An Optimal Execution of Composite Service in Decentralized Environment

Yashwant Dongre, Rajesh Ingle
Department of Computer Engineering, Pune Institute of Computer Technology
Savitribai Phule Pune University, Pune, India

***Abstract*—It is important for service-oriented architectures to consider about how the composition of web services affects business processes. For instance, a single web service may not have been adequate for most complex business operations, needing the use of multiple web services. This paper proposed a novel technique for optimal partitioning and execution of the services using a decentralized environment. The proposed technique is designed and developed using a genetic algorithm with multiple high task allocations on a single server. We compared three existing techniques, including meta-heuristic genetic algorithm, heuristics like Pooling-and-Greedy-Merge (PGM) technique, and Merge-by-Define-Use (MDU) technique, to a simulation of Business Process Execution Language (BPEL) partition using genetic algorithm through multiple high tasks allocation to single server node. The proposed technique is practical and advantageous. In terms of execution time, number of server requests, and throughput, the proposed technique outperformed the existing GA, PGM, and MDU techniques.**

***Keywords*—Genetic algorithm; service composition; decentralized execution; composite service**

I. Introduction

In Service-Oriented Architecture (SOA), web services are the most important and widely implemented technologies which are interoperable machines-to-machines interactions that happen over networks [1]. A Large number of connected heterogeneous devices containing objects are expected to deploy than existing deployed devices in the coming few years. These devices require a reliable connection between them anywhere and forever which provides the ability to collect data. The large availability of devices is advantageous. Traditional services such as traffic control and healthcare are experiencing a shift to a new category of service industry demands.

Decentralized execution environments for Business Process Execution Language (BPEL) processes are necessary due to numerous reasons, starting from the outsourcing of process fragments to the need for runtime performance optimizations without modifying process models [2]. Many factors make affect partitioning or finding an appropriate distribution of the business program or process. This paper focuses on these factors in order to describe them and to offer a high-level outline of a possible process partitioning approach [3]. Whenever multiple software components and service providers are intricate in business programs, a composition of web service is essential to create a composite web service that combines multiple web-based services to collaborate with one another [4]. As the scale and number of web services have increased, many service providers are offering candidate web services that are operationally corresponding but ensure dissimilar levels of non-functional parameter values [5]. Apart from functionalities, as well as non-functional necessities often defined by QoS are employed by web services to come across operator demands [5, 6], and to satisfy business needs, and service level agreements (SLAs) are becoming an integral part of web based composition process of services.

The selection of QoS-responsive web-based services is the process used to choose web services from amongst a set of candidate web services for every activity in a business workflow, such as Web Services Business Process Execution Language (WS-BPEL) [1], designed to optimize the global QoS as a function of customer preferences and constraints. This is well known NP-hard optimization problem. Numerous researches have discussed requirements of QoS for endways systems addressing this issue [7, 8, 9]. The composition of service is the process by which new services get added to existing ones as per the functional requirement of the application. In this process for adding new services, various options may be available with functionally similar services, at that time non-functional parameters will be selection criteria.

The decentralized execution of composite service performs partitioning of programs scripted in BPEL into a number of subprograms that are smaller than the original script. Each subprogram executes on a distinct node or different BPEL node/server [10]. Fig. 1 illustrates a decentralized execution of a BPEL scripted programme. This is partitioned into five subprograms. Then subprograms are deployed on isolated nodes/servers, BS1 to BS5. The communication between them happens by asynchronous messaging. This decentralized composite service's execution is effective for improvement in QoS parameters like throughput and response time of composite service.

The work presented here partitioning a BPEL data-intensive process using a genetic programming approach [11]. The topology used for execution is distributed. It is based on numbers represented in fuzzy logic, where integrated processing and measurable service composition are performed using communication latency and costs within and between partitions. Optimization of service composition is performed by utilizing decision-making through multiple attribute groups and evaluation of cut-set matrix. In [12], using a combination of case analysis and simulations, verify the reliability of the algorithm. [13], this technique allows partitioning well-structured and unstructured processes by using graph transformations based on the representation of process structure graphs.

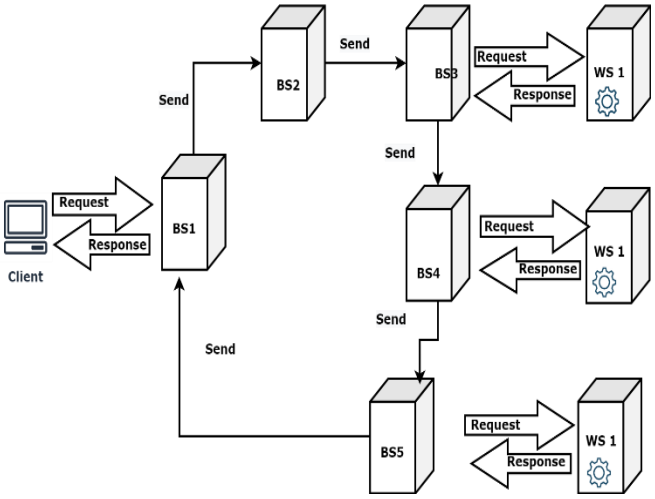

Fig. 1. Decentralized Topology for Execution of Composite Service in Composition.

