International Journal of Enhanced Research in Science, Technology & Engineering
ISSN: 2319-7463, Vol. 13 Issue 10, October-2024, Impact Factor: 8.375

Analysis of Resource Allocation Performance Using MDP and Genetic Algorithm

Malege Sandeep[1], Dr. Rohita Yamaganti[2]

[1]Research Scholar, Department of Computer Science and Engineering, P. K. University, Shivpuri, M.P
[2]Associate Professor, Department of Computer Science and Engineering, P. K. University, Shivpuri, M.P

ABSTRACT

Optimal distribution of resources within the context of known limits and unknown uncertainties is the goal of resource allocation techniques, which pose a significant problem in many fields. To represent dynamic decision-making processes under uncertainty, the MDP framework offers a robust probabilistic model, and the GA searches for optimum or near-optimal solutions by using evolutionary principles. This study presents an approach to dynamically allocating cloud resources for NFV components using Markov Decision Process (MDP) applied to an NP-hard issue. In addition, Bayesian learning method is applied to monitor the historical resource usage in order to predict future resource reliability. Our suggested technique outperforms comparable approaches, according to experimental data..

Keywords: Network Functions Virtualization, Resource Allocation, Markov Decision Process, Bayesian Learning

INTRODUCTION

Numerous fields rely on efficient resource allocation, including manufacturing, healthcare, transportation, cloud computing, and telecommunications. Achieving efficient resource allocation is becoming more challenging as systems get more sophisticated and the need for optimum use increases. Markov decision processes (MDPs) and genetic algorithms (GAs) are two methods that have been extensively used to address resource allocation issues. When it comes to improving speed, saving cost, and making sure resources are distributed fairly, these solutions provide complimentary options. Allocating resources is the practice of systematically distributing limited resources across competing activities in order to accomplish goals. Limitations on time, money, or other resources are common in the actual world that affect this procedure. For example, in cloud computing, it is necessary to dynamically assign resources like CPU, memory, and storage in order to maximize processing efficiency and minimize energy usage. Allocating resources, such as delivery routes or cars, in logistics is also about minimizing operating costs and meeting tight deadlines. The presence of several interacting variables in these situations makes them intrinsically complicated due to the frequent involvement of stochastic settings.

When it comes to modeling decision-making in stochastic contexts, MDP is a strong approach. Finding the best policies for sequential choice issues might be a bit of a challenge, but MDP offers a systematic method based on dynamic programming and probability theory. When the system's state changes over time as a result of the selected actions and probabilistic transitions, it becomes very valuable. The four main components of MDP are states, actions, transition probabilities, and rewards. Using an MDP model of the resource allocation issue, decision-makers may assess the costs and benefits of both short-term and long-term benefits. To decrease delay and adapt to changing traffic circumstances, MDP may be used in network routing, for instance, to find the ideal pathways for data packets. As the number of states and action spaces increases, however, MDP's scalability becomes an increasingly pressing issue. The so-called "curse of dimensionality," in which computing demands grow unmanageable, is a common outcome of large-scale resource allocation challenges. This constraint is often filled by using approximation techniques like value iteration and reinforcement learning. Even with all these improvements, MDP may not be the best choice for situations with a lot of uncertainty or lack of structure since it relies on exact model parameters and transition probabilities.

However, Genetic Algorithms (GA) provide an alternative that is based on heuristics and is motivated by evolutionary and natural selection concepts. When dealing with optimization issues involving big and complicated search spaces, GAs shine where more conventional mathematical approaches fail. Optimal or near-optimal solutions may be found by GAs by repeatedly evolving a population of candidate solutions, which allows them to investigate a broad range of options. To guarantee search adaptability and robustness, GA's key operators—including mutation, selection, and

crossover—imitate biological processes. Among the many uses of GAs in resource allocation, job scheduling, load balancing, and portfolio optimization are just a few examples. By optimizing the deployment of virtual machines in the cloud, GA may minimize energy usage without compromising service quality, for example. For issues involving partial or ambiguous data, GAs are a good alternative to MDP as they don't need explicit modeling of transition probabilities. When applied to contemporary high-performance computing systems, GAs' parallel nature also enables quicker convergence.

