha. R and Sangavai. K in [2]. Vertex Subset Degree Preserving Spanning Tree is defined as a spanning tree T of the graph G (V, E) such that $\deg_T(v_i) = \deg_G(v_i)$ for all v_i in A, which is a nonempty subset of V and is denoted as A-DPST. This new class of spanning trees has many applications in various kinds of networks and one such application in sensor networks is established in [12]. In [12], the authors proposed a distance based routing algorithm for a sensor network using A-DPST. For all the definitions and concepts used in this paper related to graph theory, we referred [9].

1.3. Graphs and Computer Science

Graphs have been useful in plentiful fields of computer science. In the software development lifecycle, during Requirements Specification, Data Flow Diagrams (DFDs) are constructed. These are essentially graphs in which the vertices represent transformations and edges characterize data flows. Finite State Machines and Petri Nets are used in capturing the requirements of synchronous and asynchronous systems due to the alluring graphics scheme. During design, Graphical Design Notation (GDN) is essentially a graph used for describing relations among modules. In such graph representations, directed edges represent the dependency of one software component on another. During testing, the control flow of a program uses directed graphs to represent the sequence of instruction execution. Also, Software process management has been aided by the use of network diagrams for calculating early start and late finish dates (CPM and PERT techniques) [5].

1.4. Class diagrams

Class diagrams can be perfectly mapped to graphs where vertices represent the classes, while edges correspond to a selected type of relationship (e.g. association, generalization, composition, etc.) Let us assume a graph G = (V, E) represents the class diagram of the object-oriented system. V is the set of vertices corresponding to the classes of the system while E is the set of all edges representing a particular kind of relationship between the classes. For example, the process of a sub-class inheriting the functionality of a superclass is known as generalization. It is symbolized with a straight connected line with a closed arrowhead pointing towards the superclass. If generalizations are to be represented, a directed edge (p, q) ∈ E indicates the inheritance of p from q [4]. Consider the simple design in Figure 1.1.

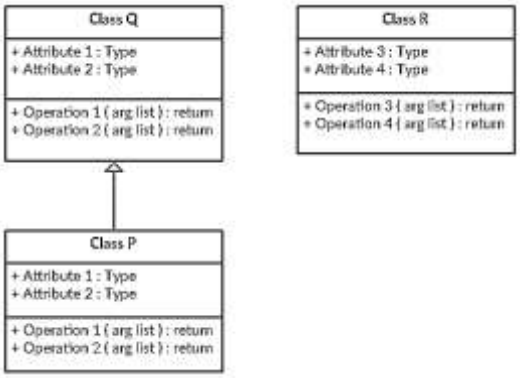


Figure 1.1

The graphical representation and the generalization matrix of the system design in Figure 1.1 is given in Figure 1.2

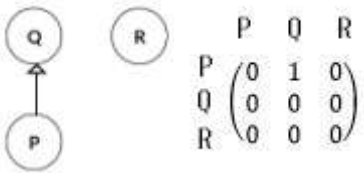


Figure 1.2

1.5. Clustering

Clustering is the method of recognizing an analogous group of entities in data. Entities within a cluster have the same characteristics and differ from entities in other clusters. Clustering algorithms are applied for software modularization. Modularizing a software is done by grouping together related software entities, thereby providing a sophisticated understanding of the system. Software designers incline to design modules such that they can be executed and compiled separately and independently. The modular design follows 'divide and conquer' problem-solving strategy. In the OO systems domain, clustering can be viewed as the process of partitioning the system into sets of strongly communicating classes or hierarchy of classes. Such dense communities of classes exhibiting intense interaction in terms of method invocations might imply relevance of functionality or even possible reusable components [11]. Clustering can also be helpful in reducing the search space of algorithms that seek to identify patterns or specific structures within an OO system. There are several

methods of software clustering available in the literature. Among those, spectral graph partitioning techniques first appeared in the early seventies in the research work of Donath and Hoffman [5] and Fiedler [6], [7]. They explored the properties of the algebraic representations of graphs (Adjacency Matrix, Laplacian Matrix) and introduced the idea of using Eigen vectors of the Laplacian Matrix to partition graphs. In [1], Alexander Chatzigeorgiou et.al proposed the method of software clustering using the concept of the Laplacian matrix of the graph corresponding to a system.

