A survey of Maximizing reliability of distributed computing system with task allocation using machine learning

Labiba Tasfiya Jeba
Department of Computer Science and Engineering (CSE)
School of Data and Sciences (SDS)
BRAC University
Dhaka, Bangladesh
labiba.tasfiya.jeba@g.bracu.ac.bd

Md Sabbir Hossain
Department of Computer Science and Engineering (CSE)
School of Data and Sciences (SDS)
BRAC University
Dhaka, Bangladesh
md.sabbir.hossain1@g.bracu.ac.bd

Md Mustakin Alam
Department of Computer Science and Engineering (CSE)
School of Data and Sciences (SDS)
BRAC University
Dhaka, Bangladesh
md.mustakin.alam@g.bracu.ac.bd

Annajiat Alim Rasel
Department of Computer Science and Engineering (CSE)
School of Data and Sciences (SDS)
BRAC University
Dhaka, Bangladesh
annajiat@gmail.com

***Abstract*—The purpose of this work is to maximize system dependability by addressing the problem of task allocation (i.e., which processor should each task of an application be allocated to) in heterogeneous distributed computing systems. It is well known that the job allocation issue is NP-hard in the strict sense. For this issue, we suggest a novel swarm intelligence approach based on the honeybee mating optimization (HBMO) method. To identify the optimum answer within a manageable computing time, the HBMO-based technique combines the strength of simulated annealing, evolutionary algorithms, and a quick problem-specific local search heuristic. We examine the algorithm's performance for a variety of factors, including the quantity of tasks, the quantity of processors, the proportion of average communication time to average computing time, and the density of task interaction in the applications. By contrasting our approach with previously developed work allocation algorithms for optimizing system dependability published in the literature, its efficacy and efficiency are proved.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

In place of the extremely expensive massively parallel computers, distributed computing systems are now the primary execution environments for parallel applications. These systems frequently feature processing nodes with a wide range of characteristics that may be coupled by various types of communication networks. In comparison to centralized ones, heterogeneous distributed computing systems offer a number of benefits, including high throughput, cost/performance advantages, resource sharing, and extendibility. The task allocation problem, which is a significant factor in the efficiency of executing a parallel program on a heterogeneous system, involves distributing the tasks that make up the application to the system's best-suited processors. An ineffective task allocation might undermine the benefits of parallelization by failing to fully use the capabilities of a distributed system. This issue has received a lot of academic attention, with the focus mostly on performance measures such decreasing the sum of execution and communication time [14] [4] [24] or the turnaround time for applications [4] [20]. Distributed systems by their very nature are more complicated than centralized ones. Failures of networks and machines are inevitable in such a huge system and may negatively impact any applications running on it. Therefore, assuring distributed systems' dependability is crucial, coupled with work distribution, especially for some vital applications.

To achieve fault tolerance and subsequently dependability in this setting, system redundancy has a long history [8] [10] [12] [15] [18] [22]. A distributed computing system is redundant if it has hardware redundancy (such as several processors at a processing node and multiple communication links linking each pair of processing nodes) as well as software redundancy (such as job replication among processing nodes) [12]. This strategy is pricey, though. Additionally, system redundancy is frequently unavailable or impractical. Therefore, if we want to attain high system dependability, we must use job allocation approach. This study examines the work allocation issue that seeks to maximize distributed systems' dependability without adding any more hardware or software expenditures.

II. Literature Review

Exact and approximate ways to maximizing distributed system dependability without redundancy have been presented in recent decades [6] [15] [21] [25]. The state space search algorithms, first proposed by [21] and later improved by

