GMADCH: H-Design Modularization for Software Systems with Incoherent Call Graphs

Masoud Azizi[1], Dr. Habib Izadkhah[2], Prof. Ayaz Issazadeh[3]
[1,2,3]Department of Computer Science, Tabriz University
mablue92@tabrizu.ac.ir, izadkhah@tabrizu.ac.ir, isazadeh@tabrizu.ac.ir

October 2025

Abstract

Benchmarking and modularizing software systems with incoherent call graphs is a challenging nonlinear problem with deep implications for maintainability and architecture. GMADCH introduces a novel H-design algorithm based on Levenshtein distance and vocabulary congruence, enabling logical grouping by conceptual similarity. This article presents the mathematical foundation, algorithmic details, evaluation, and comparison with prior graph-based and clustering approaches.

1 Introduction

Modern software systems often contain modules with weak or absent interconnections, resulting in incoherent call graphs. Graph-based modularization approaches (e.g., [1, 2]) struggle with such systems, resorting to random assignment or heuristic clustering. GMADCH offers a text-based similarity approach, leveraging Levenshtein distance and vocabulary congruence to enable logical, maintainable clustering.

2 Related Work and Background

Software modularization, clustering, and architecture recovery have been extensively studied. Graph-based algorithms (e.g., depth-based, hierarchical, evolutionary) and string matching methods (e.g., Levenshtein, Jaccard, Ellenberg) form the foundation for systems like GMADCH. See [1, 4, 5, 7, 8, 2] for core methodologies.

3 Definitions

3.1 Call Dependency Graph

A call dependency graph $G = (V, E)$ represents software entities (files, classes, functions) as nodes V and their relationships (calls, references) as edges E.

3.2 Levenshtein Distance

Given two strings s_1, s_2, the Levenshtein distance $d_L(s_1, s_2)$ is the minimum number of edit operations (insertions, deletions, substitutions) needed to transform s_1 into s_2 [3]. Formally,

$$d_L(s_1, s_2) = \begin{cases} |s_1|, & |s_2| = 0 \\ |s_2|, & |s_1| = 0 \\ \min \begin{cases} d_L(\mathrm{tail}(s_1), s_2) + 1 \\ d_L(s_1, \mathrm{tail}(s_2)) + 1 \\ d_L(\mathrm{tail}(s_1), \mathrm{tail}(s_2)) + [s_1[0] \neq s_2[0]] \end{cases} \end{cases} \tag{1}$$

3.3 Vocabulary Congruence Scoring (H-Design)

For each word w in a file, its conceptual H-design score is:

$$\mathrm{score}(w) = \mathrm{freq}(w) + \sum_{\substack{w' \in D \\ w' \neq w}} \frac{\mathrm{freq}(w')}{d_L(w, w')} \tag{2}$$

where D is the dictionary (global or user-provided).

4 Algorithm Description

4.1 Preprocessing

- Extract all words from code files, excluding short words and programming keywords.
- Build dictionary D (auto or user-provided list).

4.2 Scoring and Tag Selection

For each code file:

- Calculate $\mathrm{score}(w)$ for all words w.
- Select top-k tags (default $k = 3$) per file.

4.3 Folder Grouping and Clustering

Files are grouped by folder and shared tags, revealing conceptual clusters and aiding maintainability.

5 Mathematical Formalism

5.1 Similarity Matrix

Let S_{ij} represent similarity between entities i and j:

$$S_{ij} = 1 - \frac{d_L(w_i, w_j)}{\max_{i,j} d_L(w_i, w_j)} \tag{3}$$

This matrix underlies hierarchical clustering and modularization.

5.2 Modularization Quality

Following [1], modularization quality MQ:

$$MQ = \frac{i}{i + j} \tag{4}$$

where i is internal edges, j is external edges.

5.3 Clustering Algorithms

GMADCH relates to hierarchical clustering (UPGMA, WPGMA), centroid linkage, and K-means/K-medoids methods [4, 11, 12, 13].

6 Evaluation

We evaluated GMADCH on large open-source systems (e.g., Microsoft Calculator, financial trading platforms). Compared to previous graph-based and random modularization algorithms, GMADCH improved MoJo, MoJoFM, reduced clustering error, and enhanced maintainability.

6.1 Results

Algorithm	MoJo	MoJoFM	Time (s)
GMA (random)	37.3	22.3	0.017
GMADC (text-based)	37.2	22.5	0.76
GMADCH (H-design)	**34.7**	**29.2**	0.82

Table 1: Comparison of modularization quality and performance.

7 Limitations

- Levenshtein scoring can become computationally expensive for very large dictionaries; parallelization and filtering are used. - Tag selection may be impacted by code style and documentation quality. - Future work includes advanced similarity matrices and mapping to sparse graphs.

8 Future Work

- Applying Johnson's algorithm for sparse graphs [9]. - Refining similarity measures with semantic and structural code analysis. - Integrating clustering indices and optimizing for distributed computation.

9 Conclusion

GMADCH advances modularization for heterogeneous software, combining string metric theory and vocabulary analysis. The H-design approach connects entities by conceptual similarity, avoiding random clustering and enabling logical scaling of incoherent call graphs.

10 References

References

[1] H. Izadkhah, I. Elgedawy and A. Isazadeh, "E-CDGM: An Evolutionary Call-Dependency Graph Modularization Approach for Software Systems," Cybernetics and Information Technologies, pp. 70-90, 2016.

[2] B. Pourasghar, H. Izadkhah, A. Isazadeh and S. Lotf, "A Graph-based Algorithm for Software Systems Modularization by Considering the Depth of Relationships," 2020.

[3] V.I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," Soviet Physics Doklady, vol. 10, no. 8, pp. 707–710, 1966.

[4] R.R. Sokal, C.D. Michener, "A statistical method for evaluating systematic relationships," University of Kansas Scientific Bulletin, 1958.

[5] D. Gusfield, "Algorithms on Strings, Trees and Sequences," Cambridge University Press, 1997.

[6] S. Ducasse, D. Pollet, "Software architecture reconstruction: A process-oriented taxonomy," IEEE Transactions on Software Engineering, 35(4), 573–591, 2009.

[7] G. Navarro, "A guided tour to approximate string matching," ACM Computing Surveys, 33(1), 31–88, 2001.

[8] P. Andritsos, V. Tzerpos, "Information-theoretic software clustering," IEEE Transactions on Software Engineering, 31(2), 150–165, 2005.

[9] T.H. Cormen, C.E. Leiserson, R.L. Rivest, "Introduction to Algorithms," MIT Press and McGraw-Hill, 1990.

[10] V. Estivill-Castro, "Why so many clustering algorithms," ACM SIGKDD Explorations Newsletter, 4(1), 65–75, 2002.

[11] J. Macqueen, "Some Methods for Classification and Analysis of Multivariate Observations," Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, 1967.

[12] H. Steinhaus, "Sur la division des corps matériels en parties," Bull. Acad. Poland. Sci., 1957.

[13] E. Forgy, "Cluster analysis of multivariate data: efficiency versus interpretability of classifications," Biometrics, 1965.