Partitioning acts as the bridge between ideal and useful parallelism. Program dependencies reveal ideal parallelism through their control and data parallelism [14]. Any two-activity executions that are not related by control or data dependencies either directly or indirectly can be executed in parallel. The parallelism that can be used on a multiprocessor system is a subset of ideal parallelism [15]. Graph partitioning with optimization of performance is an NP-Hard problem in terms of computational complexity [16].

In the above-surveyed papers, no work is focused on assigning multiple high-cost nodes on a single server. The work done in [16] is based on real-time scenarios but any execution topology is unclear. Also, most works on decentralized execution [17] presented hypothetical data and hence the real-time execution of web services composition is needed to increase its acceptance in real-time use. The objectives of the paper are mentioned below.

1) In this paper, the BPEL program can be classified into High-Cost Task (HCT), and Low-Cost Task (LCT) statements.

2) In which, execution of more than one HCTs can be possible on a single server node and comprise the INVOKE, REPLY and RECEIVE statements.

3) The REPLY and RECEIVE statements are mostly kept at the last server node and the first node respectively from where client requests for composite service.

4) The INVOKE statements must be kept for execution on the corresponding web service or nearest location of web service as per availability of load capacity of the server node.

5) A LCT can be kept for execution on any server. LCT tasks are ASSIGN, SEND, IF, etc.

The following is how the rest of the paper is organized: The section two elaborates on related/previous work. Section III describes the proposed mathematically constructed algorithm flow. Section IV discusses the results analysis and discussion. Section V is where the paper's conclusion is found.

II. Related Work

This section paper reviews existing methods and frameworks that researchers have used to partition and execute Composite Services of Composition using a Decentralized Environment.

The author in [10], first introduced the BPEL program execution in the decentralized environment as Program Partitioning Problem. A program dependence graph is used with a set of portable and fixed types of activities. MDU, as well as the PGM heuristic approach, are proposed for partitioning the program [18]. The optimal partitions are determined using the MDU algorithm. The best partition is a partition that executes portable tasks to optimize (maximize) throughput. PGM algorithm is used to reduce the data on the server and the total number of messages to overcome the large computation time of MDU.

As [19], anticipated, Domain-Specific selection of service operates at the communication level which facilitates the application. However, there is a nonexistence of interoperability. Availability, accuracy, response time, and throughput, are considered QoS non-functional parameters. [20] the proposed algorithm is evolutionary which is employed to discover the best-fitting composition of service if the number of services is one. However, recursive processes are implemented when the number of services is one. Different selection strategies will choose different tasks.

Author in [21], survey the techniques of slicing programs that are recycled for performing according to the Program Dependency Graph (PDG).PDG is the greatest common approach in this survey and is effective since it handles data and control dependencies. In BPEL program tasks need allocation of specific servers for execution, due to this existing PDC-based approaches imperfect to apply to the problem of the BPEL program partition.

Author in [22], presents a method for realizing data flows in decentralized Internet of Things (IoT) systems based upon DX-MAN semantics. The algebraic semantics of the model enables direct data links between service producers and consumers of data. As a result, the data space of a decentralized environment is used to write data in and read data out. The authors validated the approach by means of the Blockchain of smart contracts. Results indicate approach is scalable with the growth in IoT systems size.

In [23] traditional techniques for optimization like Multidimensional Multiple-Choice Knapsack(MMCK) and Integer Linear Programming (ILP) are presented to report the problem of QoS-WSC. However, due to time complexity being exponential, these approaches have limited scalability when the problem size is small. To overcome these problems, approximate algorithms based on evolutionary search are proposed to find an optimal solution. Evolutionary computation methods such: Genetic Algorithm (GA) [24]–[26], Ant Colony Optimization (ACO) [27, 28], and Particle Swarm Optimization (PSO) [29], with an extraordinary amount of web-based services and various QoS aspects, algorithms were recycled to catch the service composition with optimal solution plan within a practically undersized period of interval.

By combining all objectives with a function for fitness these methods decrease to a single-objective problem (such as weighted sum techniques and fraction-based techniques).

Author [30], presents the hyper-cube peer-to-peer topology based on distributed system architecture. Efforts are made toward improving the average time and throughput of BPEL processes through the use of decentralized algorithms. The presented algorithms are supported a given BPEL process with decomposition. The presented approach provides a monitoring mechanism. Based on the experimental results, they discovered that the proposed architecture is better suited for long-running, data-intensive processes.

Using typed digraphs and a graph transformation technique [17], propose a technique for creating decentralized service compositions. They discuss the topology and interaction characteristics of the solutions. Based on experimentation, the authors describe a method for ranking topologies. Decentralized compositions are said to have low response times and high throughput on average.

Author [31], provides decentralized execution environments that optimize BPEL-based business processes through the use of shared spaces that represent a communication network among agents (intelligent) and a set of agents (cooperative) to execution of shared services.

