ECEA 2019

5th International Electronic Conference on Entropy and Its Applications

18–30 November 2019

Chaired by Prof. Geert Verdoolaege

Sponsored by: entropy MDPI

Information Theoretic Objective Function for Genetic Software Clustering

Habib Izadkhah [1,*], and Mahjoubeh Tajgardan[2]

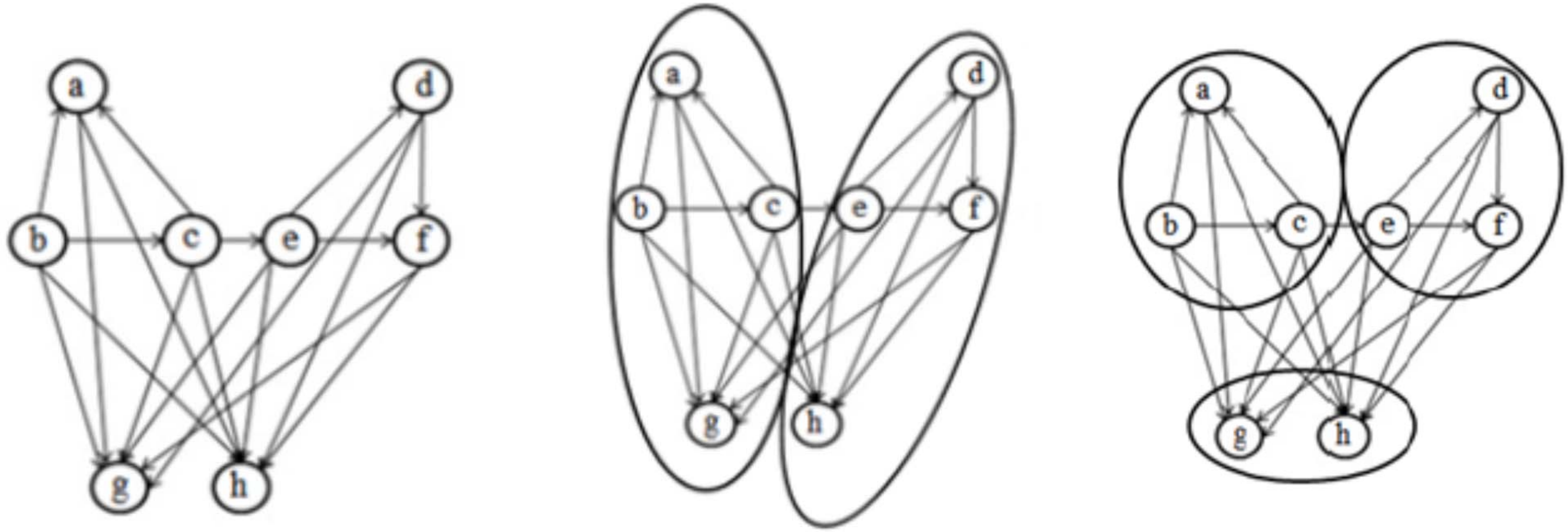
[1] Department of Computer Science, University of Tabriz, Tabriz, Iran;
[2] Department of Computer Science, University of Tabriz, Tabriz, Iran.

* Corresponding author: izadkhah@tabrizu.ac.ir

Abstract: Software clustering is usually used for program comprehension. Since it is considered to be the most crucial NP-complete problem, therefore, several genetic algorithms have been proposed to solve this problem. In the literature, there exist some objective functions (i.e., fitness function) which are used by genetic algorithms for clustering. These objective functions determine the quality of each clustering obtained in the evolutionary process of genetic algorithm in terms of cohesion and coupling. The major drawbacks of these objective functions are the inability to (1) consider utility artifacts, and (2) apply on another software graph such as artifact feature dependency graph. To overcome the existing objective functions limitations, this paper presents a new objective function. A new objective function is based on information theory, aiming to produce a clustering in which information loss is minimized. For applying the new proposed objective function, we have developed a genetic algorithm aiming to maximize the proposed objective function. The proposed genetic algorithm, named ILOF, has been compared to that of some other well-known genetic algorithms. The results obtained confirm the high performance of the proposed algorithm in solving nine software systems. The performance achieved is quite satisfactory and promising for the tested benchmarks.

