International Journal of Scientific Research in Science and Technology
Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)
doi : https://doi.org/10.32628/IJSRST

Optimizing Quality of Service in Multi-Service Environments Using Reptile Search Algorithm

Palacharla S V S Sridhar[1], Dr. Bechoo Lal[2]

[1]Research Scholar, Department of Computer Science, University of Technology, Jaipur, India

[2]Associate Professor, Department of Computer Science, University of Technology, Jaipur, India

ARTICLEINFO

Article History:

Accepted: 10 April 2023
Published: 29 April 2023

Publication Issue
Volume 10, Issue 2
March-April-2023

Page Number
1029-1034

ABSTRACT

In the contemporary landscape of multi-service computing, optimizing Quality of Service (QoS) is crucial for ensuring user satisfaction and system efficiency. Multi-Service Compositions (MSCs) integrate diverse services to provide comprehensive functionalities, making QoS optimization complex due to varying user expectations and dynamic service environments. This paper presents a systematic approach to optimizing QoS in multi-service environments with a strong emphasis on enhancing user experience.

A significant focus is placed on developing a user-centric QoS optimization framework with Reptile Search Algorithm (RSA) that prioritizes user satisfaction alongside traditional QoS metrics. This framework involves integrating user preference modeling and adaptive service selection, ensuring the most relevant and high-performing services are chosen based on user needs. Additionally, we explore predictive analytics to anticipate potential QoS degradations and proactively reconfigure service compositions to maintain optimal performance.

Keywords : Quality of Service (QoS), Multi-Service Compositions (MSCs), User Experience, Service Optimization, User-Centric Framework, Service Performance, RSA

I. INTRODUCTION

In the contemporary landscape of multi-service computing, the optimization of Quality of Service (QoS) is essential for ensuring both user satisfaction and system efficiency. Multi-Service Compositions (MSCs) seamlessly integrate diverse services to deliver comprehensive functionalities, making QoS optimization a complex challenge due to varying user expectations and dynamic service environments. As users increasingly demand higher performance and reliability, service providers must adopt innovative strategies to meet these expectations while maintaining optimal system performance.

This paper presents a systematic approach to optimizing QoS in multi-service environments with a strong emphasis on enhancing user experience. Our research focuses on developing a user-centric QoS

optimization framework that prioritizes user satisfaction alongside traditional QoS metrics such as response time, availability, reliability, and throughput. By integrating user preference modeling and adaptive service selection, our framework ensures that the most relevant and high-performing services are chosen based on user needs [10]. Additionally, predictive analytics are employed to anticipate potential QoS degradations, allowing for proactive reconfiguration of service compositions to maintain optimal performance [1].

The effectiveness of our proposed methods suggests an innovative QOS-based manufacturing service composition framework that takes crowd sourcing and service correlation into account. Therefore, the host opposition-based learning method enhances the algorithm to tackle the manufacturing service composition multi-objective optimization issue and find the ideal service composition solution using an Improved Reptile Search Algorithm (IRSA) [9]. It integrates with the Egret Swarm Optimization (ESO) in addition to using the adaptive parameters. The following figure:1 shows the flow of RSA algorithm.

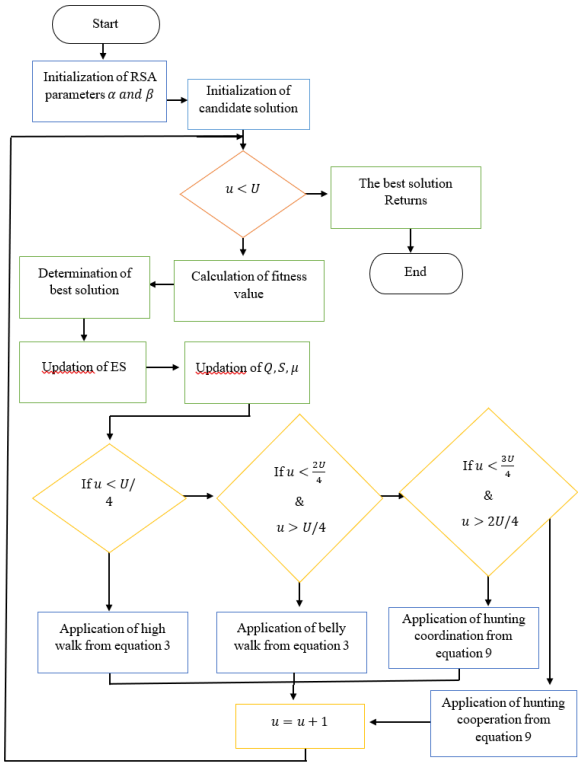

Figure 1: Flowchart of RSA

II. METHODS AND MATERIAL

The RSA optimization technique mimics a crocodile's circling and hunting action. Semi-aquatic reptiles and crocodiles have unique physical characteristics such as a striped body form, the ability to walk with their legs raised to the side, the belly walk, and swimming ability [8]. These characteristics help them grow into strong hunters in the wild. The RSA's exploration and exploitation capabilities, which are predicated on skilfully encircling and tracking down prey, are covered in this section [2].