Integer linear programming (ILP) takes existed widely used towards address the composition of services issue [32], Pareto dominance [33],QoS constraints decomposition [34], [35], reinforcement learning (RL) [36], or a combination of various techniques. Many evolutionary computation-based algorithms have been developed in recent years [37], [38] and swarm intelligence(SI) based [39], [40]. Compositional approaches that are QoS aware obligate remained proposed so that service compositions for the near-to-optimal solution can be found fairly quickly. The services composition with swarm-based intelligence and evolutionary-based computation approaches are presented in [40], [41], [42]. Researchers propose to achieve a trade-off between service composition with a near-optimal solution and a condensed computation time while achieving compositions of service in standings of QoS parameter optimality.

Meta-heuristics are also called approximation algorithms since they seek to discover the search space using various methods [43]. They do not recompense unusual consideration of optimization problems' mathematical environment or special experience involved with them. In [44], used Genetic Algorithms (GAs) to resolve the web-based service composition problem and demonstrated that GAs can be recycled efficiently for optimizing the composition of web-based services. Furthermore, expanding multiple objectives GA, [45], obtained adequate solutions in an undersized time. However, the difficulty of worst-case GA remains exponential for time complexity, which cannot be used for large applications where scaling is high.

It is complex enough to choose services based on multiple criteria even when one focuses on web composite service with a single user taking place in the pipeline. Few researchers proposed techniques, [46], [47] pay focus on the multi-attribute combinatorial auction (iterative) between various providers for services, whereas [48] pursues to advance these techniques by means of motivation device. We examined the user's QoS preference in the Big Data space, along with their service trust, in order to select services. Recently, [49], analyzed the problem of service composition from the perspective of a general Pareto-optimality to shrink the service composition's search space. An approximation estimate of the Pareto principle of optimality in polynomial time remained used in [50].

Author [51], presents customer authentic cost calculation method named Integrated Multi-Level Composite Service Model. They demonstrated the application of customer management in composition. However, lack in addressing quality parameters. In [52], researchers introduce blockchain architecture for the semantic composition of web services and develop QoS aware algorithm in terms of accuracy but considered QoS like throughput and response time.

The survey shows that most of the existing research works did not handle the optimality issue of service composition problem. Some of the research work on service composition has not even reached up to the phase of execution of composite service which is a major outcome of composition workflows. Therefore, an opportunity for research work on this breach is suitable.

III. Proposed Algorithm

The current section proposed a Multi High Tasks Genetic Algorithm (MHGA) technique which is novel and based on the allocation of multiple high-cost tasks on a single server node for the execution of composite service. This research discovers the usage of the improved genetic algorithm to address the problem of partitioning the BPEL program [13, 20, 22, 24, 26, 37, 44]. In addition, at hand is the absence of an evaluation or assessment model for simulation techniques of the BPEL program for the partitioning problem. This work develops a novel simulation model which is evaluating the effectiveness and efficiency of the new Multi High Tasks Genetic Algorithm. Here presents the proposed MHGA approach architecture and layout.

The BPEL development is self-possessed of a set of declarations called activities. These activities can remain categorized as high-cost activities and low-cost activities. High-cost activities, such as RECEIVE, REPLY, and INVOKE can be deployed and executed particular or nearest server node or engine of workflow. Any workflow engine can be assigned low-cost activities like SEND, ASSIGN, and IF.

A. The Description of Problem

The description of the BPEL program partitioning problem: $G = (HCT, LCT, DD, CD)$, is the program dependency graph is input as shown in Fig. 2. Where, $= \{h_1, h_2, .. h_n\}$, n stands a number of High-cost jobs in program. $LCT = \{l_1, l_2, .. l_m\}$, m stands a number of Low-cost jobs in the program.

A set of data dependencies. $DD = \{< v_i, v_j > \| v_i, v_j \in HCT \cup LCT\}$. A set of control dependencies $CD =$

$\{< v_i, v_j, Cd_p > \parallel v_i, v_j \in HCT \cup LCT \text{ and } Cd_p \in B\{boolean\ values\}\}$.

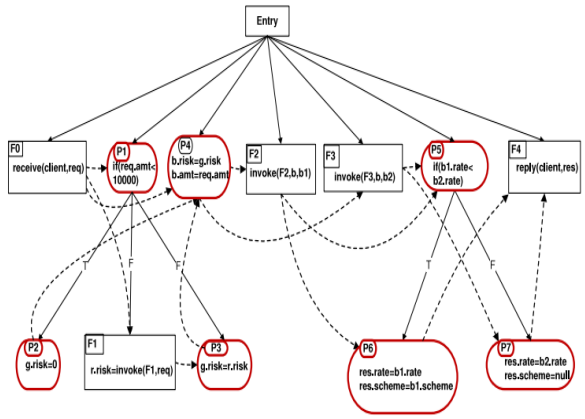

Fig. 2. BPEL Program Partitioning with Dependency Graph Example.

The computation cost on each n server (S) for tasks containing $\{receive, reply, invoke, send, assign\ if\}$ shown by $\{Cost_{receive}(n), Cost_{reply}(n), Cost_{invoke}(n), Cost_{send}(n), Cost_{assign}(n), Cost_{if}(n)\}$.

