ISSN XXXX XXXX © 2017 IJESC

Research Article Volume 7 Issue No.12

Honey-Bee Foraging Algorithm for Load Balancing in Cloud Computing Optimization

Dr.Sudha Senthilkumar[1], Dr.K.Brindha[2], Prof. Rathi.R[3], Prof. Angulakshmi[4], Dr. Jothi[5], YashVardhanThirani[6]
Associate Professor[1], Assistant Professor (Selection Grade) [2], Assistant Professor (Senior) [3, 4], B.Tech Student[6]
School of Information Technology and Engineering
VIT Universty, Vellore, Tamilnadu, India[1, 2, 3, 4, 6]
Ambo University, Ethiopia[5]

Abstract:
Optimization algorithms are searching techniques where the objective is to find an optimal solution to a problem, for satisfying one or more constraint functions, pertaining to a set of rules. Studies of animals and insects have culminated in a number of models of swarm intelligence. Within these swarms, the group behavior of animals or insects is very complicated. The group behavior of a swarm of animals or insects arises from the behaviors of the individuals. Scientists came up with computational optimization methods based on biological sciences such as Genetic Algorithms, Particle Swarm Optimization, and Ant Colonies. The aim of this particular research paper is to describe an optimizing algorithm called the Honey Bee Foraging Algorithm, derived from the natural behavior of honey-bees, to find the ideal solution. This performs an exploitative neighbourhood search along with random explorative search. In this particular research paper, after understanding the foraging behavior of honey-bees, the HBFA - Honey Bee Foraging Algorithm and its other versions are demonstrated and are implemented to optimize many functions, and the findings are contrasted with the findings obtained using other algorithms. The findings conclude that the Honey Bee Foraging Algorithm offers some profit over other optimization techniques based on the problem.

Keywords: Foraging; Honey-Bee, Behaviour, Algorithm, Swarm Intelligence, Swarm-Based Optimization, Adaptive Neighbourhood Search, Cloud Computing.

I. INTRODUCTION

Cloud Computing is a framework for distributing IT services to client systems where resources are retrieved from the internet via web-based applications and tools. [1] Cloud Computing architecture allows the accessing of data so long as an electronic device can connect to the internet. Cloud Computing technology is becoming more and more popular with each day because the information being accessed is stored and retrieved from "cloud", or internet, and thus does not require users to access the information from a particular place. In a Cloud Computing environment, when a virtual machine is overloaded with many tasks over its capacity, the tasks have to be removed and submitted to a relatively free virtual machine to distribute the load among machines. Research suggests that Load Balancing in Cloud Computing can be successfully achieved by modelling the foraging characteristic of honey-bees. [2] Swarm Intelligence is described as the group problem solving abilities of insects or animals. It is the direct consequence of self-arrangement in which the interactions of lower-level members create a global level structure that may be deciphered as intelligence. These lower level interactions are governed by a set of rules that individual members of the colony follow with out any knowledge of its larger impacts. Individual members in a colony only have local information about their surroundings. Through communication, local-level interactions affect the global arrangement of the colony. Self-arrangement is created by several elements. Here, positive feedback is the first rule of self-arrangement. It is a set of lucid rules that help in generating the complicated structure. Attraction of honey bees to a fertile flower is an example of this procedure. The second element of self-arrangement is Negative feedback, which limits the impact of positive feedback and helps in creating a counter influencing system. Additionally, Randomness is the third element in self-arrangement. It adds an uncertain element to the system and enables colonies to test solutions for their most daunting problems (fooding, nest locations). Lastly, there are multiple interactions among individuals. There should be a threshold number of individuals who are able to interact with one another to convert their local-level activities into an interconnected living organism. After combining these elements, a decentralized structure is created. [3] A hierarchical structure is used to distribute the critical duties; there is no control over individuals but over instincts. This leads to creation of an efficient structure that helps the colony to thrive in spite of obstacles. [8] There are many animal species that gain from similar processes that help them to survive and to procreate. Furthermore, some other organisms follow similar rules to profit from each other's abilities. Up to a point, even the human body may be construed as a self-organized system. Cells in the body profit from each other's strength and distribute the responsibilities of overcoming the challenges, which can often be fatal for the individual cell. [9] The aim of this paper is to describe an optimization algorithm called the Honey Bee Foraging Algorithm to find the optimal solution. The algorithm performs an exploitative neighbourhood search along with a random explorative search. The HBFA has been successfully applied to multi-objective optimization, neural network training, data clustering, optimizing the design of mechanical components design-optimization, image processing. [4], [10]