[15], are the most effective exact algorithms for this problem. They use the concept of branch and bound with underestimates and module independence to lower the average computational effort in determining the best task distribution. However, it has been shown that the computational complexity of the dependability issue in distributed systems is NP-hard in the strict sense [15]. Due to the exponential rise in computation time and memory requirements as issue size rises, these precise techniques can only handle instances of problems of a reasonable size. Heuristic algorithms, on the other hand, may arrive at close to ideal or optimal answers in a reasonable amount of time. Heuristic search algorithms are therefore required in practice. Most current research has been on creating meta-heuristic algorithms to address the issue. In order to rapidly locate a close-to-optimum allocation, Vidyarthi and Tripathi (2001) suggested a solution process employing a simple genetic algorithm (GA). In order to solve the issue, [6] created a simulated annealing (SA) algorithm and successfully compared its performance to that of the branch-and-bound method. A hybrid approach that combines particle swarm optimization and a hill climbing heuristic was proposed by [25]. They said that for the test-cases examined, their method outperformed a GA in terms of efficacy and efficiency. Given the difficulty of the work allocation problem and the rising demand for distributed computing, it is advisable to look at other methods for creating effective heuristic algorithms for the issue. In this research, we offer a novel method for handling the job assignment issue with the aim of optimizing the system dependability, based on the honeybee mating optimization (HBMO) strategy [1] [2] [3].

A. Methodology

The author employed HBMO, a fresh approach to swarm intelligence that was first put out by Abbass in 2001, to resolve the 3-Sat puzzle [1] [2]. Since then, it has been successfully used to address a number of computer science and engineering issues, including cluster analysis , traveling salesman problem [19], and partitioning and scheduling problem in the design of embedded systems [17]. Utilizing HBMO is appealing because of its natural metaphor, ease of use, and high quality solutions. As a result, we want to broaden the applicability of HBMO to the issue of work distribution in distributed computing. There are now a variety of HBMO variations, which vary in the quantity of workers, queens, and a simulated annealing process. The single queen, single worker model [1] is covered in this section. Fig. 2's simple flowchart, which depicts several fundamental facets of the model, serves as an overview of the method.

There is just one queen, one worker, drones, and broods in this algorithm. The artificial queen has a genotype, or collection of genes, that might be seen as providing a comprehensive answer to the issue at hand. The queen also possesses a spermatheca where she stores the sperms of drones she encounters during mating flights. Artificial drones have just half of a genotype, much as haploid actual drones. In the artificial model, a drone has a genotype that fully resolves the issue being studied and a mask that is used to hide half of the genes chosen at random. The drone's sperm is made up of the non-masked genes. Additionally, drones are produced apart from the queen in order to prevent inbreeding; as a result, it is considered that they are unrelated [3]. A worker bee is a non-reproductive female bee that supports a bee colony by carrying out certain activities. Because of this, the artificial worker bee's purpose is limited to caring for the brood. As a result, the worker is used as a problem-specific local search heuristic, whose goal is to enhance the genotypes of broods in order to find better solutions. Each of the several methods that make up the HBMO algorithm corresponds to a distinct stage in the honeybee reproductive process. An optimization problem's solution space may be thought of as a sequence of state transitions, which can be compared to mating flight. According to a probability rule, the queen mates with the drone she encounters in each state as she advances from one to the next based on her speed. The energy content of the queen is initially increased before each mating trip, but it decreases during the flight. The queen returns to her nest when the energy reaches a crucial level or when its spermatheca is full [2].

New broods (trial solutions) are produced by mating the haploid genotypes of the chosen drones with those of the queen, which may then be enhanced by using the worker to carry out local search. If the latter is superior to the former, the queen is then replaced with the fittest brood. After then, the surviving broods are exterminated, and a fresh mating flight is started up until a termination condition is satisfied. Through successive generations, the reproduction process tends to improve the genotype of the queen, which enhances the effectiveness of the current problem's solution.

In this part, the whole method is provided after a brief presentation of a few HBMO components tailored to the work allocation for maximizing distributed systems' dependability.

III. Analysis

We have chosen to test the proposed HBMO algorithm by simulating a wide range of scenarios similar to those used by other researchers [6] [15] [21] [25]. This is because there are no generally accepted standard benchmark tests for the evaluation of task allocation algorithms with the goal of maximizing reliability in heterogeneous distributed computing systems. The properties of the produced issue cases are controlled by a set of parameters that are varied to produce a huge simulation dataset. The following list of key parameters is provided:

- The quantity of tasks in a TIG application (N).
- A distributed computing system's processor count (P).
- Variation coefficient (COV). It is the ratio of the mean task execution time to the standard deviation of task execution times. COV, which describes the variation in task execution durations on various processors, is a measure of how heterogeneous workloads and processors are.
- The cost of communication to computation ratio (CCR). It measures the proportion between average communication and

computation costs. A TIG is categorized as a computation-intensive application if its CCR is low; a communication-intensive application if its CCR is high.