Throughput of the server is calculated by the server capacity and cost.

$$TH(S_n) = \frac{Capacity_{(n)}}{Cost_{(n)}} \quad (1)$$

Where $TH(S_n)$ is the throughput of the server. $Cost(n)$ is the total computational costs of tasks allotted on server S_n.

$$Cost(n) = \begin{Bmatrix} RE(n) * Cost_{receiver}(n) + RP(n) * Cost_{reply}(n) \\ +IV(n) * Cost_{invoke}(n) + SD(n) * Cost_{send}(n) \\ +AS(n) * Cost_{assign}(n) + IF(n) * Cost_{if}(n) \end{Bmatrix} \quad (2)$$

$Cost(n)$ is total cost on server S_n, where RE, RP, IV, SD, AS, IF are number of receive, reply, invoke, assign, send, if respectively.

$$F_{obj}(X) = min(TH(S_n)) \quad (3)$$

Throughput of plan X, which is minimum throughput among all servers in plan. The output of the plan X such that, $X = \{< L_m, H_n > \parallel H_n \in HCT, L_m \in LCT\}$. LCT and HCT assigned on partition such that Fitness is Maximal i.e., $F_{obj}(X)$ as well as precedence and control dependency constraints are satisfied.

B. Proposed MHGA Technique

The proposed technique is based on allocation of multiple (i.e. more than one) numbers of high cost tasks on single server nodes unlike any previous approaches. While allocation of high cost tasks on server nodes, available capacity of node is considered and checked for eligibility of placement of tasks on node using equation (4), if eq. (4) is true then only allocation of tasks is done on specific nodes. Here required capacity is nothing but cost of high cost task labeled as HCT with as current high cost task Otherwise node is allowed for execution of single task.

$$Available(Capacity(S_n)) \geq Required(Capacity(H_n)) \quad (4)$$

The model shown Fig. 3 is used to represent placement of lost cost and high cost task in solution. Where the integer number below LCT represents the number of low cost tasks. The number below HCT in representation is used for showing the number of high cost tasks allowed for execution on a particular server node.

C. Algorithm

The algorithm is proposed as in Algorithm 1. In step 1 algorithm generates solution with keeping LCT and HCT on server with respective partition. Next step calculates fitness using F_obj (X). Any two best solutions get selected for crossover process. In crossover bits from solution get cross with another solution in selected pair. A mutation operation get performs on offspring generated from crossover. These newly generated solutions get updated in original population. Fitness for updated solution population gets calculated for the selection of population for crossover and mutation in next iteration. This process will get repeated over number of iterations.

Algorithm 1: Program partitioning-genetic algorithm

Input: Program Dependency Graph, Data Dependency Set and Control Dependency Set

Output: Partition Plan X

Initialization: Initialize all population with random solution plan.
Each LCT is placed on any partition.
Each HCT is placed on partition so that maximum capacity will not exceed up to maximum two HCT on each server.
$n \leftarrow 0$;
While $n \leq populationsize$ ***do***
    Calculate fitness of each solution using following fitness formula by Eqn. (3)
    Calculate throughput of the server S_n by Eqn. (1)
    Calculate the cost of the server S_n by Eqn. (2)
    $n = n + 1$;
    end
$Max_{interation}$ =200;
while $Max_{interation}$ ***do***
    Selection: best two population with highest fitness will be selected for next step;
    Crossover: operation with 0.9 probability iscross from two population;
    new two populations get updated from initial population ;
    Mutation: operation with 0.1 probabilities is mutated. one bit from each new solution population gets changed to other value;
    Calculate fitness for updated solution population;
    end
return best execution plan devising the greatest value of fitness;

D. Model of Solution Plan

As shown in Fig. 3, a model of the population as solution plan is represented as Array List with a total of a number of LCTs (m) and a number of HCT's(n).The value in Array List indicates allotted LCT for first m elements and then next n elements allotted HCT in solution as population. An m stands for the total quantity of LCT, n stands for the total quantity of HCTs in solution plan. The value of s ranges from 0 to a total number of servers minus 1 (i.e. s_n-1, wherever s_n stands for the total quantity of servers) for allotment.

Population as solution plan									
LCT_1	LCT_2	LCT_3	LCT_4	...	LCT_m	HCT_1	HCT_2	...	HCT_n
s	s	s	s	...	s	s	s	...	s

m : total number of LCT

n : total number of HCT

s : range from [0 to total number of servers -1] for allotment

Fig. 3. Model of Population as Solution Plan.

IV. Results and Discussion

The proposed work is simulated on the Window 7 64 bits operating system. To carry out this work at least 4GB Ram and Intel Core i3processors are required. The Java Platform (JDK1.8) is used to design a model and calculate the performance parameters. The work focused on analyzing the simulation of proposed techniques (MHGA) for the partition of the BPEL program generated from composite service in service composition.

