Indian J.Sci.Res. 7 (2): 214-217, 2017 ISSN: 0976-2876 (Print)
ISSN: 2250-0138(Online)

GREY WOLF ALGORITHM FOR SOFTWARE ORGANIZATION

KAWAL JEET[1]

Department of Computer Science, D.A.V. College, Jalandhar, Punjab, India

ABSTRACT

Nature has always been a source of inspiration for solving complex engineering problems. It could be easily observed in the latest optimization algorithms that have been inspired by nature. Grey Wolf Optimization algorithm is a recent optimization algorithm that has been inspired by hunting behavior of grey wolves. In this paper, this algorithm has been investigated for the process of software organization. Software organization refers to the placements of software entities into appropriate clusters. Hybrid of Grey Wolf algorithm with Genetic Algorithm has also been investigated for software organization and has observed improved efficiency. The algorithms have been applied for clustering of five C, C++ and java based software applications and compared to some of the existing search based software organization techniques.

KEYWORDS: Nature-Inspired Algorithm; MoJoFM, EdgeSim, Software Modularization.

Nature has always been a source of inspiration for many researchers. Nowadays, solution to most of the problems is nature-inspired. Nature-inspired algorithm is a relatively new area with a brief history at this early stage of development. Even then as compared to the traditional and well-established technique they have still proved their great potential, flexibility and efficiency as well as ever-increasing and wide variety of applications.

Grey Wolf Optimization algorithm (GWO) is a recent nature-inspired algorithm inspired from grey wolves (Mirjalili, et al., 2016). They follow a very strict social dominant hierarchy.

- At the top, there are leaders are called alphas (x_α) those are responsible for taking decisions on hunting, sleeping place, time to wake, and so on.
- At second level in the hierarchy of grey wolves are betas (x_β) those acts as alpha in case one of the alpha wolves dies or becomes very old.
- At third level in the hierarchy of grey wolves are deltas (x_γ) those have to submit to alphas and betas, but dominates the omega.
- The lowest ranking grey wolves are *omegas* (also called scapegoat) those have to submit to all the other dominant wolves.

The main phases of gray wolf hunting are as follows:

- The social hierarchy is utilized to find the best solutions obtained so far.
- The encircling mechanism is utilized to define a circle-shaped neighborhood around each candidate solutions which can be further extended to hyper-sphere of large dimensions.
- The random parameters A and C are exploited by the candidate solutions to develop hyper-spheres of different random radii.
- The hunting method helps candidate solutions to locate the prey.
- Exploration and exploitation are performed by a and A which are adapted during each iteration.

GWO algorithm has observed a wide variety of applications. In this paper, application of GWO has been investigated for the cause of software organization of five C, C++ and java based software

A software system is composed of entities that could be a class, function or variables which are related to each other due to procedure calls, inheritance relationships, variable references, etc. The syntactic structures of these systems could be represented as a graph called a Module Dependency Graph (MDG), where the nodes are the entities and edges are the relations between these entities. Large numbers of source code analysis tools such as Jdeps, Code Dependency Analyser etc. are available that could be used to retrieve these MDGs. The problem of finding the best clustering for a given set of clusters in an MDG is an NP-hard search problem. So, automated assistance to partition MDG's is required. Large number of search-based techniques has been used in past two decades for the cause of software

[1]**Corresponding author**

organization. Some of these are described in Table I.

Table I: Literature review

Approach	Optimization Objectives	Algorithm Used
(Harman et al.,2002)	MQ is ratio of cohesion and coupling	Modified Genetic Algorithm (GA)
(Praditwong, 2011)	MQ	Grouping Genetic Algorithm (GGA)
(Ibrahim et al., 2014)	MQ	Cooperative clustering
(Hussain et al.,2015)	MQ	PSO
(Praditwong et al., 2011)	Cohesion, Coupling, MQ, Number of clusters, Number of isolated clusters, Equal Size clusters	Two-Archive algorithm
(Mkaouer, et al., 2015)	Number of packages, cohesion, coupling, classes per package, the number of code changes, vocabulary based similarity and the similarity between history of code changes	NSGA-III

The prime contributions of this research work are listed below:

- To use Grey Wolf Optimization algorithm as multi-objective optimization technique for the process of software organization.
- To compare proposed multi-objective GWO and its hybrid to that of existing Two-Archive, MOGA and NSGA-II based approaches for the cause of software organization.

