A Framework to Investigate and Evaluate Genetic Clustering Algorithms for Automatic Modularization of Software Systems

Saeed Parsa and Omid Bushehrian

Iran University of Science and Technology , Narmak , Tehran , Iran
{Saeed_Parsa,Boushehrian}@yahoo.com

Abstract. In this paper a software environment, called DAGC, is described. The main idea behind the design of DAGC is to facilitate research works in design and development of genetic clustering algorithms for automatic re-modularization of software systems. Within the DAGC environment, clustering algorithms may be assembled or modified by simply selecting the parts from an extendable list of components. Using this distinguishing feature of the DAGC framework, a new algorithm with a new encoding and crossover operator was evolved.

Keywords: Genetic Clustering, encoding Algorithm, Automatic Modularization

1 Introduction

Automatic clustering algorithms are used within the context of program understanding to discover the structure (architecture) of the program under study. Clustering is a key activity in reverse engineering to discover a better design of the systems [3, 4]. Genetic algorithms are widely believed to be effective on NP-complete global optimization problems, such as clustering, and they can provide good sub-optimal solutions in reasonable time [1].

Well known frameworks such as Bunch [4], CRAFT [2] and GAME [6] have provided environments to run and view the results of clustering on software systems. It should be noted that even if there are reusable software components for GA, the success of respective implementations still depends on appropriate definitions of the concepts for the specific problem. Our objective has been to develop an environment to experiment with the effects of applying different schemes for the components of genetic clustering algorithms for software re-modularization. To achieve this, we have developed a flexible software environment called DAGC.

2 Genetic Clustering

The idea has been to develop a software environment to facilitate researchers' investigations on development of optimal genetic clustering algorithms for automatic

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3037, pp. 699–702, 2004.

re-modularization of software systems. To achieve this, a comprehensive study of the existing algorithms was carried out [1, 4, 5]. We arrived with the results listed below:

1. There are a fixed set of component types, mostly used in these algorithms.
2. For each component type, a standard interface could be defined.
3. Using standard interfaces, various schemes may be applied to define the components without any need to change the existing source code for the algorithm.
4. For each component, a number of implementations could be provided.
5. A general clustering algorithm, calling the components could be developed.
6. New algorithms could be created by selecting the components.

In general, there are a fixed set of operators and components appearing in genetic clustering algorithms. DAGC makes use of a generalized genetic clustering algorithm. By defining a function with a standard interface for each of the component types, various schemes for the components may be selected or created within the DAGC, without any need to change the body of the algorithm.

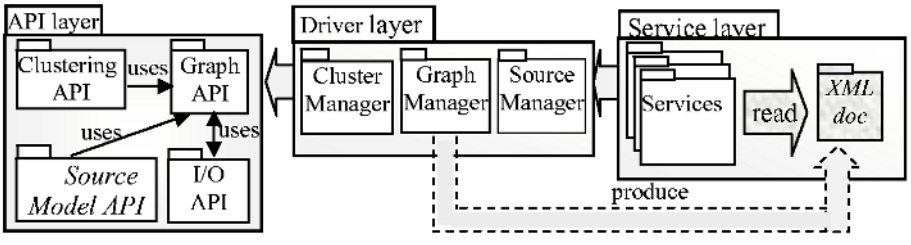

Fig. 1. The Architecture of the DAGC Framework

3 Architecture

As shown in figure 1, the DAGC architecture consists of three interacting layers: API, Driver and Services.

The API layer is an independent layer which includes a useful and complete set of java classes. These classes are arranged in four packages as follows:

- ClusteringAPI: The ClusteringAPI package provides useful classes and interfaces for the genetic components.
- GraphAPI: The GraphAPI package provides a set of classes which can be used to generate and display three kinds of benchmark graphs called Random, Caterpillar and software graphs [1].
- I/O API: This package provides a number of useful classes for loading and writing graph files. Here, we support three formats for importing or exporting graph files which are: Dotty-AT&T, XML and Text.
- SourceModelAPI: This package contains a number of classes to extract call graphs from a Java source code.

The service layer provides all the interfaces of the DAGC framework. At this moment DAGC presents the following services:

- Clustering Service: This service allows the user to create a new GA.

- Data Analyzer Service: This service analyzes the clustering results
- Customizing Service: This service provides some templates for genetic components and let us to customize them for building our new components.

4 DAGC Clustering Algorithm

A new algorithm, called DAGC was developed within the DAGC environment by simply substituting the parts of an existing algorithm called Bunch [4], with the corresponding components, selected from an extendable list of components. Substituting a component, the algorithm was executed within the environment on random graphs or class dependency graphs extracted from a given source code.