The proposed MHGA technique performance is compared with the three existing heuristic GA [13], [22], MDU, and PGM [10] techniques. In our simulation, the range of low-cost tasks (LCT) and high-cost tasks (HCT) are varying from 07:05 to 70:50. During the evaluation, computation time is observed in milliseconds keeping the population size of particles fixed to 20and the number of iterations to 200. Other GA parameters setting are crossover =0.9(probability), mutation =0.1 (probability) so that probability of crossover + mutation =1.0. The cost value for receive, reply, invoke, send, assign, if are taken as 0.6, 0.45, 2.5, 0.5, 0.6, 0.6 respectively from benchmark [10].

Fig. 4 shows the performance of the proposed (Multi High Tasks Genetic Algorithm) MHGA algorithm and GA algorithm from existing work. From Fig. 4 and Table I it is observed that the proposed algorithm partitioned the BPEL program for complex applications with high tasks in reasonable time similar to the GA algorithm for varying numbers of LCTs and HCTs. But through a new approach resource (server) utilization will be better and topology with a smaller quantity of server node scan be simple than existing approaches. As existing approaches use a total number of server nodes in topology equals to the quantity of HCT in the program which is very tedious and impractical for implementation especially in the case of the high number of HCTs.

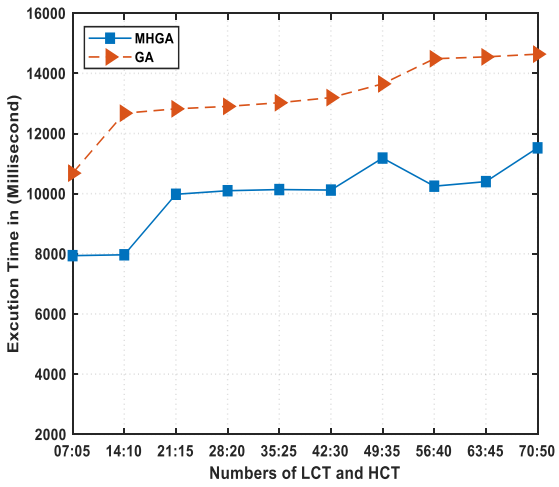

Fig. 4. Comparison of Execution Time between Proposed MGHA Approach with Existing GA with respect to Number of LCT and HCT.

TABLE I. Comparison of Time for Proposed MGHA Approach with Existing GA

Sr. No.	No. of LCT	No. of HCT	Time for MHGA	Time for GA
1	7	5	7940	10682
2	14	10	7968	12675
3	21	15	9980	12821
4	28	20	10098	12902
5	35	25	10137	13024
6	42	30	10120	13194
7	49	35	11184	13650
8	56	40	10249	14485
9	63	45	10401	14548
10	70	50	11523	14639

Our approach reduces the number of server nodes to up to half of the HCTs in the program which is practically possible for implementation. Fig. 5 shows the number of server nodes allotted for the proposed MHGA algorithm vs GA algorithm from existing work. From Fig. 5, it is observed that the proposed algorithm partitions the BPEL program within a smaller number of server nodes than the existing approach.

The simulation results are brief in Fig. 6. As shown, the average amount of throughput (request/seconds) of the business workflow beneath the partitioning solution plan found by the proposed MHGA and GA technique. The proposed MHGA techniques show a higher throughput as compared to the GA techniques.

It can be observed from Fig. 7, the computation time comparison among the proposed MHGA, GA, MDU, and PGM. The proposed MHGA techniques required less computation time as compared to existing techniques such as GA, MDU, and PGM. The proposed techniques performed better with the low and high-cost tasks. The PGM and GA technique also employs much fewer computation times than MDU in the attainment of a solution plan. GA technique also performed better than the existing MDU and PGM techniques. More specifically, MDU and PGM techniques are not addressing optimality issues of

optimization problems like GA-based technique hence detailed comparison for execution time with these approaches is not shown in this work.

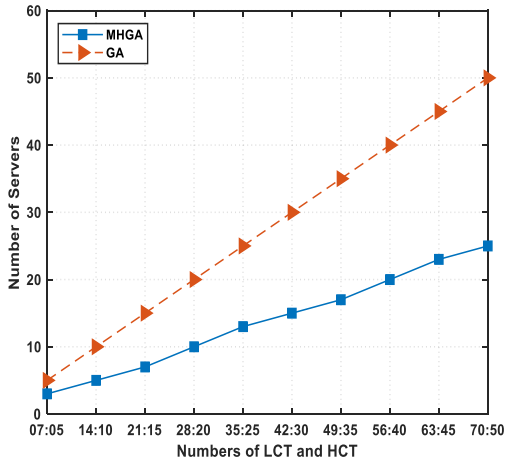

Fig. 5. Comparison of Number of Servers Utilized between Proposed MGHA approach with Existing GA with respect to Number of LCT and HCT.

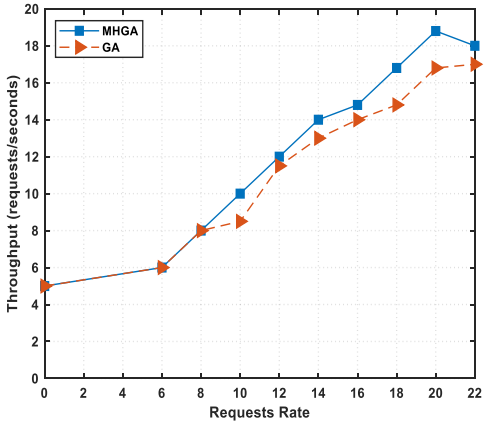

Fig. 6. Comparison of Throughput between Proposed MGHA approach with Existing GA with respect to Request Rate.