II. Proposed Algorithm

In general, software clustering aims at partitioning a software system into subsets of modules/components, so that the modules in each cluster share some common trait. A common criterion for partitioning is to come up with clusters that exhibit high consistency and low coupling. Therefore, we require disjoint clusters of the classes. In this paper, we propose a novel clustering algorithm using the concept of vertex subset degree preserving spanning trees. Using the concept of A-DPST we form a spanning forest in the graph corresponding to the class diagram through which we are achieving our goal of forming clusters. The advantage of our proposed algorithm is the identification of cluster heads also along with clustering. These cluster heads are a distinctive member of the respective cluster, based on their importance in the given system. The flow of execution of our algorithm is described in the flowchart in Figure 2.1.

2.1. Flow Chart

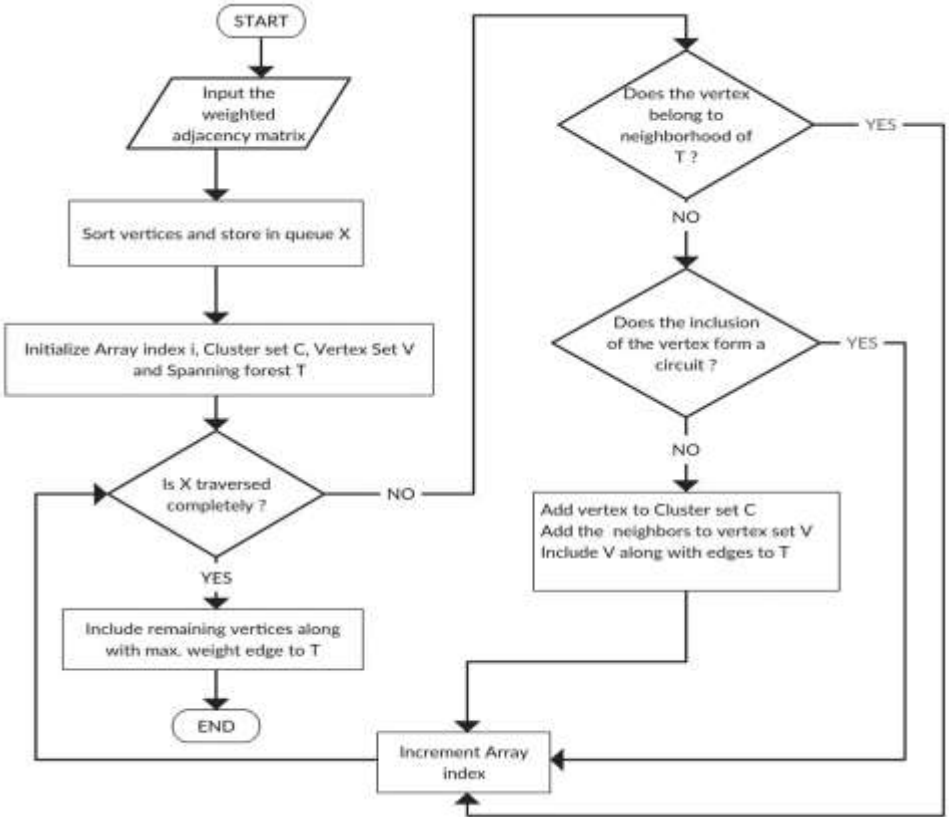


Figure 2.1

2.2 Proposed Algorithm

This algorithm starts with the formation of the adjacency matrix of the graph corresponding to the class diagram of the system by writing 0's in the non-adjacent positions of the vertices and corresponding edge weights in the adjacent positions. X is the array of all vertices along with their weights, which are calculated as the sum of the edge weights of those incidents onto it (respective row/column sum in the adjacency matrix) in the graph. x_i is the i^{th} vertex in X. V_T is the set of all vertices that are already visited. N[T] is the closed neighborhood of vertices in T. At the end of the algorithm's progress, X will be empty and V_T will contain all the vertices of the graph. T becomes a spanning forest of the graph whose components are distinct clusters and C is the set of all cluster heads.

Algorithm:

Input: Weighted adjacency matrix of the graph G (V, E) corresponding to the given class diagram of an object-oriented system containing n classes.
Output: Clustered classes with cluster heads.

1. Sort the vertices in a non-increasing order, based on their weight and store them in array X
2. Initialize i =0, V_T=Ø, C=Ø,and T=Ø
3. Consider the vertex x_i
4. If $x_i \notin N[T]$ and the inclusion of x_i does not form a circuit, then

i. $C = C \cup \{x_i\}$
ii. $V_T = V_T \cup \{N[x_i]\}$
iii. Include V_T along with their edges to T

5. Increment i and repeat step4 until the array X is traversed completely.
6. For any vertex x_i not in T,
   Include x_i along with the maximum weighted edge.
7. T contains all the clusters and C contains all cluster heads.

