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.acin

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 clsutering

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.
- Comparison of modularization results of proposed hybrid approaches to that of existing Two-Archive approach [7].

TABLE I. DESCRIPTION OF THE TEST PROBLEMS

Software System	Modules in MDG	Edges in MDG	System Description
Foursquare	54	6	Open source popular game
Sudoku	74	5	Open source popular game
Apps Organizer	135	13	Open source Organizer
Punjab Kesari	1234	32	Proprietary famous Punjabi newspaper (Developed by 'Converse New Media)
Desi coupon	244	4	Proprietary advertisement management app (Developed by 'Iniz Solutions'& yet to be launched)

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 Modularziation Using Black Hole Algorithm

BH algorithm is an optimization algorithm that searches for optimal solution on the basis of a set of objectives (mentioned

DOI reference number: 10.18293/SEKE2016-004

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).

$$x_i^j = 1 + rand(0,1)(\mathrm{n} - 1) \quad (1)$$

where i=1,2,…,Pop (Pop is the population size as described in Table III); j=1,2,…,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.

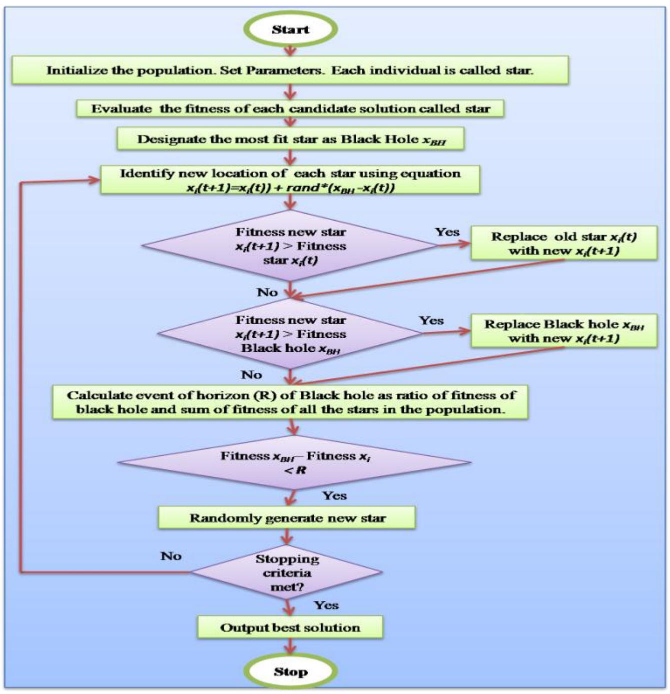

Figure 1. Black Hole Algorithm

A. *Multi-Objective Weighted Black Hole Algorithm (MOWBH)*

MOWBH algorithm is applied to the problem of software clustering. In order to calculate the fitness of an individual, weighted sum approach has been used. In this approach, all g objectives (f_k) mentioned in Table II are combined to make a single objective (F) as shown in (2). Use of random weights leads to sufficient diversity to obtain good quality clusters. The sum of weights of all the g objectives should be 1. This approach is easy to implement and widely used for multi-objective optimization. To overcome negative impact of randomness, the algorithm is executed 30 times with different random weights and the solution with least value of F is selected as the output [15].

$$F = \sum_{k=1}^{g} w_k f_k \ and\ \sum_{k=1}^{g} w_k = 1 \quad (2)$$

These algorithms are highly dependent on the weights and in case of conflicting objectives; allocation of weights is sometimes difficult. To overcome these problems, we investigated the use of Pareto optimization approaches [13] for optimizing the modularization of mobile applications.

TABLE II. OBJECTIVES TO BE OPTIMIZED

Objective	**Optimization**	**Description**
Number of Clusters (NP)	Minimize	Lesser the number of clusters more is the number of modules per cluster [7]
Number of IntrA-edges (NAE)	Maximize	Dependencies among modules in the same cluster [7]
Number of IntEr-edges (NIE)	Minimize	Dependencies among modules in different clusters [7]
Number of Modules per Cluster (NCP)	Minimize	It is conflicting to objective NP. It tends to create equal sized clusters [7]
Number of Cyclic Dependencies (NCD)	Minimize	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]
Number of Isolated Clusters (NIP)	Minimize	Clusters with a single module [9]

TABLE III. CONTROL PARAMETERS FOR SOFTWARE CLUSTERING

Parameter	**Value**	**Comments**
Population size (N_s)	200	Manually tested by repeated executions of the algorithms.
Generations	10 * n or When output does not change for 300 consecutive generations.	Stopping criteria
Number of variables to be optimized (n)	Number of modules to be decomposed.	Each individual is composed of n decision variables.
Size of REP	1% of the size of population	To keep track of best (non-dominating) solutions
Crossover function (for GA)	Arithmetic	*Child=R1 * Parent1+ R2 * Parent2* Where *R1*, *R2* are independent random numbers between 0 and 1. Ideal value for software Clustering (found by manual testing): 0.6.
Mutation function (for GA)	Uniform	Ideal value for software Clustering (found by manual testing): 0.02.