• Task interplay volume (D). The task interaction density estimates the ratio of the inter-task communication required for a TIG and may function as one of the major variables that influence the issue complexity. It measures the likelihood that there will be an interaction between two tasks.

To test the approach with various issue sizes, we conducted tests where the number of processors P in heterogeneous distributed computing systems is modified as 6, 8, and the number of jobs N as 10, 20, and 30. We take into account nine possible TIGs with three different task interaction density values (0.2, 0.5, and 0.8) and three different CCR values (0.5, 1.0, and 2.0) for each pair of (N, P). Ten issue instances are randomly created for each problem set, yielding a total of 540 unique problem instances in our dataset that covers a wide spectrum of distributed computing system applications and eliminates bias towards one particular allocation technique. The COV based technique presented in [5] is used to create the predicted execution times of all jobs while taking heterogeneity into account since it offers more control over the spread of execution time values than the typical range based method [7]. The estimated time to compute matrix, ETC = EipNP, indicates how long job I took to complete on processor p. First, a gamma-distributed task vector with a mean of 20 seconds and a COV of 0.9 is constructed, with its size equal to the number of tasks (task heterogeneity). Then, using each element of the task vector as the mean and a COV of 0.9 (machine heterogeneity), the ETC values for each task on all machines are generated using a gamma distribution. The execution time matrix has no unique structure. In other words, the fact that processor p outperforms processor q on job I does not necessarily indicate that they will perform similarly on other tasks. The uniform distributions across the following ranges are used to generate random values for the other system parameters: the failure rate of processors and communication links is yielded in the ranges (0.00005, 0.00010) and (0.00015, 0.00030), respectively; the memory requirement of each task ranges from 5 to 15, and the memory capacity of each processor varies from /M t to /M t; the transmission rates of links are assumed to be in the range (1, 10); the volume of data to be transmitted among tasks are randomly generated such that the communication to computation ratio (CCR) is 0.5, The same approach is used to create both the processing load needs for jobs and the processing power of processors.

We applied a SA from [6] and a hybrid PSO (HPSO) from [25] for comparative purposes since all of them are newly suggested evolutionary algorithms. This allowed us to evaluate the efficacy and efficiency of the HBMO method. All three methods are written in MATLAB 6.5 and operate on Windows XP on a 1.6 GHz Pentium Dual processor with 1 GB of main memory. The spermatheca size of the queen is set at 5, the cooling rate in Eq. (11) is set at 0.9, and the number of broods is assumed to be equal to the spermatheca size of the queen. When the algorithm reaches its maximum number of iterations (which is equal to the number of jobs in the application to be assigned) or when the best solution cannot be improved further after a sufficient number of iterations, the algorithm is stopped (this number is set to 10). We cite [6] [25] for comprehensive details on the SA and HPSO implementations.

The quality of the solutions (the system dependability) and the necessary execution time utilized for the benchmarks are the performance parameters taken into account. We run each method 20 times for each issue instance since all the algorithms use stochastic techniques, which might cause each separate run of the algorithm on a certain testing problem case to have a different answer. The comparative findings obtained using the three techniques are shown in Tables 1 and 2. The captions "Ravg," "Rstd," and "Tavg" in the tables that follow indicate, in turn, the average system reliability attained by the corresponding algorithm, its standard deviation, and the average execution time (in seconds) required by the corresponding algorithm to solve the problem instance created for the experiment.