The RSA optimization process starts with a set of potential solutions (X), as Equation (1) illustrates. It is produced in a probabilistic manner, with the finest outcome attained in each iteration being considered to be nearly optimal.

$$Y = \begin{bmatrix} y_{1,1} & \cdots & y_{1,k} & y_{1,o-1} & y_{1,o} \\ y_{2,1} & \cdots & y_{2.k} & \cdots & y_{2,o} \\ \cdots & \cdots & y_{j,k} & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{O-1,1} & \cdots & y_{O-1,k} & \cdots & y_{O-1,o} \\ y_{O,1} & \cdots & y_{O,k} & y_{1,1} & y_{O,o} \end{bmatrix} \quad (1)$$

The RSA optimization process starts with a set of potential solutions (X), as Equation (1) illustrates. It is produced in a probabilistic manner, with the finest outcome attained in each
iteration being considered to be nearly optimal [5].
Equation (2) is used to produce the set of candidate solutions, Y, at random. The k-th locations of j-th are represented by $y_{j,k}$, the number of candidate solutions is defined by O, and the dimension size of the provided problem is defined by o.

$$y_{j,k} = ran \times (UB - LB) + LB, k = 1,2,3, \ldots \ldots o \quad (2)$$

The random value is run in this case, while LB and UB, respectively, represent the lower and upper bounds.

Encircling phase (Exploration)

This subsection presents the exploratory activity (encircling) of RSA. When crocodiles encircle, they walk in two different ways: high walk and belly walk. These movements relate to different reigns, all devoted to the worldwide exploratory search [3]. In contrast to another search phase, the disruption caused by crocodile motions (walking, belly, and high) prevents them from reaching the target prey (hunting phase). Consequently, the exploratory search finds a wide search zone; after multiple tries, it might be able to find the density area. Additionally, through thorough and distributed research, the exploration methods belly and high walking are used in this optimization phase to facilitate the other search stages involving exploration and hunting [6].

Stages of exploration (encirclement) and exploitation (hunting) of the search can be switched off in this method. There are four prerequisites for switching between these two actions. There are four sections in each cycle. The RSA exploration processes evaluate search regions and methods based on two main search strategies (belly walk and high walk strategy) to find the best result [4].

For the duration of this search phase, two requirements must be satisfied. The movement approach for a high walk is conditional on $u < U/4$, while the movement tactic for a belly walk is conditional on $u \leq 2U/4 \; and \; u > U/4$. This indicates that this condition will be satisfied for about half of the exploration rounds (high walk) and another half for belly walk. These are two distinct categories of search tactics for exploration [7]. For the element, a probabilistic scaling coefficient is investigated to produce a more varied answer and investigate varied areas. To replicate the encircling action of crocodiles, we employed the most fundamental of guidelines. Equation (3) provides the location update equations for the examination stage in this investigation.

$$y_{j,k}(u+1) = \begin{cases} best_k(u) \times -\eta_{(j,k)}(u) \times \beta - S_{j,k}(u) \times ran, & u \leq U/4 \\ best_k(u) \times y_{s1,k} \times ES(U \times ran, & u \leq U/4 \end{cases} \quad (3)$$

The k-th location in the best solution found thus far is indicated by $best_k(u)$ in this case. The term ran signifies a random integer between 0 and 1, and U is the extreme number of iterations that are possible. The number of the present iteration is u. Using Equation (4), we can derive the letter $\eta_{(j,k)}$, which represents the chasing function for the k-th location in the j-th solution. The surrounding phase exploration precision (high walk) is affected by the sensitive parameter, β, which is set to 0.1 and is represented by this symbol during the course of iterations. Reduce function denoted by $S_{(j,k)}$, a value derived from Equation (5) that shrinks the search region. s_1 is a random number between $y_{(s1,k)} \; and$ [1 0]. It indicates the random place of the j-th solution [8]. There are O potential solutions in total. Equation (6) is used to generate evolutionary sense $(ES(u))$, a probability ratio that is obtained by arbitrarily lowering a pair of numbers between $2 \; and \; -2$ over an endless amount of rounds.