2.3. Illustration of the Algorithm

To demonstrate this clustering technique let us consider a hypothetical object-oriented system consisting of an Airport model system given in Figure 2.2. In this system, there are totally 11 classes with their own attributes and methods. All these classes communicate among themselves to ensure the effective functioning of the system. This system is modeled as a graph by representing each class as a vertex and their relationship as an edge between the corresponding vertices. The weight associated with the edges is based on the multiplicity. Multiplicity is a definition of the cardinality of some collection of elements by providing an inclusive interval of non-negative integers to specify the allowable number of instances of the described element. Multiplicity interval has some lower bound and upper bound. Here, we assigned the multiplicity upper bound of the relation as the edge weight.

For example, the association between the classes, 'Plane Model' and 'Seat' has a multiplicity upper bound of 250. This means an individual plane model may contain a maximum of 250 seats. Hence the edge connecting the vertices representing these two classes is assigned a weight of 250.

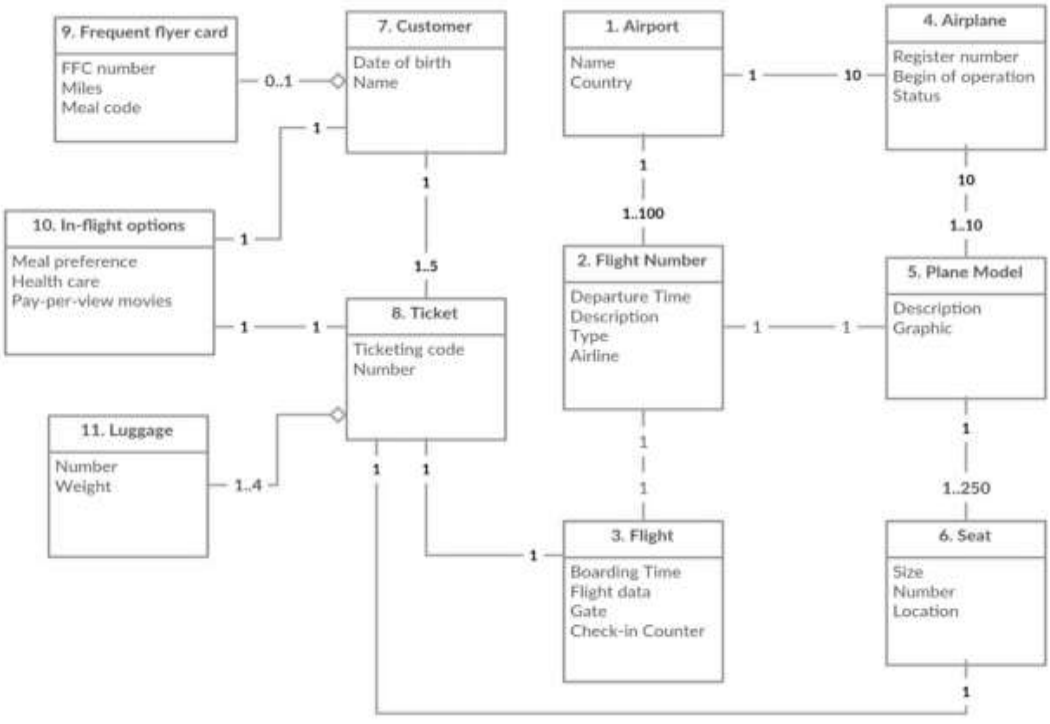


Figure 2.2

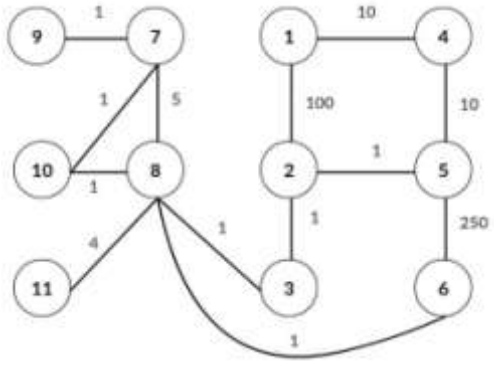


Figure 2.3

The following is the table of the vertices and their weights

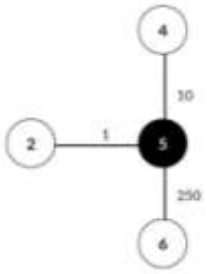
Vertex	1	2	3	4	5	6	7	8	9	10	11
Weight	110	102	2	20	261	251	7	12	1	2	4

Table 2.1

After sorting the vertices in non-increasing order of their weight, the array X becomes,
X = {5, 6, 1, 2, 4, 8, 7, 11, 3, 10, 9}

Iteration 1:

Initially,
i = 0, V_T = Ø, C = Ø and T = Ø
x_0= 5
C= {5}
V_T= {5, 2, 4, 6}



Iteration 2:

i = 1, V_T = {5, 2, 4, 6}, C = {5}
x_1= 6
$x_1 \in N[T]$
Hence ignore.