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 MOWBH 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].

1) MoJoFM as Assessment Criteria

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

$$MoJoFM = \left(1 - \frac{mno(A,B)}{\max\,(mno(\forall A,B))}\right) * 100$$

mno(A,B)=min(move and join operations to transform A to B), max(mno(∀A,B))=most distant decomposition from reference decomposition.

2) EdgeSim as Assessment Criteria

$$EdgeSim = \frac{Weight(\gamma)}{Weight(E)} * 100$$

Where E is the set of all edges in a given MDG and γ 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.

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 Tables 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 ($x_i(t+1)$) for each i^{th} clustering ($x_i(t)$)

$$x_i(t+1)=x_i(t)) + rand*(x_{BH} - x_i(t))$$

Step 3 ***[Search for a better solution]:*** Evaluate fitness of each new clustering $x_i(t+1)$. If new candidate is better than the current candidate, then replace the current solution with this new. 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.

$$R = \frac{F_{BH}}{\sum_{i=1}^{Pop} F_i}$$

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

Step 1 ***[Initialize population]*****:**

Step 1.1: Encode and initialize the population of possible clustering solutions. Set parameters as shown in Table III.

Step 1.2: Evaluate fitness of each candidate in the population using objective functions mentioned in Table II.

Step 1.3: Store the clustering that represent non-dominated vectors in the temporary repository (REP) and generate hyper-cubes to maintain best solutions.

Step 1.4: Select current best non-dominated clustering achieved so far and designates it as Black Hole (X_{BH}).

Repeat steps 2 to 6 until stopping criteria is met (as shown in Table III)

Step 2 ***[Identify new possible solutions]*****:** For each iteration t, identify new location ($x_i(t+1)$) for each job sequence ($x_i(t)$) by using

$$x_i(t+1) = x_i(t) + rand * (x_{BH} - x_i(t)$$

Step 3 ***[Search for a better solution]*****:** Evaluate fitness of each new clustering $x_i(t+1)$. If new candidate solution is better than the current candidate solution taking into consideration multiple objectives and their non-dominance, then replace the current solution with this new solution else ignore it. This step 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 clustering $x_i(t+1)$ is better than the current Black Hole (x_{BH}), then designate this new clustering as new Black Hole (x_{BH}).

Step 4.2: Calculate the radius of event of horizon (R) of the Black hole clustering in non-dominated Pareto front by calculating components of radius on the basis of objectives mentioned in Table II. For each objective (h), the component of the radius is

$$R(h) = \frac{f_{BH}}{\sum_{i=1}^{pop} f_i(h)}$$

Where f_{BH} is the fitness value of the Black Hole clustering and $f_i(h)$ is the fitness value of the h^{th} objective of i^{th} clustering. For the problem of software clustering under consideration, the number of objectives (*h*) is equal to 6, *Pop* is the size of the population under consideration (Table III).

Step 4.3: For each individual in the population and BH, if difference in fitness value of every corresponding objective function (*h*) dominates corresponding component of R i.e. R ($f_i(h)$- f_{BH} dominates $R(h)$), the candidate clustering is discarded and a new star is generated randomly.

Step 5 ***[Apply Genetic algorithm]:***

Step 5.1 ***[Input]*****:** Take the population of candidate clustering from Step 4 as input population of chromosomes. Set parameters as shown in Table III. Calculate the fitness of each solution using the non-dominance approach on the basis of objectives mentioned in Table II.

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 (non-dominating), then accept it.

Step 6 ***[Update best solutions]:*** Update hyper-cubes and REP to maintain current non-dominated clustering.

Step 7 ***[Output]:*** Return REP which includes resulting non-dominated clustering.