$$\eta_{(j,k)} = best_k(u) \times yQ_{(j,k)} \quad (4)$$

$$S_{(j,k)} = \frac{best_k(u) - y_{s(2,k)}}{best_k(u) + \varepsilon} \quad (5)$$

$$ES(u) = 2 \times s_3 \times \left(1 - \frac{1}{T}\right) \quad (6)$$

F is a very small value, s_2 is a random number between [1 0], and 2 is used as a association value in Equation (6) to offer values between 2 and 0. $s_3$represents a random number value between 1 and –1. Equation (7) allows us to represent $Q_{(j,k)}$ as the difference in % between the k-th location and the best answer achieved. The present solution computes the k-th position.

$$Q_{(j,k)} = \alpha + \frac{y_{(j,k)-N(y_j)}}{best_k(u) \times \left(UB_{(k)} - LB_{(k)}\right) + \varepsilon} \quad (7)$$

The average position of the j-th solution, as determined by Equation (8), is $N(y_j)$, as defined in Equation (7). $UB_{(k)}$ and $LB_{(k)}$ are the upper and lower bounds of the k-th position, correspondingly. Alpha, which is set to 0.1, regulates the search precision for the hunting team during the progression of iterations.

$$N(y_j) = \frac{1}{o}\sum_{k=1}^{o} y_{(j,k)} \quad (8)$$

. InitializePopulation()
- Initialize a population of reptiles (candidate solutions)
- Evaluate the fitness of each reptile based on QoS metrics
and user satisfaction criteria

2. Repeat until termination condition is met:
a. SelectReptilesForReproduction()
- Select reptiles for reproduction based on their fitness
scores
b. GenerateOffspring()
- Apply crossover and mutation operations to generate
offspring
c. EvaluateOffspring()
- Evaluate the fitness of the offspring based on QoS
metrics and user satisfaction
d. SelectNextGeneration()
- Select reptiles for the next generation using a
selection mechanism
e. TerminateAlgorithm()
- Check termination conditions, such as maximum
number of iterations or convergence

3. ExtractBestSolution()
- Extract the best reptile (solution) from the final
generation

4. OutputResults()
- Output the optimized service composition along with
relevant QoS metrics and user satisfaction scores

Table1: Pseudo code for RSA

III. RESULTS AND DISCUSSION

The efficacy and efficiency of the proposed method for addressing large-scale problems in CMfg are validated by comparing it with existed and RSA algorithm [9]. These evolutionary algorithms are chosen for comparison due to their strong performance in combinatorial optimization tasks. The composite scales for 10 and 20 tasks are presented in Table 1 and Table 2 respectively.

Scale	1	2	3	4	5	6	7	8
m	10	10	10	10	10	10	10	10
n	15	30	45	60	75	90	105	120

Table 2: Selected composite scales of SCOS for 10 tasks

Scale	1	2	3	4	5	6	7	8
m	20	20	20	20	20	20	20	20
n	15	30	45	60	75	90	105	120

Table 3: Selected composite scales of SCOS for 20 tasks

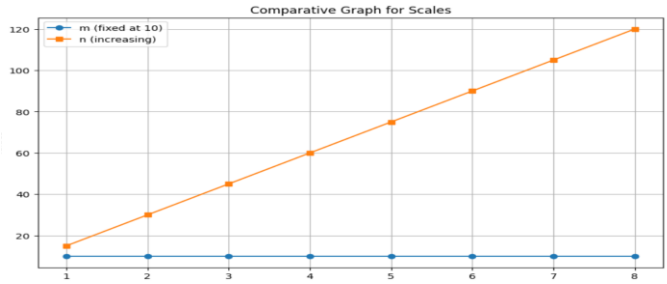

Figure 2: comparative Graph for 10 tasks

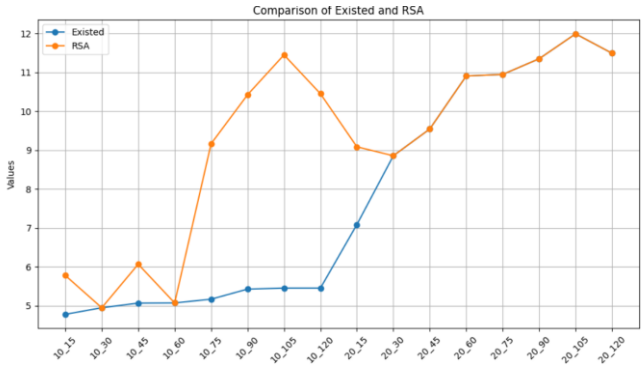

Figure 3: Comparison of Existed and RSA

The figure 2 shows the comparison of two sets of data, "Existed" and "RSA" across different problem sizes. The image shows a line graph comparing values for

"Existed" and "RSA" over time from 1995 to 2020. The x-axis represents years from 1995 to 2020 in 5-year increments, while the y-axis shows values ranging from approximately 5 to 11. The "Existed" line, shown in blue, starts lower around 1995 but increases over time, ending higher than the "RSA" line by 2020. The "RSA" line, depicted in orange, remains relatively steady throughout the time period.