Iteration 3:

i = 2, V_T = {5, 2, 4, 6}, C = {5}
x_2 = 1
Adding x_2 will form a circuit.
Hence ignore.

Iteration 4:

i = 3, V_T = {5, 2, 4, 6}, C = {5}
x_3 = 2
$x_3 \in N[T]$
Hence ignore.

Iteration 5:

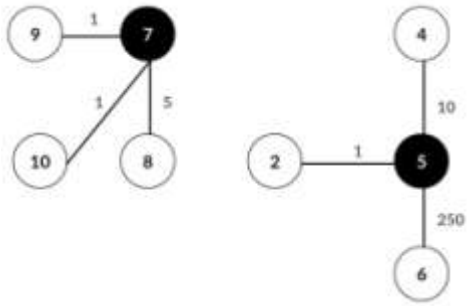
i = 4, V_T = {5, 2, 4, 6}, C = {5}
x_4 = 4
$x_4 \in N[T]$
Hence ignore.

Iteration 6:

i = 5, V_T = {5, 2, 4, 6}, C = {5}
x_5 = 8
$x_5 \in N[T]$
Hence ignore.

Iteration 7:

i = 6, V_T = {5, 2, 4, 6}, C = {5}
x_6= 7
C= {5, 7}
V_T= {5, 2, 4, 6, 8, 9, 10}



Iteration 8:

i = 7, V_T = {5, 2, 4, 6, 7, 8, 9, 10}, C = {5, 7}
x_7= 11
$x_7 \in N[T]$
Hence ignore.

Iteration 9:

i = 8, V_T = {5, 2, 4, 6, 7, 8, 9, 10}, C = {5, 7}
x_8= 3
$x_8 \in N[T]$
Hence ignore.

Iteration 10:

i = 9, V_T = {5, 2, 4, 6, 7, 8, 9, 10}, C = {5, 7}
x_9= 10
$x_9 \in N[T]$
Hence ignore.

Iteration 11:

i = 10, V_T = {5, 2, 4, 6, 7, 8, 9, 10}, C = {5, 7}
x_{10}= 9
$x_{10} \in N[T]$
Hence ignore.

Iteration 12:

And the process ends by connecting the other remaining vertices 1, 3 and 11 using the maximum weight edge to either of the clusters. Therefore, the vertex set of the resulting tree V_T = {5, 2, 4, 6, 7, 8, 9, 10, 1, 3, 11} = V
And the cluster head set becomes C = {5, 7}
The resulting T which is a spanning forest, is in Figure 2.4. Therefore, the classes are clustered into two, whose cluster heads are 5 and 7 respectively.

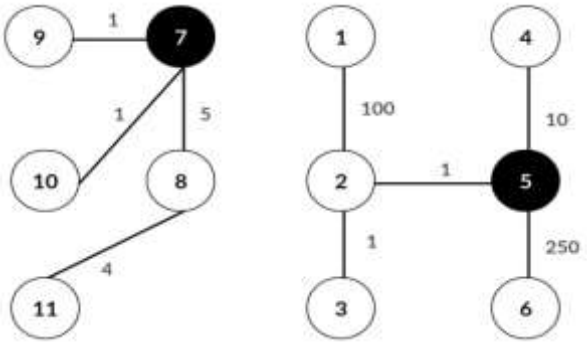


Figure 2.4

III. Implementation And Sample Output

In this section, the performance result of the proposed clustering algorithm is presented. The algorithm is implemented in Java. The weighted adjacency matrix of the graph corresponding to the class diagram of the Airport model system is given as the input. By using which the graph associated with the system is generated. In this graph, the nodes represent the different classes of the system and the edges correspond to the relationship between the corresponding vertices. The weights associated with the edges are the multiplicity upper bound. After running our algorithm, the graph is clustered into two clusters whose cluster heads are node 5 and node 7. The outputs after the run of our algorithm on Airport model system obtained using JFrame given in Figure 3.1.