Keywords: Clustering; Modularization; Genetic Algorithm;

Most search-based software clustering algorithms use Artifact Dependency Graph (ADG) (or Module Dependency Graph) for modeling a software system



The left figure comprises six program files namely a – f and two utility files namely g and h;

Libraries and drivers are examples of utility artifacts. Libraries provide services to many of the other artifacts, and drivers consume the services of many of the other artifacts. These files should be isolated in one cluster in the clustering process because they tend to obfuscate the software's structure.

In this paper, using information theory and the concept of entropy, a new objective function is proposed, which can solve the problems mentioned in the existing objective functions and improves the quality of clustering. The aim is to propose a new objective function that an evolutionary algorithm (e.g., genetic algorithm) can use to put artifacts with the minimum information loss into the same cluster.

The BasicMQ has five shortcomings, as follows:

- the execution time of BasicMQ is high, which restricts its application to small systems,
- unable to handle the ADGs with weighted edges,
- only considers cohesion and coupling in the calculation of the clustering quality,
- unable to handle the non-structural features,
- unable to detect utility artifacts.

$$BasicMQ = \frac{1}{k}\sum A_i - \frac{1}{\frac{k(k-1)}{2}}\sum E_{ij}$$

The TurboMQ has three drawbacks, as follows:

- only considers cohesion and coupling in the calculation of the clustering quality,
- unable to handle the non-structural features,
- unable to detect utility artifacts.

$$TurboMQ = \sum_{i=1}^{k} CF_i$$

$$CF_i = \frac{2\mu_i}{2\mu_i + \sum_{j=1}^{k} (\varepsilon_{i,j} + \varepsilon_{j,i})}$$

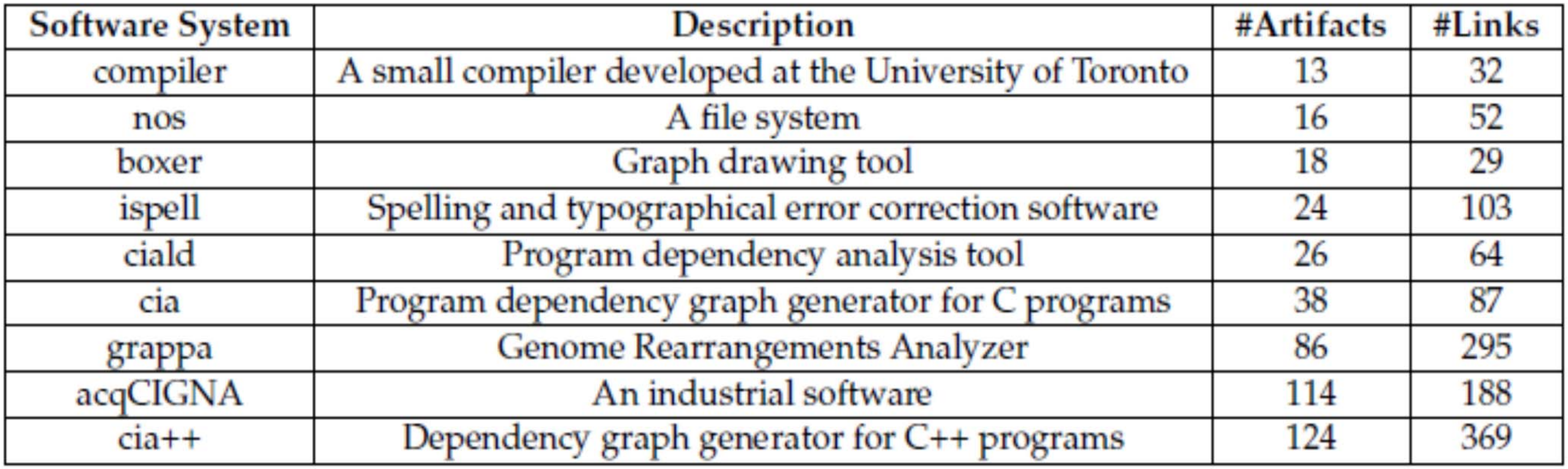
Most search-based software clustering algorithms use BasicMQ and TurboMQ as objective function.