The average running times for the SA, HPSO, and HBMO algorithms for the benchmark under consideration are divided by the average running time of the HBMO algorithm to provide the normalized running timings. As can be seen from the tables, the HBMO algorithm outperforms the HPSO algorithm by a small margin. In terms of the mean performance across all experiments, the superiority of HBMO over SA is more pronounced, and there are no problems for which the SA or the HPSO could report performance gains over the HBMO algorithm. For six processor systems, the HBMO algorithm is on average 2.914% better than the SA and the HPSO in terms of solution quality, while for eight processor systems, the HBMO algorithm is on average 3.388% better than the SA and the HPSO, respectively. This shows that the HBMO is more effective than the SA and the HPSO on the test-cases studied. When it comes to execution time, the algorithms' times get longer as the number of tasks rises. For systems with six processors, the SA method executes 1.71 times slower on average than the HBMO approach, while the HPSO algorithm executes 1.22 times slower than the HBMO strategy for systems with eight processors. The HBMO also performs best in terms of the system reliability's standard deviation. These findings show that the suggested HBMO algorithm is a workable substitute for job assignment in the heterogeneous distributed computing system with the aim of optimizing dependability.

The plotted values are the average percentage improvement in system reliability for the HBMO algorithm over the SA and the HPSO algorithms for 10 simulated experiments on each of the 27 scenarios, respectively. Fig. 1 is a more detailed comparison of the HBMO performance to the SA and HPSO performances. There is no discernible difference between the three algorithms for small-scale problem instances, according to trends in the mean performance improvements of the HBMO over the SA and HPSO. However, as the number of tasks, task interaction density, or CCR increases, HBMO outperforms SA and HPSO

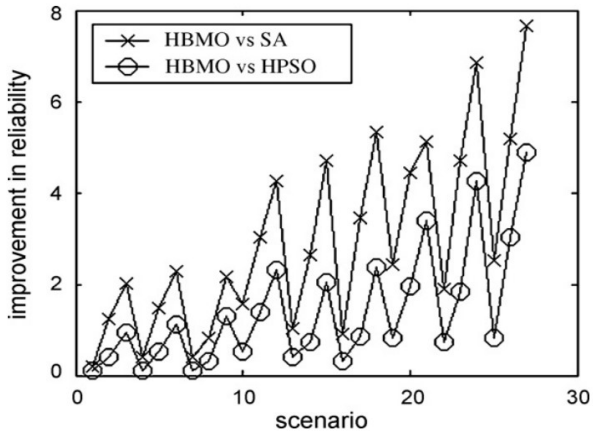

Fig. 1. HBMO vs HPSO Performance Comparison

significantly.

In compared to SA and HPSO, the findings show that HBMO is better suited to large-scale work allocation problems.

The following analysis may be used to determine why the HBMO algorithm performs better than the SA and the HPSO in terms of efficacy and efficiency. In order to achieve a suitable balance between exploration and exploitation, the HBMO algorithm combines the advantages of meta-heuristics with the problem-specific local search heuristic. First, the algorithm can efficiently explore the solution space of the issue under inquiry since the chosen solutions (drones) are created independently of the currently best answer (the queen). Additionally, a well-designed local search heuristic may provide a local optimization result, allowing it to efficiently refine the search result. It may also be used to enhance the performance of current approximation work allocation techniques.

A. Conclusion and Future work

This study, to the best of our knowledge, is the first to apply the HBMO method to the job allocation issue in heterogeneous distributed computing systems with the goal of optimizing dependability. A number of randomly generated mapping issue examples are used to compare the performance of the proposed approach with a SA and an HPSO algorithm. The findings demonstrated that for all test situations, the HBMO algorithm's solution quality was superior to that of the SA and the HPSO. In addition, the HBMO algorithm operates more quickly than the SA and the HPSO. The task allocation issue can be solved using the HBMO approach, according to these findings. It would be interesting to adapt our method in future study to address challenges involving multi-objective task assignment or other performance criteria.

REFERENCES

[1] Abbass, H.A., 2001a. A single queen single worker honey-bees approach to 3-SAT. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 807–814.

[2] Abbass, H.A., 2001b. A monogynous MBO approach to satisfiability. In: International Conference on Computational Intelligence for Modeling, Control and Automation, pp. 207–214.

[3] Abbass, H.A., 2001c. Marriage in honey bees optimization (MBO): a haplometrosis polygynous swarming approach. In: Proceedings of the Congress on Evolutionary Computation, pp. 207–214.