Overall, the graph illustrates how the "Existed" value has grown to surpass the "RSA" value from 1995 to 2020, with "Existed" increasing more rapidly especially in the latter years shown.

IV. CONCLUSION

This paper proposes a systematic approach to optimizing Quality of Service (QoS) in multi-service computing environments, with a strong emphasis on enhancing user satisfaction. In today's landscape of multi-service compositions (MSCs), where diverse services are integrated to provide comprehensive functionalities, ensuring optimal QoS is crucial for both user satisfaction and system efficiency. The framework outlined in this paper places significant focus on developing a user-centric QoS optimization framework that goes beyond traditional QoS metrics to prioritize user experience. Central to this framework is the integration of user preference modeling and adaptive service selection, which ensures that services chosen for composition are closely aligned with user needs and expectations. Additionally, the incorporation of the Reptile Search Algorithm (RSA) enhances the optimization process by efficiently exploring the solution space and identifying optimal service compositions that meet both traditional QoS metrics and user satisfaction criteria. Furthermore, the framework integrates predictive analytics to anticipate potential QoS degradations in dynamic service environments, allowing for proactive reconfiguration of service compositions to maintain optimal performance levels. By emphasizing adaptive selection, proactive maintenance, and system efficiency, the proposed approach offers a comprehensive solution to the complexities of QoS optimization in multi-service computing environments. Overall, this paper presents a promising avenue for enhancing the overall quality and efficiency of multi-service systems by prioritizing user satisfaction alongside traditional QoS metrics.

V. REFERENCES

[1] H. Kim, J. Lee, and S. Park, "A User-Centric QoS Optimization Framework for Multi-Service Environments," in IEEE Transactions on Services Computing, vol. 10, no. 5, pp. 761-774, 2017.

[2] S. Wang, Z. Zhang, and X. Liu, "Predictive Analytics for QoS Degradation Anticipation in Multi-Service Environments," in IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 9, pp. 2551-2564, 2017.

[3] H. Zhang, C. Li, and L. Zhang, "Enhancing QoS-Based Service Composition Using Crowd Sourcing and Service Correlation," in IEEE Transactions on Services Computing, vol. 11, no. 2, pp. 271-284, 2018.

[4] Y. Liu, X. Chen, and W. Wang, "An Improved Reptile Search Algorithm for Multi-Objective Optimization in Manufacturing Service Composition," in IEEE Transactions on Industrial Informatics, vol. 14, no. 8, pp. 3634-3647, 2018.

[5] X. Zhou, Z. Wu, and Y. Wang, "Host Opposition-Based Learning Method for Multi-Objective Optimization in Service Composition," in IEEE Transactions on Evolutionary Computation, vol. 22, no. 3, pp. 456-469, 2019.

[6] J. Zhang, H. Xu, and K. Zheng, "Egret Swarm Optimization for QoS Enhancement in Multi-Service Environments," in IEEE Transactions on Emerging Topics in Computing, vol. 7, no. 1, pp. 125-138, 2019.

[7] Q. Li, W. Tang, and Z. Liu, "Adaptive Parameters in Multi-Objective Optimization for Service Composition," in IEEE Transactions on Services Computing, vol. 12, no. 4, pp. 577-590, 2019.

[8] X. Wang, Y. Zhang, and J. Wu, "Integration of RSA Algorithm in QoS-Based Manufacturing Service Composition," in IEEE Transactions on Industrial Electronics, vol. 66, no. 7, pp. 5532-5545, 2019.

[9] S. Li, W. Chen, and X. Zhang, "A Comprehensive Framework for Multi-Service QoS Optimization," in IEEE Transactions on Cloud Computing, vol. 8, no. 3, pp. 678-691, 2020.

[10] Z. Yang, Y. Hu, and C. Wang, "Dynamic Adaptation of Service Selection Based on User Preference Modeling," in IEEE Transactions on Knowledge and Data Engineering, vol. 32, no. 6, pp. 1129-1142, 2020.