Software Clustering using Hybrid Multi-Objective Black Hole Algorithm

Kawal Jeet  
Department of Computer Science  
D.A.V. College,  
Jalandhar, India  
kawaljeet80@gmail.com

Renu Dhir  
Department of Computer Science and Engineering  
Dr. B. R. Ambedkar N. I. T.,  
Jalandhar, India  
dhirr@nitj.ac.in

Abstract—Software clustering is the process of organizing software units into appropriate clusters so as to efficiently modularize complex program structure. In this paper, we investigate the use of hybrids of Black Hole algorithm (developed using weighted aggregation, auxiliary archive and Genetic Algorithm) to optimize multiple objectives for clustering of android mobile applications. It is empirically and statistically observed that multi-objective Black Hole algorithm when improved using Genetic Algorithm and auxiliary archive outperforms Two-Archive algorithm and its counterparts.

Keywords- bio-inspired algorithm, edgesim, nature-inspired algorithm, serach based software engineering, software clustering

I. INTRODUCTION

Human beings are always inspired by nature. Over the past couple of decades, a large number of complex research problems have found their solutions in nature-inspired algorithms such as Black Hole (BH) algorithm, Genetic Algorithm (GA) etc. BH algorithm [1] is inspired by the black hole theory of the universe and GA is inspired by Darwin’s survival of the fittest. Literature has a many instances where nature-inspired algorithms are applied to various fields of software engineering such as software testing [2], software effort estimation [3], and software clustering [4-7] etc. Software clustering refers to the placement of software units in an appropriate cluster which is useful to identify the cluster responsible for a particular functionality. It not only improves the structure of the system but also enhances the system comprehension. It is hence useful in both the development and maintenance of a software system [8].

Large numbers of companies are developing mobile applications for the users of their domain. The developers in these companies are in immense stress to produce high-quality applications within deadlines. So, need to develop automated techniques to improve their maintainability have been aroused. It is believed that well-clustered mobile applications are easy to maintain. In this paper, BH algorithm along with its hybrids is applied for modularization of five android applications (described in Table I). The prime contributions of this research work are listed below:

- Formulation and investigation of the use of BH algorithm as multi-objective optimization technique for the process of software modularization of android mobile applications.
- Investigation of the impact of hybridizing BH algorithm with GA and auxiliary archive.

II. LITERATURE REVIEW

Various search based optimization techniques have been applied to software clustering in past. A remarkable work in this field includes the use of GA and Hill Climbing algorithm (Bunch tool) [6] for automatically clustering software. They used the representation of the given software as a Module Dependency Graph (MDG) and Modularization Quality (MQ) is optimized to get desired clustering efficiently. MQ is further defined as the ratio of cohesion and coupling [6]. Praditwong et al. [7] [9] used six objectives for automatic software clustering and this approach outperforms Bunch tool. The authors of [10] used multi-objective GA for software modularization. In another work [11], the sum of intra-edges, inter-edges and the number of changes between original and updated clustering are used as fitness objectives using NSGA-II. This technique has been found to be successful for re-clustering. In another work [12], the authors used cooperative clustering for software modularization on the basis of MQ. With increase in size of problem, performance of this approach degrades. Particle Swarm Optimization (PSO) [4] and BH [13] algorithm has also been used for software clustering using MQ as optimization objective. Mkaouer et al. [14] applied NSGA-III algorithm for modularization of software using seven objectives. The approach is applicable if evolutions of the software are carefully maintained.

III. SOFTWARE MODULARIZATION USING BLACK HOLE ALGORITHM

BH algorithm is an optimization algorithm that searches for optimal solution on the basis of a set of objectives (mentioned
in Table II) often conflicting with each other. The general structure of BH algorithm is shown in Figure 1. To implement BH, the population of individuals is initialized using (1).

\[ x_i^j = 1 + \text{rand}(0,1)(n - 1) \]

where \( i = 1, 2, \ldots, \text{Pop} \) (\( \text{Pop} \) is the population size as described in Table III), \( j = 1, 2, \ldots, n \) (\( n \) is the number of modules to be clustered). The control parameters to be used for implementing Black Hole algorithms are shown in Tables III. Since the BH algorithm is random, so each experiment has been conducted repeatedly 30 times, and the results thus obtained are analyzed and compared to that of existing Two-Archive algorithm based approach [7, 9]. NP, NAE, NIE, NCP, NCD and NIP described in Table II have been used as metrics for comparison. The problem of software clustering is formulated as a minimization problem. NAE (Table II) is a maximization objective and is reformulated as minimization objectives by negating its value.