The DAGC clustering algorithm makes use of our new encoding and recombination schemes. In our new encoding scheme, each chromosome is a permutation of N integers. Here, the m^{th} gene of the chromosome, instead of holding a partition number k (which means node m of the graph resides in partition k), holds a value 1=<p<=N. To decode a chromosome into a clustering the following procedure can be used:

```
Suppose Array C holds a permutation of graph nodes 1 to N
  if C[i] >= i then Create a new cluster and assign node i to it.
  Else For i=1 to N do Assign node i to the same cluster as node C[i]
```

Since our coding is a permutation, we need an order based recombination operator to preserve the permutation form in each offspring. Here, a one point recombination operator is used. Suppose two parents P_1 and P_2 such that:

$P_1 = C_1C_2\ldots C_kC_{k+1}C_{k+2}\ldots C_L$ and $P_2 = D_1D_2\ldots D_kD_{k+1}D_{k+2}\ldots D_L$

To create two offspring O_1 and O_2, first, parents are split at position k. First parts of two offspring are created using the first parts of two parents as follows:

$O_1 = C_1C_2\ldots C_k$ and $O2 = D_1D_2\ldots D_k$

To fill the second parts of two offspring, we create the following list:

$C_{k+1}\ D_{k+1}C_{k+2}\ D_{k+2}\ldots C_L\ D_L$

Starting from the first element in the list, if the element does not belong to the first offspring, O_1, it will be augmented to O_1. Otherwise, it will be added to the second offspring, O_2. Analogously, we can consider the list $D_{k+1}\ C_{k+1}D_{k+2}\ C_{k+2}\ldots D_L\ C_L$ and generate two other offspring.

5 Evaluating DAGC Algorithm

To compare our new genetic algorithm, DAGC, with the Bunch [4], 9 different random graphs [1] with more than 200 vertices were used. Each graph was clustered 5 times, using both the DAGC and Bunch. The average and variance of the clustering results are shown in figure 2. Figure 2.b demonstrates the stability of the DAGC encoding scheme in comparison with the encoding scheme used by the Bunch.

To evaluate the performance of our permutation-based encoding scheme, the TurboMq fitness function was used. A comparison of the quality of the clustering results is presented in figure 2.a. Since the GA search space is highly reduced by using our permutation-based coding scheme, better clustering quality results were produced.

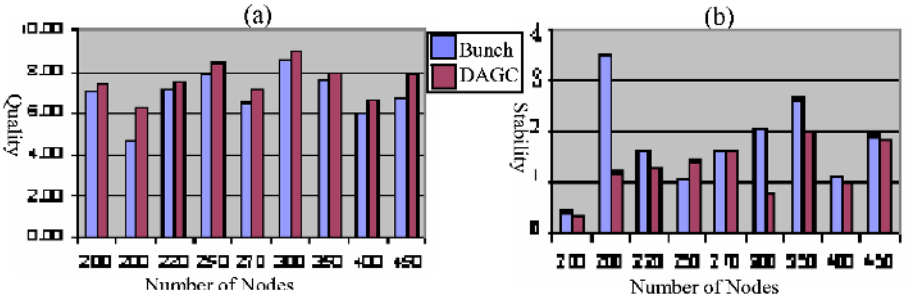

Fig. 2. Comparison of the results

6 Conclusions

In this paper we showed how a flexible environment for investigating clustering algorithms for re-modularization of software systems can be created. Using this environment, researchers can assemble new clustering algorithms by trying different schemes for the components and observing the effects of changing each component. The environment is capable of keeping different schemes for the components. Thereby, the components can be easily selected to assemble new algorithms. Using these distinguishing features of the DAGC, an efficient clustering algorithm was evolved by trying different schemes for the components of the Bunch genetic clustering algorithm. Since the GA search space is highly reduced by using our permutation-based coding scheme, better clustering quality results were produced.

References

1. Bui T. N., Moon B. R., "Genetic algorithm and graph partitioning," *IEEE Trans. Comput.*, vol. 45, pp. 841–855, July 1996.
2. Brian S. Mitchell, Spiros Mancoridis,"CRAFT: A Framework for Evaluating Software Clustering Results in the Absence of Benchmark Decomposition", IWPC, IEEE 2001.
3. A. F. Brito,"A Coupling Guided Cluster Analysis Approach to Reengineer the Modularity of Object Oriented Systems", Conf. on Soft. Maintenance and Reengineering, IEEE 2000.
4. Brian S. Mitchell,"Bunch: A Clustering tool for the Recovery and Maintenance of Software System Structure", International Conf. of Software Maintenance, IEEE 1999.
5. Cincotti A., Cuttelo V., Pavone M., "Graph Partitioning using Genetic Algorithms with ODPX", Proceedings of the World Congress on Computational Intelligence, IEEE 2002
6. Gokel N., Drechsler R., Becker B., "GAME: A Software Environment for Using Genetic Algorithms in Circuit design", Applications of Computer Systems, 240-247, 1997.
7. Brian S. Mitchell, Spiros Mancoridis, "Comparing the Decompositions Produced by Software Clustering Algorithms Using Similarity measurements ",in the proceedings of international conference on software maintenance (ICSM'01),Italy, IEEE 2001
8. David L. Woodruff, Optimization Software Class Libraries, Handbook, March 2002.