[4] Ahmad, I., Dhodhi, M.K., 1995. Task assignment using problem-space genetic algorithm. Concurrency: Practice and Experience 7, 411–428.

[5] Ali, S., Siegel, H.J., et al., 2000. Representing task and machine heterogeneities for heterogeneous computing systems. Tamkang Journal of Science and Engineering 3 (3), 195–207.

[6] Attiya, G., Hamam, Y., 2006. Task allocation for maximizing reliability of distributed systems: a simulated annealing approach. Journal of Parallel and Distributed Computing 66, 1259–1266.

[7] Braun, T.D., Siegel, H.J., et al., 2001. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing system. Journal of Parallel and Distributed Computing 6, 810–837.

[8] Charles Elegbede, A.O., Chu, C., Adjallah, K.H., Yalaoui, F., 2003. Reliability allocation through cost minimization. IEEE Transactions on Reliability 52, 106–111.

[9] Chen, W.H., Lin, C.S., 2000. A hybrid heuristic to solve a task allocation problem. Computer and Operations Research 27, 287–303.

[10] Chiu, C.C., Yeh, Y.S., Chou, J.S., 2002. A fast algorithm for reliability-oriented task assignment in a distributed system. Computer Communications 25, 1622–1630.

[11] Fathian, M., Amiri, B., Maroosi, A., 2007. Application of honey bee mating optimization algorithm on clustering. Applied Mathematics and Computation 190, 1502–1513.

[12] Hsieh, C.C., 2003. Optimal task allocation and hardware redundancy policies in distributed computing systems. European Journal of Operational Research 147,430–447.

[13] Hsieh, C.C., Hsieh, Y.C., 2003. Reliability and cost optimization in distributed computing systems. Computer and Operations Research 30, 1103–1119.

[14] Kafil, M., Ahmad, I., 1998. Optimal task assignment in heterogeneous distributed computing systems. IEEE Concurrency 6, 42–51.

[15] Kartik, S., Murthy, C.S.R., 1995. Improved task-allocation algorithms to maximize reliability of redundant distributed computing systems. IEEE Transactions on Reliability 44, 575–586.

[16] Kartik, S., Murthy, C.S.R., 1997. Task allocation algorithms for maximizing reliability of distributed computing systems. IEEE Transactions on Computers 46, 719–724.

[17] Koudil, M., Benatchba, K., et al., 2007. Using artificial bees to solve partitioning and scheduling problems in codesign. Applied Mathematics and Computation 186(2), 1710–1722.

[18] Kumar, A., Agrawal, D.P., 1993. A generalized algorithm for evaluating distributedprogram reliability. IEEE Transactions on Reliability 42, 416–426.

[19] Marinakis, Y., Marinaki, M., 2009. A hybrid honey bees mating optimization algorithm for the probabilistic traveling salesman problems. In: IEEE Congress on Evolutionary Computation (CEC 2009), pp. 1762–1769.

[20] Salman, A., Ahmad, I., Al-Madani, S., 2002. Particle swarm optimization for task assignment problem. Microprocessors and Microsystems 26, 363–371.

[21] Shatz, S.M., Wang, J.P., Goto, M., 1992. Task allocation for maximizing reliability of distributed computer systems. IEEE Transactions on Computers 41, 1156–1168.

[22] Tom, P.A., Murthy, C., 1998. Algorithms for reliability-oriented module allocation in distributed computing systems. Journal of System and Software 40, 125–138.

[23] Vidyarthi, D.P., Tripathi, A.K., 2001. Maximizing reliability of distributed computing systems with task allocation using simple genetic algorithm. Journal of Systems Architecture 47, 549–554.

[24] Yin, P.Y., Yu, S.S., Wang, P.P., Wang, Y.T., 2006. A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems. Computer Standard Interface 28, 441–450.

[25] Yin, P.Y., Yu, S.S., Wang, P.P., Wang, Y.T., 2007. Task allocation for maximizing reliability of a distributed system using hybrid particle swarm optimization. Journal of System and Software 80, 724–735.