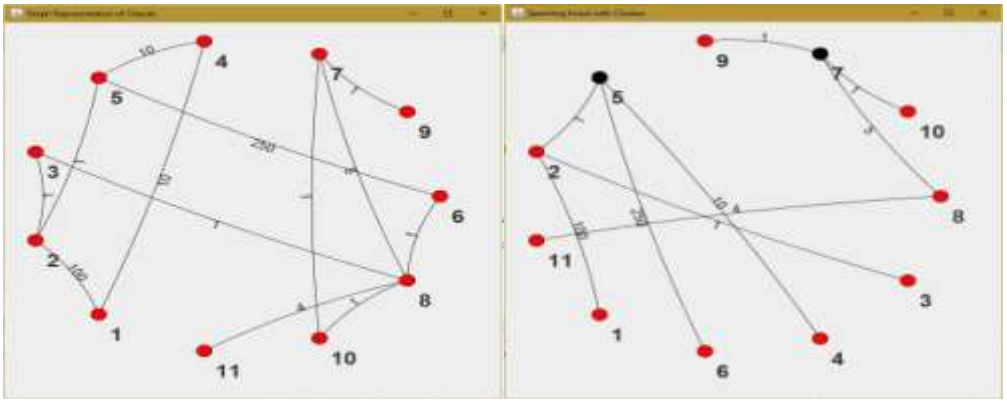


Figure 3.1(a) **Figure 3.1**(b)

Figure 3.1(a) represents the graph and Figure 3.1(b) represents the spanning forest with two clusters whose cluster heads are node 5 and 7 respectively

.

IV. Conclusions And Future Work

In the software domain, a significant application of cluster analysis is to modularize a software system by grouping together software entities that are similar or related to each other. During development, most of the effort is usually devoted to the understanding of the software system. This task is facilitated if a system is well clustered, making it easier to change and evaluate the side effects of a change. In this paper, we have proposed a novel software clustering algorithm using the concept of vertex subset degree preserving spanning trees. We also illustrated by considering an Airport model system. This algorithm is implemented in JAVA and its correctness is verified by applying it to the Airport model system. The other concepts of graph theory like dominating sets, independent sets etc. could also be used for serving the similar needs in software design.

References

[1]. Alexander Chatzigeorgiou, Nikolaos Tsantalis, George Stephanides, "Application of Graph Theory to OO Software Engineering", WISER '06 Proceedings of the 2006 international workshop on interdisciplinary software engineering research, 2006.
[2]. Anitha. R and Sangavai. K, "On Vertex Subset Degree Preserving Spanning Trees", International Journal of Computational and Applied Mathematics, Vol. 2, No. 2, pp. 115–123, 2007.
[3]. Booch G, Maksimchuk R A, Engel M W, Young B J, Conallen J, Houstan K A, "Object Oriented Analysis and Design with Applications", Third edition, Addison-Wesley, 2007
[4]. Booch G, Rumbaugh J and Jacobson I, "The Unified Modeling Language User Guide", Second edition, Addison Wesley Professional, 2005.
[5]. Donath, W. E., and Hoffman, A. J. Lower bounds for the partitioning of graphs. IBM Journal of Research and Development, 17, (Sep. 1973), pp. 420-425.
[6]. Fiedler, M. Algebraic connectivity of graphs. Czechoslovak Mathematical Journal, 23(98), (1973), pp. 298-305.
[7]. Fiedler, M. A property of eigenvectors of non-negative symmetric matrices and its applications to graph theory. Czechoslovak Mathematical Journal, 25(100), (1975), pp. 619-633.
[8]. Ghezzi, C., Jazayeri, M., and Mandrioli, D. Fundamentals of Software Engineering. 2nd edition, Prentice Hall, Upper Saddle River, NJ, 2003.
[9]. Jonathan Gross, Jay Yellen, "Graph Theory and Its Applications", CRC Press, New York, 2006.
[10]. Roger S Pressman, "Software Engineering - A practitioner's approach", McGraw Hill International Edition, Singapore, 2009
[11]. Roman Bazylevych, Roman Burtnyk, "Algorithms for software clustering and modularization", 2015 Xth International Scientific and Technical Conference "Computer Sciences and Information Technologies" (CSIT), 2015.
[12]. Sangavai. K and Anitha. R, "Application of Vertex Subset Degree Preserving Spanning Trees in Sensor Networks", Discrete Mathematics, Algorithms and Applications Vol.2, No. 3 277–289, 2010.

M. Amudhan "Graph Theory Based Software Clustering Algorithm "International Journal of Engineering Science Invention (IJESI), vol. 07, no. 09, 2018, pp 61-67