II. THE HONEY BEE FORAGINGALGORITHM PSEUDO-CODE

Generate the initial population as n, set best patch size as m, set elite patch size as e, set the count of forager bees attracted

to the of elite sites as nep, set the number of forager bees around the best non-elite patches as nap, set the size of the neighborhood as nigh, set the maximum number of iterations as IterMax, and set the limit of error as Error.
i=0
Generate an initial population.
Calculate Fitness Quotient of the initial population.
Sort the initial population based on the fitness result.
While i FitnessQuotienti-1
1. i=i+l;
2. Select the elite patches and non-elite, best patches for neighborhood search.
3. Attract the forager bees to the elite patches and non-elite best patches.
4. Calculate the Fitness Quotient of each patch.
5. Sort the results based on their fitness.
6. Assign the rest of the bees for global search to the non-best locations.
7. Calculate the Fitness Quotient of non-best patches.
8. Sort the results according to their fitness.
9. Run until termination criteria is met.
END

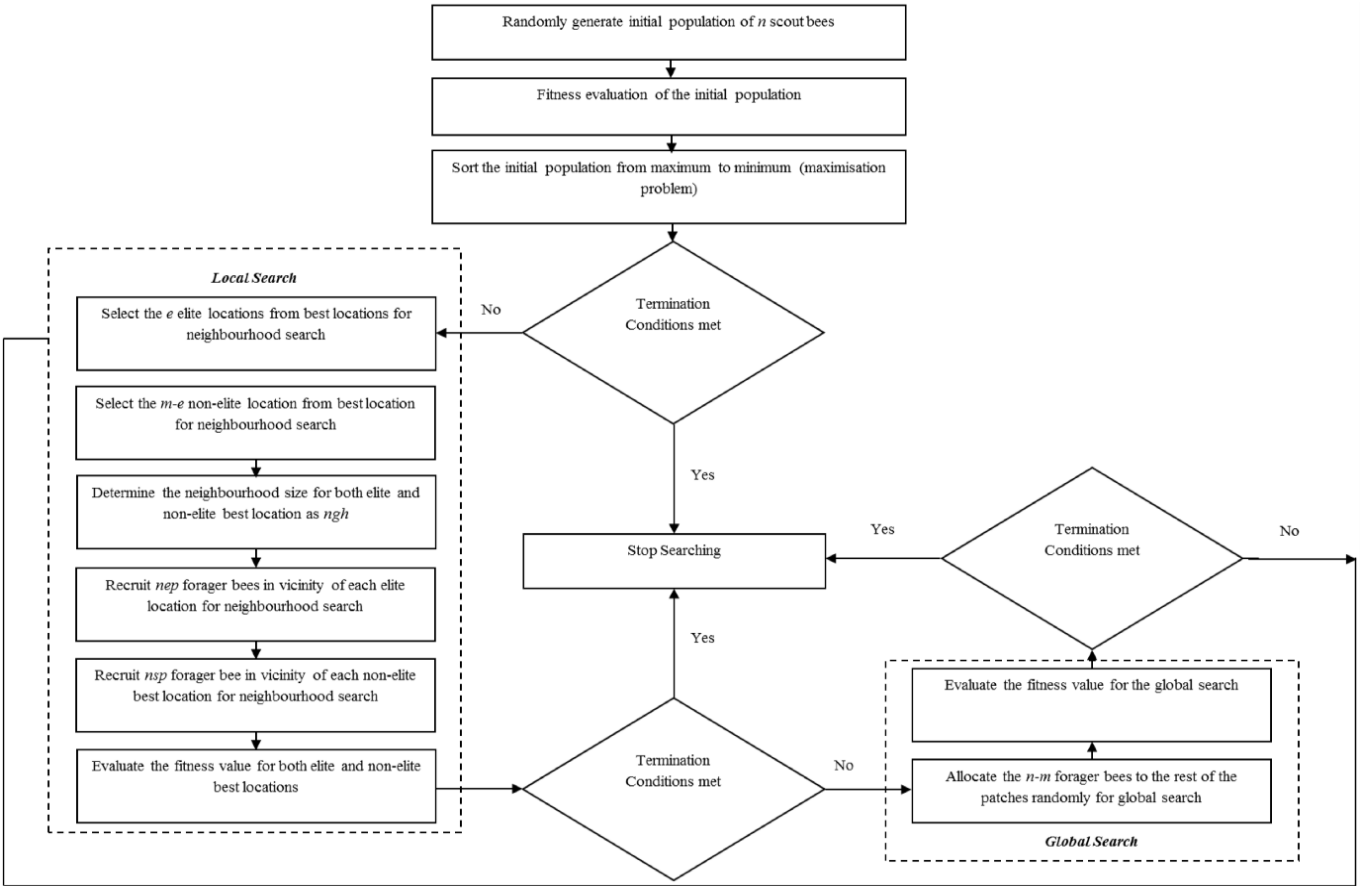


Figure.1. Flow chart Honey-Bee Algorithm

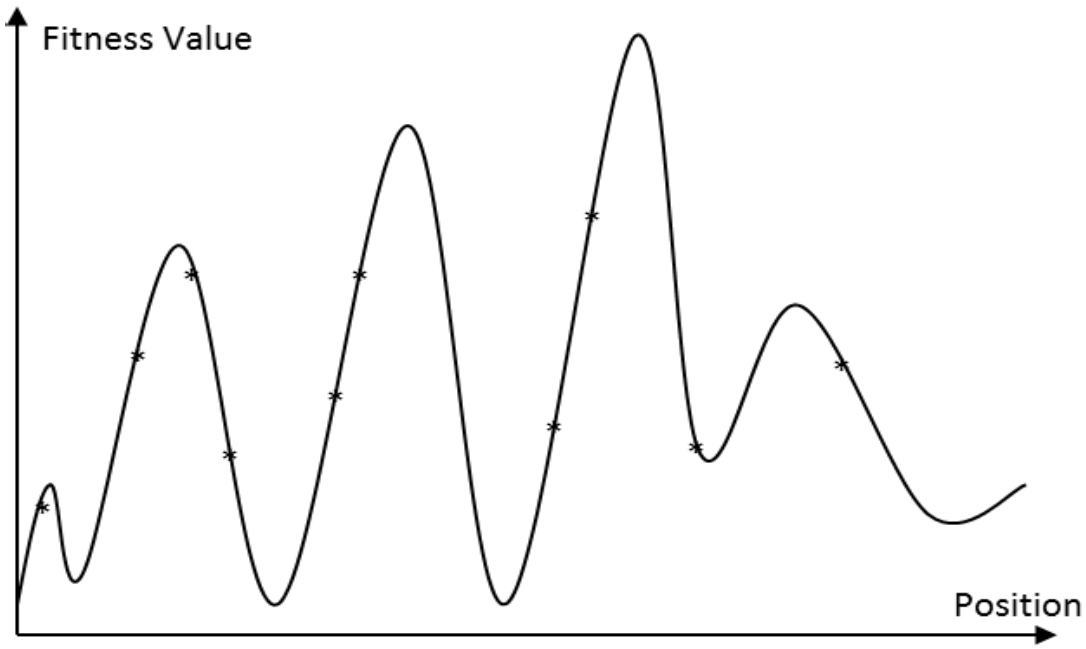


Figure.2. The initial selected n patches and their calculated Fitness Quotients

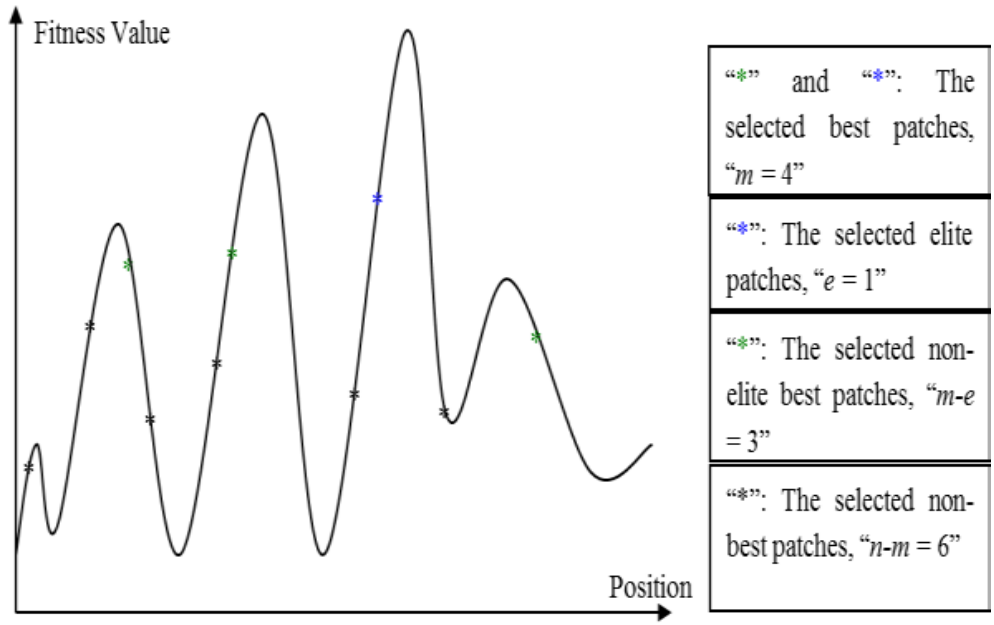


Figure.3. The selected elite and non-elite best patches

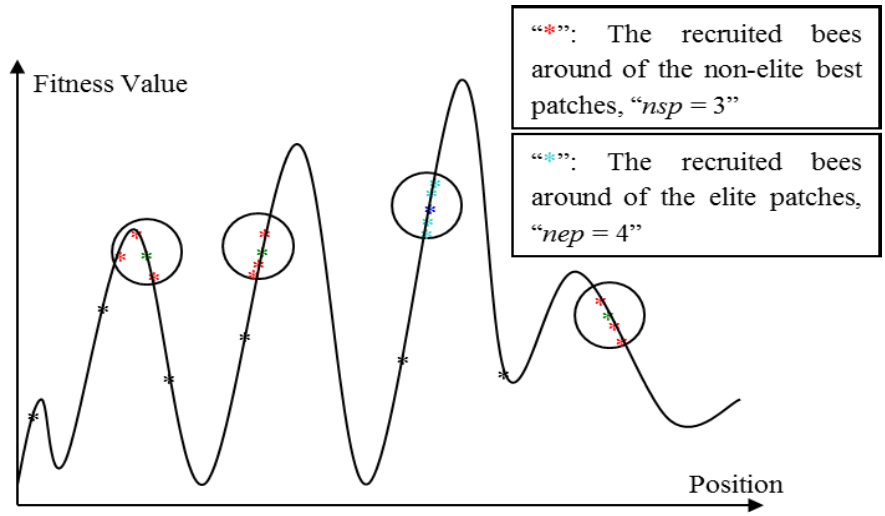


Figure.4. Attraction of forager bees towards elite and non-elite best locations

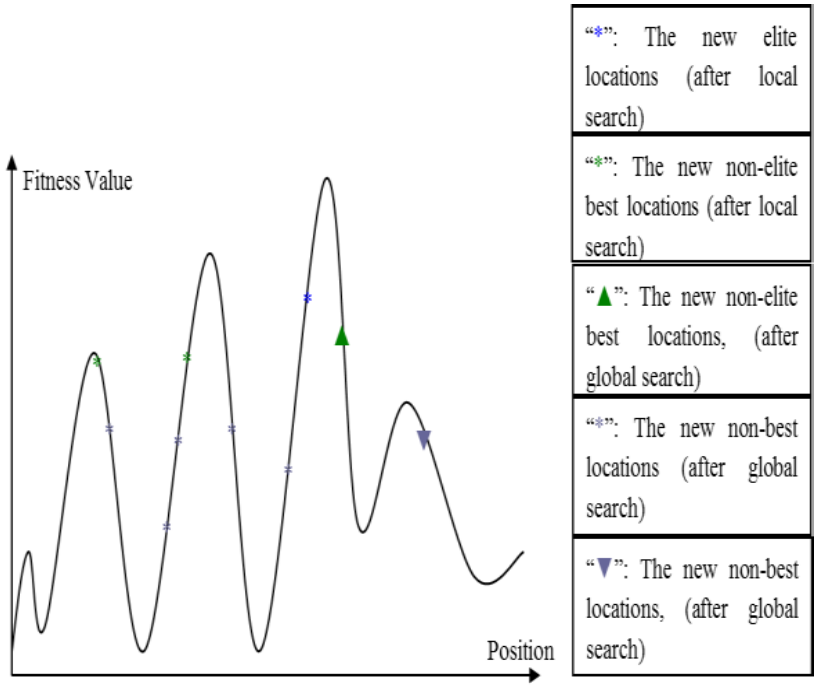


Figure.5. Results using HBFA after local and global search.