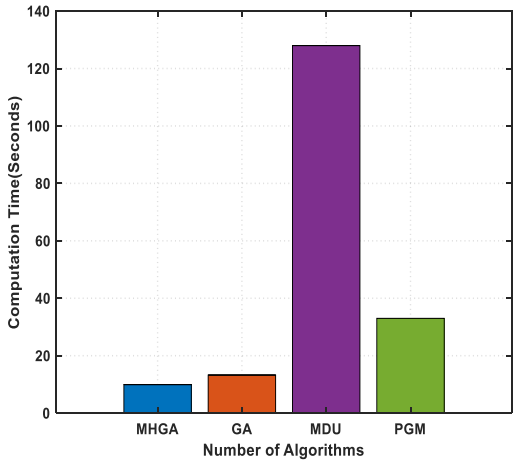

Fig. 7. Comparative Analysis of Average Computation Time between Proposed MGHA approach with Existing GA, MDU and PGM.

V. Conclusion

This paper performed a simulation of BPEL partition using a genetic algorithm through multiple high tasks allocation to a single server node. This work analyzes the existing genetic algorithms for the solution, execution time, and their nature as well as their number of server node requirements for actual execution. From the simulation results, it can be seen that the proposed MHGA technique is suitable for partitioning a large number of tasks in BPEL programs. The server node requirement in terms of quantity for the proposed approach is less than the existing ones; hence execution topology for large composite applications will be simple through the proposed approach. The proposed MHGA technique performed better than the existing GA, MDU, and PGM techniques in terms of the execution time, amount of server requests, and throughput.

In future work, consideration of issues like control dependencies and data dependencies can be explored in more detail for real-time data and complex application scenarios in dynamic environments.

References

[1] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun, "Personalized QoS-Aware Web Service Recommendation and Visualization," IEEE Trans. Serv. Comput., vol. 6, no. 1, pp. 35–47, 2013, doi: 10.1109/TSC.2011.35.

[2] M. B. Juric, B. Mathew, and P. G. Sarang, Business process execution language for web services: an architect and developer's guide to orchestrating web services using BPEL4WS. Packt Publishing Ltd, 2006.

[3] D. Wutke, D. Martin, and F. Leymann, "A method for partitioning BPEL processes for decentralized execution.," in ZEUS, 2009, pp. 109–114.

[4] L.-J. Zhang, J. Zhang, and H. Cai, Services computing. Springer, 2007.

[5] L. Qi, Y. Tang, W. Dou, and J. Chen, "Combining Local Optimization and Enumeration for QoS-aware Web Service Composition," in 2010 IEEE International Conference on Web Services, Miami, FL, USA, Jul. 2010, pp. 34–41. doi: 10.1109/ICWS.2010.62.

[6] L. Barakat, S. Miles, I. Poernomo, and M. Luck, "Efficient Multi-granularity Service Composition," in 2011 IEEE International Conference on Web Services, Washington, DC, USA, Jul. 2011, pp. 227–234. doi: 10.1109/ICWS.2011.25.

[7] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient QoS-aware service composition," in Proceedings of the 18th international conference on World wide web - WWW '09, Madrid, Spain, 2009, p. 881. doi: 10.1145/1526709.1526828.

[8] W. Ahmed, Y. Wu, and W. Zheng, "Response Time Based Optimal Web Service Selection," IEEE Trans. Parallel Distrib. Syst., vol. 26, no. 2, pp. 551–561, Feb. 2015, doi: 10.1109/TPDS.2013.310.

[9] S.-Y. Hwang, C.-C. Hsu, and C.-H. Lee, "Service Selection for Web Services with Probabilistic QoS," IEEE Trans. Serv. Comput., vol. 8, no. 3, pp. 467–480, May 2015, doi: 10.1109/TSC.2014.2338851.

[10] M. G. Nanda, S. Chandra, and V. Sarkar, "Decentralizing execution of composite web services," in Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, Vancouver BC Canada, Oct. 2004, pp. 170–187. doi: 10.1145/1028976.1028991.

[11] Z. Brahmi and I. Feddaoui, "Decentralized orchestration of BPEL processes based on shared space," in 2015 6th International Conference on Information Systems and Economic Intelligence (SIIE), 2015, pp. 60–65.

[12] S. Zheng, D. Feng, and J. Yu, "The algorithm of Web services composition of group decision-making based on fuzzy numbers," in Proceedings of the 3rd International Conference on Computer Science and Application Engineering, 2019, pp. 1–7.

[13] Y. Yu, H. Ma, and M. Zhang, "A genetic programming approach to distributed execution of data-intensive web service compositions," in Proceedings of the Australasian Computer Science Week Multiconference, 2016, pp. 1–9.

[14] J. Ferrante, K. J. Ottenstein, and J. D. Warren, "The program dependence graph and its use in optimization," ACM Trans. Program. Lang. Syst. TOPLAS, vol. 9, no. 3, pp. 319–349, 1987.