In information theory, higher entropy reflects more uncertainty; in contrast, lower entropy represents more certainty. In the clustering problem, lower entropy is preferred. In the clustering of software, it is ideal that the selection probability of each feature of an artifact is the same before and after clustering.

We define information loss as follows:

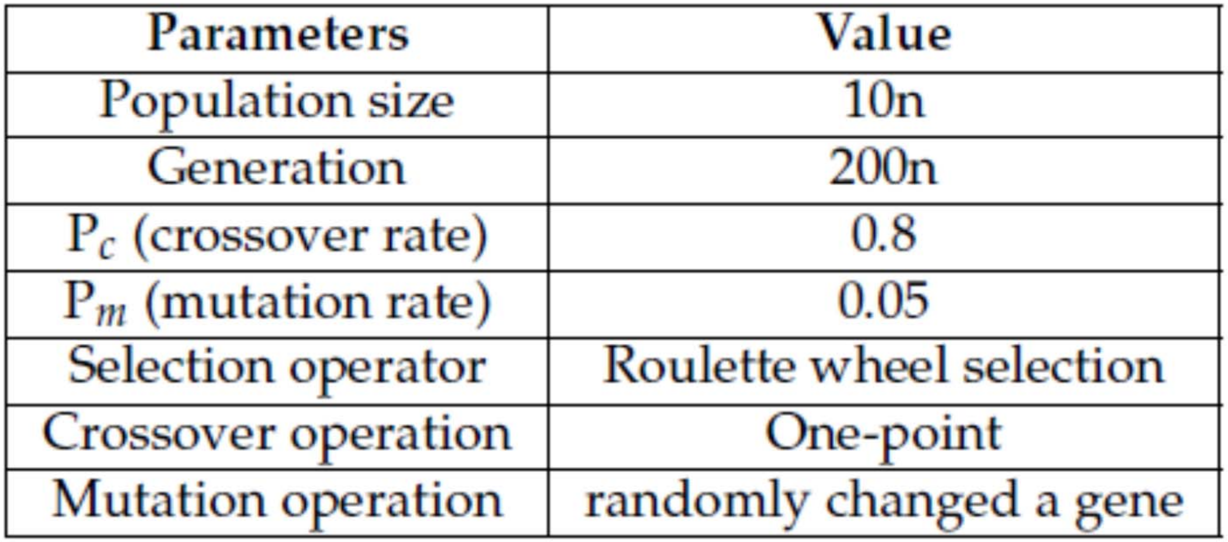
$$\delta I(a_i, a_j) \quad = \tfrac{1}{2}\sum_{f\in F} p(f|a_i)\log\frac{p(f|a_i)}{\overline{p}(f)} + \tfrac{1}{2}\sum_{f\in F} p(f|a_j)\log\frac{p(f|a_j)}{\overline{p}(f)}$$