REVIEW OF LITERATURE

Peng, Zhihao et al., (2022) Because it assigns jobs to the most suitable resources for execution—be it a grid, a cloud, or a network of peers—task scheduling is a crucial part of every distributed system. High temporal complexity, slower program execution durations, and non-simultaneous processing of input tasks are some of the drawbacks of common scheduling techniques. On heterogeneous distributed computing systems, exploration-based scheduling algorithms take a long time to execute because they need several techniques to prioritize work. Consequently, such systems have bottlenecks when it comes to job prioritizing. It is reasonable to use quicker algorithms to prioritize jobs based on their execution time. One evolutionary strategy to solving complicated problems rapidly is the genetic algorithm (GA). In order to schedule tasks on cloud computing with different priority queues, this study suggests a parallel GA that uses a MapReduce architecture. The primary objective of this research is to find the best way to use MapReduce architecture for cloud computing job scheduling in order to decrease overall execution time. There are two steps to the proposed method's task scheduling process. Initially, the GA was used with heuristic approaches to distribute tasks to processors. Afterwards, the GA was utilized with the MapReduce framework to distribute jobs to processors. Our tests take into account diverse resources with distinct task execution capabilities and the possibility of executing a job on several resources with different durations. Particle swarm optimization, intelligent water drops, moth-flame optimization, and the whale optimization algorithm are among the techniques that the suggested approach surpasses.

Khamayseh, Yaser et al., (2018) On the one hand, wireless networks have limited resources and experience several restrictions, making resource distribution a highly difficult process. In contrast, most resource allocation issues are NP-hard and need a large number of calculations. As a result, practical and efficient answers are badly needed. Fairly distributing the available resources of the network to all active users is the focus of resource allocation concerns. While defining fairness is challenging, this study takes into account the users' and the network operator's (service provider) perspectives on the matter. There are several applications for bio-inspired algorithms, which provide straightforward and efficient answers to complex issues. An efficient method for allocating resources in wireless networks for multimedia is presented in this article by use of a Genetic Algorithm. We use simulation to test how well the suggested approach works. The simulation results demonstrate that the suggested approach outperformed the others.

Ezugwu, Absalom et al., (2016) Managing grid resources effectively to execute the many tasks sent to the grid is crucial to the grid computing system's operational effectiveness. This study delves into a different approach to effectively search, match, and allocate distributed grid resources to tasks in a manner that satisfies the resource need of every grid user job. An approach to resource selection is proposed that utilizes populations based on multisets and is based on the notion of genetic algorithms (GA). Additionally, the article introduces a hybrid GA-based scheduling system that effectively finds the optimal resources for user workloads in a conventional grid computing setting. To improve the GA components' search capabilities in a wide problem space, the suggested resource allocation technique incorporates additional mechanisms, such as populations based on multiset and adaptive matching. The significance of operator improvement on conventional GA is shown via the presentation of empirical data. The suggested addition of a second operator fine-tuning is fast, accurate, and able to handle large work arrival rates, according to the early performance findings.

Porta, Juan (2013) Planning for future land uses is the focus of this research, which employs genetic algorithms. Expert opinion and up-to-date state and federal regulations inform the selection of applicable constraints and optimization targets. This strategy may readily integrate other factors. The shape-regularity of the land use patches that are produced and the appropriateness of the land itself are two optimization criteria that are used. When allocating land, we utilize the existing plots as a baseline. The method may take a long time to execute since there might be a lot of plots that are impacted. Thus, many parallel paradigms, including multi-core parallelism, cluster parallelism, and hybrids of the two, are the subject of the work's implementation and analysis. The results of these experiments demonstrate that genetic algorithms are well-suited to solving land use planning issues.