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

TABLE IV. MoJoFM AND EDGESIM TO COMPARE MOWBHGA AND MOBHGA FOR SAMPLE ANDROID MOBILE APPLICATIONS

Mobile App	Approach	MoJoFM							EdgeSim						
		NP	NAE	NIE	NCP	NCD	NIP	Value	NP	NAE	NIE	NCP	NCD	NIP	Value
FourSquare	MOWBHGA	6	64	233	8	13	0	32.65306	6	57	240	10	13	0	61.27946
	MOBHGA	4	167	130	28	5	0	**57.14286**	4	167	130	28	5	0	**65.29966**
	Two-Archive	7	170	127	25	8	2	44.89796	7	262	35	23	0	1	64.53674
Sudoku	MOWBHGA	5	214	59	65	2	2	34.78261	5	214	59	65	2	2	59.34066
	MOBHGA	4	151	122	45	4	1	**49.27536**	2	220	53	56	1	0	**67.39927**
	Two-Archive	7	190	83	11	0	0	46.37681	7	165	108	12	0	0	62.27106
Apps Organizer	MOWBHGA	13	45	529	12	9	0	35.65892	13	45	529	12	9	0	57.49129
	MOBHGA	7	369	205	99	7	3	41.86047	13	100	474	16	0	1	**63.67247**
	Two-Archive	15	267	307	18	0	0	**46.51163**	15	400	174	17	0	0	62.71777
Punjab Kesari	MOWBHGA	45	160	5627	33	114	0	46.38429	45	138	5649	25	115	0	47.21347
	MOBHGA	32	1048	4739	400	102	6	**49.09164**	34	1513	4274	479	76	6	47.46846
	Two-Archive	15	400	174	17	0	0	48.72768	44	3233	2528	41	105	9	**49.87098**
Desi Coupon	MOWBHGA	4	845	32	238	0	2	59.3361	4	858	`19	240	0	3	**88.25542**
	MOBHGA	3	406	471	137	12	0	**60.16598**	4	660	217	212	7	0	87.68529
	Two-Archive	5	380	497	67	0	1	59.30361	6	778	99	47	0	1	86.20297

Analyzing Table IV reveals that if MoJoFM is used as validation criteria, MOBHGA results in a highest value as compared to its counterparts in 4 out of 5 sample applications and if EdgeSim is used as validation criteria, MOBHGA results in a highest value in 3 out of 5 sample applications.

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

[1] A. Hatamlou, "Black hole: A new heuristic optimization approach for data clustering," *Information Sciences*, vol. 222, pp. 175-184, 2013.

[2] P. R. Srivastava, M. Chis, S. Deb, and X.-S. Yang, "An efficient optimization algorithm for structural software testing," *International Journal of Artificial Intelligence™*, vol. 8, pp. 68-77, 2012.

[3] K. Srinivasan and D. Fisher, "Machine Learning Approaches to Estimating Software Development Effort," *Series on Software Engineering and Knowledge Engineering*, vol. 16, p. 52, 2005.

[4] I. Hussain, A. Khanum, A. Q. Abbasi, and M. Y. Javed, "A Novel Approach for Software Architecture Recovery using Particle Swarm Optimization," *International Arab Journal of Information Technology (IAJIT)*, vol. 12, 2015.

[5] A. S. Mamaghani and M. R. Meybodi, "Clustering of software systems using new hybrid algorithms," in *Computer and Information Technology, 2009. CIT'09. Ninth IEEE International Conference on*, 2009, pp. 20-25.

[6] B. S. Mitchell and S. Mancoridis, "On the evaluation of the Bunch search-based software modularization algorithm," *Soft Computing*, vol. 12, pp. 77-93, 2008.

[7] K. Praditwong, M. Harman, and X. Yao, "Software module clustering as a multi-objective search problem," *Software Engineering, IEEE Transactions on*, vol. 37, pp. 264-282, 2011.

[8] K. J. Sullivan, W. G. Griswold, Y. Cai, and B. Hallen, "The structure and value of modularity in software design," in *ACM SIGSOFT Software Engineering Notes*, 2001, pp. 99-108.

[9] K. Praditwong, "Solving software module clustering problem by evolutionary algorithms," in *Computer Science and Software Engineering (JCSSE), 2011 Eighth International Joint Conference on*, 2011, pp. 154-159.

[10] C. Kishore and A. Srinivasulu, "Multi-objective approach for software module clustering," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 2, 2012.

[11] H. Abdeen, H. Sahraoui, O. Shata, N. Anquetil, and S. Ducasse, "Towards automatically improving package structure while respecting original design decisions," in *Reverse Engineering (WCRE), 2013 20th Working Conference on*, 2013, pp. 212-221.

[12] A. Ibrahim, D. Rayside, and R. Kashef, "Cooperative based software clustering on dependency graphs," in *Electrical and Computer Engineering (CCECE), 2014 IEEE 27th Canadian Conference on*, 2014, pp. 1-6.

[13] K. Jeet and R. Dhir, "Software Architecture Recovery using Genetic Black Hole Algorithm," *ACM SIGSOFT Software Engineering Notes*, vol. 40, pp. 1-5, 2015.

[14] W. Mkaouer, *et al.*, "Many-Objective Software Remodularization Using NSGA-III," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 24, p. 17, 2015.

[15] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and multidisciplinary optimization*, vol. 41, pp. 853-862, 2010.

[16] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 8, pp. 256-279, 2004.

[17] F. Beck and S. Diehl, "On the impact of software evolution on software clustering," *Empirical Software Engineering*, vol. 18, pp. 970-1004, 2013.