PROPOSED METHODOLOGY

The steps followed in the present research work for software organization are shown in Fig. I. The fitness functions on the basis of which these entities are clustered in the present research are presented in Table II. These objectives are observed to be conflicting with each other, and hence obtaining an optimal result is very difficult.

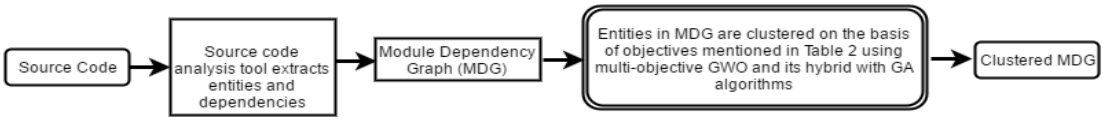

Figure I: Proposed Methodology

Table II: Objectives to be optimized

Objective	Type of optimization
Number of Clusters (NC)	Minimize
Cohesion	Maximize
Coupling	Minimize
Number of Classes per Package	Minimize
Number of Cyclic Dependencies	Minimize
Number of Isolated clusters	Minimize

Multi-Objective Grey Wolf Optimization Algorithm (MOGWO) algorithm (Mirjalili, et al., 2016) could be used to cluster MDG in appropriate clusters on the basis of optimization function described in Table II. In this way, clustering of a software system can solely be performed using static structure of the software retrieved from its source code. This is possible even when original design documents is not available. MOGWO algorithm is observed to be an efficient algorithm but similar to other it gets stuck at local optima. Large number of techniques are used in literature to direct such search based optimization algorithms

towards global optima. In this paper, GA has been used for this reason. This is because the crossover and mutation operators of GA are very effectual in full exploration of search space. The algorithm thus obtained is called MOGWGA. The parameters to be used for controlling the execution of these algorithms are shown in Table III.

Table III: **Control Parameters for Software organization**

Parameter	Description
Variables to be optimized (n)	Number of entities to be clustered
Population size (Pop)	5*n
Selection Algorithm	Tournament selection
Mutation Operator for GA	Uniform mutation with probability=0.02
Size of REP	1% of Pop

RESULTS AND DISCUSSION

Multi-Objective Grey Wolf algorithm (MOGWO) and its hybrid with GA (MOGWGA) are tested on samples described in Table IV. The results thus obtained are compared to that of Two-Archive, MOGA(Deepika & Brindha, 2012) and NSGA-II (Abdeen et al., 2009) algorithms. In order to validate the resulting clustering MoJoFM (Corazza et al., 2016) has been used as assessment criteria.

The values of MoJoFM for the clustering of sample applications using MOGWO and MOGWOGA are shown in Table V. These values provide evidence about the extent to which basic multi-objective MOGWO and MOGWOGA is successful in software organization.

As far as MOGWO is concerned, it is seen that the value of MoJoFM is lesser than that obtained by the application of Two-Archive algorithm based approach.

For MOGA based approach, the value of MoJoFM is higher in 2 out of 5 sample problems.

For NSGA-II based approach, the value of MoJoFM is higher in all the sample problems.

So MOGWO performs better than NSGA-II and MOGA as far as MoJoFM is concerned but perform less efficient than Two-Archive based approach.

As far as MOGWOGA is concerned, it is seen that for Two-Archive algorithm based approach, the value of MoJoFM is higher in 3 out of 5 sample problems.

For MOGA and NSGA-II based approach, the value of MoJoFM is higher in all sample problems.