REQUIREMENT of Load Balancing

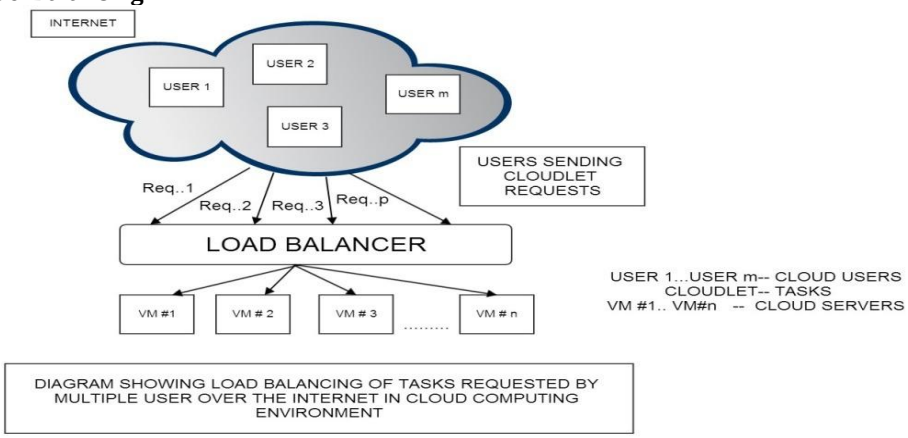
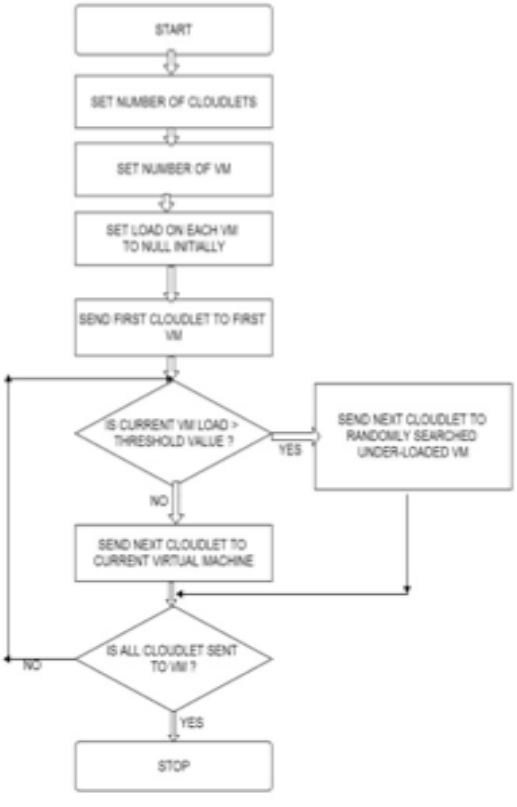


Figure.5. Load Balancer

III. IMPROVEMENT TO HBFA USING ADAPTIVE NEIGHBOURHOOD SEARCH AND SITE ABANDONMENT STRATEGY

This depicts the propose upgrades to the HBFA by using adaptive change to the neighbourhood size and site abandonment approach at the same time. Neighbourhood size change combined with site abandonment strategy has been attempted on HBFA and it was found that the convergence rate of HBFA can be slow when the promising locations are removed from the current best sites. [4] Here an adaptive neighbourhood size change and site abandonment (ANSSA) strategy is used which will avoid local least point by adjusting the neighbourhood size adaptively. It has shrinking and enhancement strategies according to the fitness evaluation. The initial ploy is to implement the shrinking strategy. The strategy works on best site following a threshold number of repetitions. The tactic works out until the repetition stops. If the number of repetitions still improves for a certain number of iterations, then enhancement strategy is used. Finally, if the number of repetitions still goes up for a number of iterations after the use of the enhancement strategy, then the site in question is abandoned and a new site is generated. [5] The following parameter was experimentally utilized for shrinking the neighbourhood size and site abandonment strategy: size of neighbourhood = ngh, shrink constant = sc, abandoned sites = aband_site. Few more parameters are introduced in this strategy. [7]