[15] V. Sarkar, "Automatic partitioning of a program dependence graph into parallel tasks," IBM J. Res. Dev., vol. 35, no. 5.6, pp. 779–804, 1991.

[16] G. Xue, J. Liu, L. Wu, and S. Yao, "A graph based technique of process partitioning," J. Web Eng., pp. 121–140, 2018.

[17] M. Pantazoglou, I. Pogkas, and A. Tsalgatidou, "Decentralized enactment of BPEL processes," IEEE Trans. Serv. Comput., vol. 7, no. 2, pp. 184–197, 2013.

[18] T. Mohsni and Z. Brahmi, "Partitioning BPEL program for decentralized execution based on Swarm Intelligence".

[19] O. Moser, F. Rosenberg, and S. Dustdar, "Domain-Specific Service Selection for Composite Services," IEEE Trans. Softw. Eng., vol. 38, no. 4, pp. 828–843, Jul. 2012, doi: 10.1109/TSE.2011.43.

[20] S.-C. Liu and S.-S. Weng, "Applying genetic algorithm to select web services based on workflow quality of service," J. Electron. Commer. Res., vol. 13, no. 2, p. 157, 2012.

[21] Y. Katsuno and H. Takahashi, "An Automated Parallel Approach for Rapid Deployment of Composite Application Servers," in 2015 IEEE International Conference on Cloud Engineering, Tempe, AZ, USA, Mar. 2015, pp. 126–134. doi: 10.1109/IC2E.2015.16.

[22] L. Ai, M. Tang, and C. Fidge, "Partitioning composite web services for decentralized execution using a genetic algorithm," Future Gener. Comput. Syst., vol. 27, no. 2, pp. 157–172, 2011.

[23] Y. Shi and X. Chen, "A Survey on QoS-aware Web Service Composition," in 2011 Third International Conference on Multimedia Information Networking and Security, Shanghai, China, Nov. 2011, pp. 283–287. doi: 10.1109/MINES.2011.118.

[24] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "An approach for QoS-aware service composition based on genetic algorithms," in Proceedings of the 2005 conference on Genetic and evolutionary computation - GECCO '05, Washington DC, USA, 2005, p. 1069. doi: 10.1145/1068009.1068189.

[25] W.-C. Chang, C.-S. Wu, and C. Chang, "Optimizing dynamic web service component composition by using evolutionary algorithms," in The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), 2005, pp. 708–711.

[26] M. C. Jaeger and G. Mühl, "QoS-based selection of services: The implementation of a genetic algorithm," in Communication in Distributed Systems-15. ITG/GI Symposium, 2007, pp. 1–12.

[27] L. Aziz, S. Raghay, H. Aznaoui, and A. Jamali, "A new approach based on a genetic algorithm and an agent cluster head to optimize energy in Wireless Sensor Networks," in 2016 international conference on information technology for organizations development (IT4OD), 2016, pp. 1–5.

[28] Q. Wu and Q. Zhu, "Transactional and QoS-aware dynamic service composition based on ant colony optimization," Future Gener. Comput. Syst., vol. 29, no. 5, pp. 1112–1119, Jul. 2013, doi: 10.1016/j.future.2012.12.010.

[29] W. Wang, Q. Sun, X. Zhao, and F. Yang, "An improved particle swarm optimization algorithm for QoS-aware web service selection in service oriented communication," Int. J. Comput. Intell. Syst., vol. 3, no. sup01, pp. 18–30, 2010.

[30] D. Arellanes and K.-K. Lau, "Decentralized Data Flows in Algebraic Service Compositions for the Scalability of IoT Systems," in 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, Apr. 2019, pp. 668–673. doi: 10.1109/WF-IoT.2019.8767238.

[31] G. Xue, D. Liu, J. Liu, and S. Yao, "A process partitioning technique for constructing decentralized web service compositions," Softw. Pract. Exp., vol. 49, no. 10, pp. 1550–1570, 2019.

[32] V. Gabrel, M. Manouvrier, and C. Murat, "Web services composition: Complexity and models," Discrete Appl. Math., vol. 196, pp. 100–114, Dec. 2015, doi: 10.1016/j.dam.2014.10.020.

[33] Q. Yu and A. Bouguettaya, "Efficient Service Skyline Computation for Composite Service Selection," IEEE Trans. Knowl. Data Eng., vol. 25, no. 4, pp. 776–789, Apr. 2013, doi: 10.1109/TKDE.2011.268.

[34] S. X. Sun and J. Zhao, "A decomposition-based approach for service composition with global QoS guarantees," Inf. Sci., vol. 199, pp. 138–153, Sep. 2012, doi: 10.1016/j.ins.2012.02.061.

[35] H. Wang, P. Ma, Q. Yu, D. Yang, J. Li, and H. Fei, "Combining quantitative constraints with qualitative preferences for effective non-functional properties-aware service composition," J. Parallel Distrib. Comput., vol. 100, pp. 71–84, Feb. 2017, doi: 10.1016/j.jpdc.2016.10.013.