So MOGWOGA performs better than all other counterparts including MOGWO, Two-archive, NSGA-II and MOGA based approaches as far as MoJoFM is concerned.

Table IV. Test Problems

Software	No of entities	No of clusters in reference clustering	Description
M-Tunis	20	5	A simple operating system
Ispell	24	10	An open source spell checker
Rcs	29	4	Open source version control tool
Bison	37	4	Parser generator
Grappa	74	10	Graph visualization and drawing tool

Table V: MoJoFM to compare clustering of sample software

Software	MOGWO	MOGWGA	Two-Archive	MOGA	NSGA-II
M-Tunis	53.125	63.862	65.625	50.75	30.612
Ispell	29.089	41.667	36.111	30.481	28.212
Rcs	58.139	68.791	69.767	43.854	37.781
Bison	41.071	66.857	64.762	50.000	36.812
Grappa	29.018	33.966	31.428	30.302	22.561

If multi-objective Grey Wolf algorithm (MOGWO) (Mirjalili, et al., 2016) is compared to its hybrid (MOGWGA) obtained by using GA before updating best solutions during each iteration, it is observed that MOGWGA performs better than MOGWO for all sample software. This is because; GA tends to extract search-based algorithms out of local optima and directs them towards better search space.

CONCLUSION AND FUTURE WORK

This work proposes the application of multi-objective Grey Wolf algorithm and its hybrid with GA for the clustering of five sample software. Six clustering objectives have been considered for application of these algorithms. The results are compared to that of Two-Archive, MOGA and NSGA-II based approaches. Empirical results indicate that MOGWGA algorithm obtained by use of GA outperforms its counterpart as well as Two-Archive, MOGA and NSGA-II based approaches for the cause of software organization.

In future, other approaches such as particle swarm optimization could be used for hybridizing GWO algorithm for obtaining even better clustering results. Clustering results will be compared to other existing search based approach. A freely accessible automated tool will also be developed for the same.

REFERENCES

Abdeen H., Ducasse S., Sahraoui H. and Alloui I., 2009. Automatic package coupling and cycle minimization. Paper presented at the International Working Conference on Reverse Engineering, WCRE'09.

Corazza A., Di Martino S., Maggio V. and Scanniello G., 2016. Weighing lexical information for software clustering in the context of architecture recovery. Empirical Software Engineering, **21**(1): 72-103.

Deepika T. and Brindha R., 2012. Multi objective functions for software module clustering with module properties. Paper presented at the International Conference on Communications and Signal Processing (ICCSP).

Harman M., Hierons R.M. and Proctor M., 2002. A New Representation And Crossover Operator For Search-based Optimization Of Software Modularization. Paper presented at the Genetic and Evolutionary Computation Conference. GECCO.

Hussain I., Khanum A., Abbasi A.Q. and Javed M. Y., 2015. A Novel Approach for Software Architecture Recovery using Particle Swarm Optimization. International Arab Journal of Information Technology (IAJIT), **12**(1).

Ibrahim A., Rayside D. and Kashef R., 2014. Cooperative based software clustering on dependency graphs. Paper presented at the Canadian Conference on Electrical and Computer Engineering (CCECE).

Mirjalili S., Saremi S., Mirjalili S.M. and Coelho L. d. S., 2016. Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. Expert Systems with Applications, **47**: 106-119.

Mkaouer W., Kessentini M., Shaout A., Koligheu P., Bechikh S., Deb K., et al.; 2015. Many-Objective Software Remodularization Using NSGA-III. ACM Transactions on Software Engineering and Methodology (TOSEM), **24**(3): 17.

Praditwong K., 2011. Solving software module clustering problem by evolutionary algorithms. Paper presented at the International Joint Conference on Computer Science and Software Engineering (JCSSE).

Praditwong K., Harman M. and Yao X., 2011. Software module clustering as a multi-objective search problem. IEEE Transactions on Software Engineering, **37**(2): 264-282.