i) The number of repetitions for each site = keep_point; keep_point records the number of repetitions for all the repeating results
ii) The repetition number of shrinking = rep_nshr; the number of shrinking is the number of repetitions necessary to start the shrinking strategy
iii) The number of repetitions until the end of shrinking = rep_nenh; This depicts the repetitions until the end of the shrinking process, and the beginning of enhancement process.



FLOW CHART OF LOAD BALANCING IN CLOUD COMPUTING BASED ON HONEY BEE BEHAVIOUR.

IV.ADVANTAGES OVER EXISTING ALGORITHMS

Results obtained in the past have shown that the process is the more efficient one, when compared to other pre-existing algorithms. The approach illustrates that there is a significant improvement in average execution time and reduction in waiting time of tasks. [6]

V.CONCLUSION

Finally, in this research, an optimization algorithm derived from the natural foraging behavior of honey bees, the Honey Bee Foraging Algorithm has been dissected, and an improved version of HBFA has been proposed. It has been applied with success on continuous type benchmark functions and compared with other popular optimization techniques. To verify the performance of the proposed algorithm, the following comparison approaches have been used: accuracy analysis, average evaluation and t-test. From the findings of the accuracy analysis and the average evaluation, the new algorithm is found to perform better on higher dimensional than lower dimensional functions. The statistical significance of the new algorithm has been computed using a t-test and the results have been compared with the basic HBFA. Based on the results, it can be inferred that the results of the new algorithm are significantly better than the results of basic HBFA. Thus the proposed algorithm performed better than the basic HBFA on higher dimensional functions such as, Ackley and Griwank.

VI. ACKNOWLEDGEMENT

I would like to thank Prof Sudha S. for providing study material for researching on this topic and the opportunity to complete this assignment.

VII. REFERENCES

[1].R. Golding. Weak-consistency Group Communication and Membership. *PhD thesis*, UCSC, 1992.

[2].Dahlia Malkhi and Douglas B. Terry. Concise version vectors in winfs. Dist. Computing, 20(3):209–219, 2007.

[3].D. Stott Parker and et. al. Detection of mutual inconsistency in distributed systems. *Trans. on Software Engineering*, 9(3): 240–246, 1983.

[4].NunoPregui¸ca, Carlos Baquero, Paulo S´ergio Almeida, Victor Fonte, and Ricardo Gonc¸ alves. Dotted version vectors: Logical clocks for optimistic replication. *CoRR*,abs/1011.5808, 2010.

[5].R. Schwarz and F. Mattern. Detecting causal relationships in distributed computations: *In search of the Holy Grail. Dist. Computing*, 3(7):149–174, 1994.

[6].W. Wang and C. Amza.On optimal concurrency control for optimistic replication.*In Proc. ICDCS*, pages 317–326, 2009.

[7]. Rathi, R., Visvanthan, P., Kanchana, R.,Anitha.A., (2013). Rule extraction using rough set approach. *International journal of applied engineering research*8 (14), 1661-1667.

[8]. Rathi, R., (2016).Hybridization of soft computing framework A survey. *International journal of pharmacy and technology* 8 (1), 3594-3602.

[9]. Senthilkumar, S., &Viswanatham, V. M. (2017).HB-PPAC: hierarchy-based privacy preserving access control technique in public cloud. *International Journal of High Performance Computing and Networking*, 10(1-2), 13-22.

[10]. Senthilkumar, S., & Viswanatham, M. (2014). ACAFD: "Secure and Scalable Access Control with Assured File Deletion for Outsourced Data in Cloud". *Journal of ICT Research and Applications*, 8(1), 18-30.