\[ \text{Fitness}_{\text{new}^{i,j}} = \frac{\text{Fitness}_{\text{old}^{i,j}} - \text{Fitness}_{\text{new}^{i,j}}}{\text{Fitness}_{\text{old}^{i,j}}} \]

where \( \text{Fitness}_{\text{new}^{i,j}} \) denotes new location of each candidate solution called star, \( \text{Fitness}_{\text{old}^{i,j}} \) denotes the most fit star as Black Hole \( x_{\text{BH}}^{i,j} \), and \( n \) is the number of modules to be clustered.

The objective is to discover the best fitness. The algorithm is repeating the process until one of the stopping criteria is met. The stopping criteria are met if the size of the population \( N \) is less than or equal to a predefined value, or if the number of generations \( G \) is greater than or equal to a predefined value.

**TABLE II. OBJECTIVES TO BE OPTIMIZED**

<table>
<thead>
<tr>
<th>Objective</th>
<th>Optimization</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Clusters (NP)</td>
<td>Minimize</td>
<td>Lesser the number of clusters more is the number of modules per cluster [7]</td>
</tr>
<tr>
<td>Number of Intra-edges (NAE)</td>
<td>Maximize</td>
<td>Dependencies among modules in the same cluster [7]</td>
</tr>
<tr>
<td>Number of Interal-edges (NIE)</td>
<td>Minimize</td>
<td>Dependencies among modules in different clusters [7]</td>
</tr>
<tr>
<td>Number of Modules per Cluster (NCP)</td>
<td>Minimize</td>
<td>It is conflicting to objective NP. It tends to create equal sized clusters [7]</td>
</tr>
<tr>
<td>Number of Cyclic Dependencies (NCD)</td>
<td>Minimize</td>
<td>Dependencies such that modules in cluster A depends on modules in cluster B and some other modules of cluster B depends on modules in cluster A [11]</td>
</tr>
<tr>
<td>Number of Isolated Clusters (NIP)</td>
<td>Minimize</td>
<td>Clusters with a single module [9]</td>
</tr>
</tbody>
</table>

**TABLE III. CONTROL PARAMETERS FOR SOFTWARE CLUSTERING**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size ( (N) )</td>
<td>200</td>
<td>Manually tested by repeated executions of the algorithms.</td>
</tr>
<tr>
<td>Generations</td>
<td>( 10 \times n ) or When output does not change for 300 consecutive generations</td>
<td>Stopping criteria</td>
</tr>
<tr>
<td>Number of variables to be optimized ( (n) )</td>
<td>Number of modules to be decomposed.</td>
<td>Each individual is composed of ( n ) decision variables.</td>
</tr>
<tr>
<td>Size of REP</td>
<td>1% of the size of population</td>
<td>To keep track of best (non-dominating) solutions</td>
</tr>
<tr>
<td>Crossover function (for GA)</td>
<td>Arithmetic</td>
<td>Child=( R_1 \times \text{Parent 1} + R_2 \times \text{Parent 2} ) where ( R_1 ), ( R_2 ) are independent random numbers between 0 and 1. Ideal value for software Clustering (found by manual testing): 0.6.</td>
</tr>
<tr>
<td>Mutation function (for GA)</td>
<td>Uniform</td>
<td>Ideal value for software Clustering (found by manual testing): 0.02.</td>
</tr>
</tbody>
</table>

**B. Multi-Objective Hybrid Black Hole Algorithm (MOBH)**

In this work, an auxiliary archive has been used to store Pareto front. Hyper-cubes have been used to maintain the best solutions for each iteration of the algorithm [16]. Although, the algorithm is very efficient in identifying optimal solutions but as the size of the problem increases, these algorithms tend to get stuck at local optima and the outputs are hence not globally optimal. In order to recover these algorithms from local optima, GA is used [5]. It leads to develop hybrid for MOBH and MOBH called MOWBHGA and MOBHGA respectively. The algorithms thus developed are shown in Figure 2 and 3 respectively.