[36] H. Wang, M. Gu, Q. Yu, H. Fei, J. Li, and Y. Tao, "Large-Scale and Adaptive Service Composition Using Deep Reinforcement Learning," in Service-Oriented Computing, vol. 10601, M. Maximilien, A. Vallecillo, J. Wang, and M. Oriol, Eds. Cham: Springer International Publishing, 2017, pp. 383–391. doi: 10.1007/978-3-319-69035-3_27.

[37] A. S. da Silva, H. Ma, and M. Zhang, "Genetic programming for QoS-aware web service composition and selection," Soft Comput., vol. 20, no. 10, pp. 3851–3867, Oct. 2016, doi: 10.1007/s00500-016-2096-z.

[38] F. Wagner, F. Ishikawa, and S. Honiden, "Robust Service Compositions with Functional and Location Diversity," IEEE Trans. Serv. Comput., vol. 9, no. 2, pp. 277–290, Mar. 2016, doi: 10.1109/TSC.2013.2295791.

[39] M. S. Hossain, M. Moniruzzaman, G. Muhammad, A. Ghoneim, and A. Alamri, "Big Data-Driven Service Composition Using Parallel Clustered Particle Swarm Optimization in Mobile Environment," IEEE Trans. Serv. Comput., vol. 9, no. 5, pp. 806–817, Sep. 2016, doi: 10.1109/TSC.2016.2598335.

[40] X. Xu, Z. Liu, Z. Wang, Q. Z. Sheng, J. Yu, and X. Wang, "S-ABC: A paradigm of service domain-oriented artificial bee colony algorithms for service selection and composition," Future Gener. Comput. Syst., vol. 68, pp. 304–319, Mar. 2017, doi: 10.1016/j.future.2016.09.008.

[41] C. Jatoth, G. R. Gangadharan, and R. Buyya, "Computational Intelligence Based QoS-Aware Web Service Composition: A Systematic Literature Review," IEEE Trans. Serv. Comput., vol. 10, no. 3, pp. 475–492, May 2017, doi: 10.1109/TSC.2015.2473840.

[42] S. Mistry, A. Bouguettaya, H. Dong, and A. K. Qin, "Metaheuristic Optimization for Long-term IaaS Service Composition," IEEE Trans. Serv. Comput., vol. 11, no. 1, pp. 131–143, Jan. 2018, doi: 10.1109/TSC.2016.2542068.

[43] M. Gendreau and J.-Y. Potvin, "Metaheuristics in Combinatorial Optimization," Ann. Oper. Res., vol. 140, no. 1, pp. 189–213, Nov. 2005, doi: 10.1007/s10479-005-3971-7.

[44] M. A. Amiri and H. Serajzadeh, "QoS aware web service composition based on genetic algorithm," in 2010 5th International Symposium on Telecommunications, 2010, pp. 502–507.

[45] H. Wada, J. Suzuki, Y. Yamano, and K. Oba, "E3: A Multiobjective Optimization Framework for SLA-Aware Service Composition," IEEE Trans. Serv. Comput., vol. 5, no. 3, pp. 358–372, 2012, doi: 10.1109/TSC.2011.6.

[46] Q. He, J. Yan, H. Jin, and Y. Yang, "Quality-Aware Service Selection for Service-Based Systems Based on Iterative Multi-Attribute Combinatorial Auction," IEEE Trans. Softw. Eng., vol. 40, no. 2, pp. 192–215, Feb. 2014, doi: 10.1109/TSE.2013.2297911.

[47] Y. Zhong, X. Li, and Q. He, "Iterative auction based service selection for multi-tenant service-based systems," in Proceedings of the Australasian Computer Science Week Multiconf a single web service may not have been sufficient for most complex business operations, erence, 2017, pp. 1–4.

[48] P. Wang and X. Du, "QoS-Aware Service Selection Using an Incentive Mechanism," IEEE Trans. Serv. Comput., vol. 12, no. 2, pp. 262–275, Mar. 2019, doi: 10.1109/TSC.2016.2602203.

[49] Y. Chen, J. Huang, C. Lin, and J. Hu, "A Partial Selection Methodology for Efficient QoS-Aware Service Composition," IEEE Trans. Serv. Comput., vol. 8, no. 3, pp. 384–397, May 2015, doi: 10.1109/TSC.2014.2381493.

[50] I. Trummer, B. Faltings, and W. Binder, "Multi-Objective Quality-Driven Service Selection—A Fully Polynomial Time Approximation Scheme," IEEE Trans. Softw. Eng., vol. 40, no. 2, pp. 167–191, Feb. 2014, doi: 10.1109/TSE.2013.61.

[51] K. Sudhakar, M. James Stephen, Trummer, B. Faltings, an "Cloud Oriented Integrated Composite Services over SOA in Distributed Computing," International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249-8958 (Online), Volume-10 Issue-3, February 202, pp. 1-7.

[52] S. Sridevi S, G. Karpagam, B. Vinoth, and J. Uma, "Investigation on Blockchain Technology for Web Service Composition: A Case Study" International Journal of Web Services Research Vol. 18, Issue no 1, 2021, pp. 70-93.