The description of tested software systems



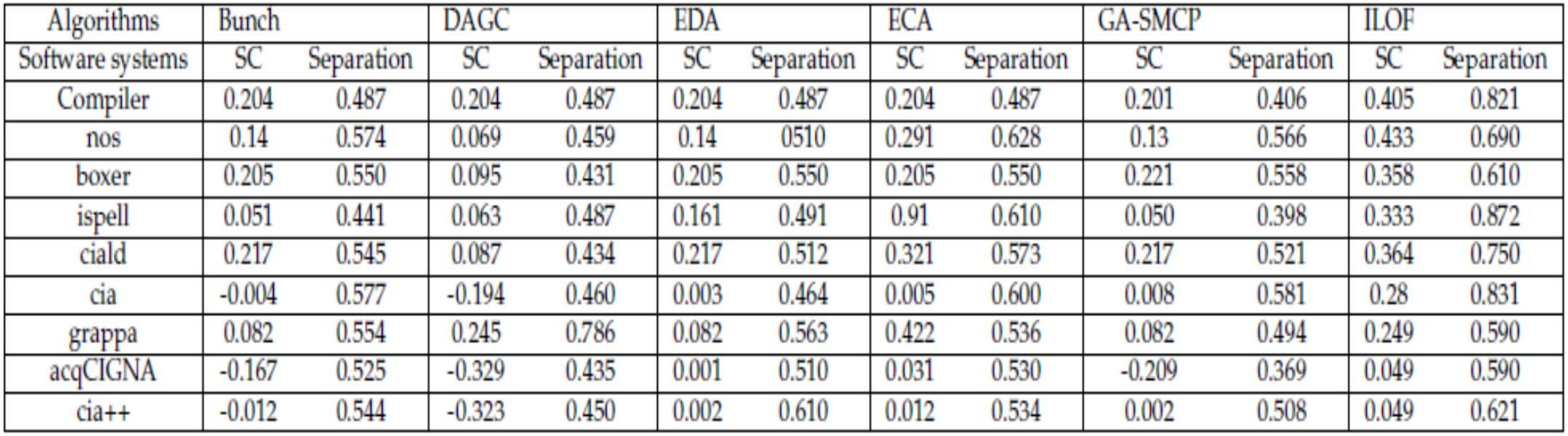
Software System	Description	#Artifacts	#Links
compiler	A small compiler developed at the University of Toronto	13	32
nos	A file system	16	52
boxer	Graph drawing tool	18	29
ispell	Spelling and typographical error correction software	24	103
ciald	Program dependency analysis tool	26	64
cia	Program dependency graph generator for C programs	38	87
grappa	Genome Rearrangements Analyzer	86	295
acqCIGNA	An industrial software	114	188
cia++	Dependency graph generator for C++ programs	124	369

The parameter setting for experiments



Parameters	Value
Population size	10n
Generation	200n
P_c (crossover rate)	0.8
P_m (mutation rate)	0.05
Selection operator	Roulette wheel selection
Crossover operation	One-point
Mutation operation	randomly changed a gene

Comparing the proposed algorithm against five search-based algorithms



Algorithms	Bunch		DAGC		EDA		ECA		GA-SMCP		ILOF	
Software systems	SC	Separation	SC	Separation	SC	Separation	SC	Separation	SC	Separation	SC	Separation
Compiler	0.204	0.487	0.204	0.487	0.204	0.487	0.204	0.487	0.201	0.406	0.405	0.821
nos	0.14	0.574	0.069	0.459	0.14	0510	0.291	0.628	0.13	0.566	0.433	0.690
boxer	0.205	0.550	0.095	0.431	0.205	0.550	0.205	0.550	0.221	0.558	0.358	0.610
ispell	0.051	0.441	0.063	0.487	0.161	0.491	0.91	0.610	0.050	0.398	0.333	0.872
ciald	0.217	0.545	0.087	0.434	0.217	0.512	0.321	0.573	0.217	0.521	0.364	0.750
cia	-0.004	0.577	-0.194	0.460	0.003	0.464	0.005	0.600	0.008	0.581	0.28	0.831
grappa	0.082	0.554	0.245	0.786	0.082	0.563	0.422	0.536	0.082	0.494	0.249	0.590
acqCIGNA	-0.167	0.525	-0.329	0.435	0.001	0.510	0.031	0.530	-0.209	0.369	0.049	0.590
cia++	-0.012	0.544	-0.323	0.450	0.002	0.610	0.012	0.534	0.002	0.508	0.049	0.621

Results and Discussion

The proposed algorithm, named ILOF, is compared on nine software systems with five algorithms in terms of silhouette coefficient, denoted by SC, and Separation. We chose the mean of results for each algorithm over 20 independent runs. The results demonstrate that the clustering achieved with ILOF are higher quality than those achieved with other algorithms for all the ADGs in terms of SC and separation.