**IV. EXPERIMENTAL RESULTS AND ANALYSIS**

In order to validate the clustering process, MoJoFM and EdgeSim have been used as assessment criteria. MOWBHGA and MOBHGA are used for clustering of sample mobile applications (Table I) and results are compared to existing Two-Archive approach [7].
I) **MoJoFM as Assessment Criteria**

Let A be the automatic clustering and B be the reference cluster structure of an object-oriented system developed [17].

\[
\text{MoJoFM} = \left( 1 - \frac{\min(A,B)}{\max(\min(A\ominus B), 0)} \right) * 100
\]

\( mno(A, B) = \min(\text{move and join operations to transform } A \text{ to } B) \)

\( \max(\text{mno}(A\ominus B)) = \text{most distant decomposition from reference decomposition} \)

II) **EdgeSim as Assessment Criteria**

\[
\text{EdgeSim} = \frac{\text{Weight}(\gamma)}{\text{Weight}(E)} \times 100
\]

Where \( E \) is the set of all edges in a given MDG and \( \gamma \) is the set of inter-edges (inter-edges in A are inter-edges in B) or intra-edges (intra-edges in A are intra-edges in B).

Higher the value of MoJoFM and EdgeSim, better is the clustering.

---

**Figure 2.** Multi-Objective Weighted Black Hole Genetic Algorithm (MOWBHGA)

**Step 1 [Initialize population]:**
- **Step 1.1:** Encode and initialize the population of candidate clustering solutions. Each individual is called a star. Set parameters as shown in Table III.
- **Step 1.2:** Evaluate the fitness of each candidate in the population on the basis of combined weighted objective \( F \) calculated by using (2).
- **Step 1.3:** Designate the solution with the least value of \( F \) as Black Hole \((X_{BH})\).

Repeat steps 2-5 until stopping criteria is met (as indicated in Table III).

**Step 2 [Identify new possible solutions]:** For each iteration \( t \), identify new location of star \((x(t+1))\) for each \( h \)-th clustering \((x(t))\)

\[
x(t+1) = x(t) + \text{rand}(X_{BH} - x(t))
\]

**Step 3 [Search for a better solution]:** Evaluate fitness of each new clustering \((x(t+1))\). If new candidate is better than the current candidate, then replace the current solution with this new one. This is required to locally search for a better sequence. It moves the current candidate randomly in search for a better solution.

**Step 4 [Update the best solution]:**
- **Step 4.1:** If the new solution is better than the current Black Hole \((X_{BH})\), then designate this new solution as new Black Hole \((X_{BH})\).

**Step 4.2:** Calculate the radius of the event of horizon \((R)\) of the Black Hole clustering.

\[
R = \frac{F_{BH}}{\text{Mean}_{Pop}} f_i
\]

Where \( F_{BH} \) is the fitness for Black Hole clustering and \( f_i \) is the fitness of \( i \)-th clustering calculated using weighted fitness function calculated using Eq. (2).

- **Step 4.3:** If a star enters this event horizon, it is absorbed by the Black Hole. It means, if \((F_{BH} - f_i < R)\), the clustering is discarded as it is regarded to be entered in event horizon of Black Hole and is thus vanished. Generate new clustering using Eq. (1) to balance the size of the population.

**Step 5 [Genetic Algorithm]:**
- **Step 5.1 [Input]:** Take the population of candidate clustering from the Step 4 as input population of chromosomes and set parameters as shown in Table III.

Calculate the fitness of each solution using function \( F \) described in Eq. (2).

- **Step 5.2 [Selection]:** Select two parent chromosomes (tournament selection of size 2) from the population on the basis of their fitness.

- **Step 5.3 [Crossover]:** Cross the parents selected in Step 5.2 to create new children and mutate new child at random positions in the chromosome.

- **Step 5.4 [Replace]:** If this new offspring is better than the parents in terms of \( F \) calculated using Eq. (2), then accept it.

- **Step 6 [Output]:** Output the candidate clustering having least value of combined objective function \( F \).

---

**Figure 3.** Multi-Objective Hybrid Black Hole Algorithm (MOBHGA)
V. CONCLUSION

This work proposes the application of multi-objective BH algorithm and its hybrids with GA and auxiliary archive for clustering of mobile applications and the resulting modularizations are compared to Two-Archive algorithm. The results indicate that MOBHGA algorithm outperforms weighted objective based hybrid MOWBHGA and Two-Archive algorithm for clustering mobile applications. In future, PSO could be investigated for hybridizing Black Hole algorithm for obtaining even better clustering results.

REFERENCES