Goel, Tushar et al., (2003) This research focuses on the GA's potential utility in real-world industrial settings with constrained budgets for simulations. In particular, we look for a way to maximize limited multi-objective optimization while minimizing the overall simulation budget, which includes both the population size and the number of generations. We do a research to find ways to make things better, but we only let 1,000 simulations run. Time savings of a considerable magnitude were achieved by taking advantage of parallelization via the use of concurrent simulations for every GA generation on an HP quad-core cluster. In addition, we looked at how to distribute computing resources efficiently to get the most speed boost. We used a finite element model with 58,000 components to simulate the

crashworthiness of an automobile, in addition to two analytical cases. With a cap of 1,000 simulations, we tested out different population sizes and generations. Monte-Carlo and space filling sampling techniques were used to compare the optimization performance. It was discovered that the GA could find numerous workable solutions with trade-offs. To get excellent trade-off solutions, it is shown that a high number of generations is desirable. There were noticeable gains in efficiency with the vehicle's layout. Additionally, this example demonstrates that a large number of viable solutions may be obtained in the first few generations using about 200 simulations, even for problems with a tiny feasible area.

Dai, Y. et al., (2003) One of the most pressing problems in software testing is determining how to best use the available testing resources to provide the highest level of dependability. Models created under the premise of simple series or parallel modules are often used in the many papers on this problem. The optimization issue becomes more difficult to solve as system configurations get more complicated. When dealing with complicated software systems structures and various goals, this study introduces an evolutionary approach for testing-resource allocation challenges. When we allocate testing resources, we think about system dependability and testing costs simultaneously. Putting the strategy into action is a breeze. We present some numerical examples to demonstrate how the technique may be used.

EXPERIMENTAL SETUP

We have devised a battery of tests to evaluate the efficiency. The RHEL 6.5 operating system was used in all trials on Dell PowerEdge Servers R720 and R810. Our network offers a speed of 100 Mbps. In order to test the NFV components, we used WorkflowSim. Researchers may use WorkflowSim, a process simulator, to test out different methods for optimizing workflows. For this reason, we simulated varying VM and NFV component sizes using WorkflowSim.

Maximizing the predicted total reward over a limited horizon is one of the criteria we use to evaluate MDP. We calculate the best finite-horizon policy given an MDP and a horizon H. Horizon H= K and H=1 were the two cases we tested. Whereas H=1 simply takes into account one state, H=K signifies that we examine k-1 states till the end. We use the abbreviations MDPk for K-horizon and MDP1 for 1-horizon MDP here. A popular scheduling method is the Genetic Algorithm (GA), which we have compared to our suggested algorithms, MDPk and MDP1.

Our experiment's GA configuration looks like this: Random seed, one-third mutation rate, one-point cross-over recombination, one-wheel selection as parents, twenty-odd generations, one-hundredth of a population size, and uniform mutation method.

Two measures, execution cost and time restriction, are used to assess the allocation strategies. The first one shows the time cost divided by the allocation, while the second one shows the cost to complete the NFV components.

We devised studies to investigate how well our suggested MDP performed under varied settings, such as with variable numbers of components, virtual machines, and due dates. We divided the whole time cost into two parts so that we could examine it more thoroughly. The time cost of initializing all conceivable states of MDP is recorded by one step that is supplied as an initialization stage. The second stage is a solution stage that keeps track of how much time it took to determine the best allocation plan.

RESULTS AND DISCUSSION

Here are the experimental findings with different numbers of NFV component jobs and a fixed number of virtual machines (VMs), as shown in Figures 1 and 2. Figure 2 shows the overall running time of the solution stage and Figure 1 shows the entire running time of the approach. By comparing the solution stage running time cost of other approaches and taking execution time into account, we find that our MDPk method performs better. Nevertheless, the approach calculates all potential allocation solutions after traversing all possible states in the startup step. It is evident from comparing Fig. 1 with Fig. 2 that the initialization step takes much more time than the solution stage. One important consideration in dynamic resource allocation is the time cost of solutions. Given that evaluating a single node takes a constant amount of time, O(1), the execution time for MDPk is O(t(v+1)), and for MDP1 it is O(t × v), where t is the number of NFV component tasks and v is the number of virtual machines..

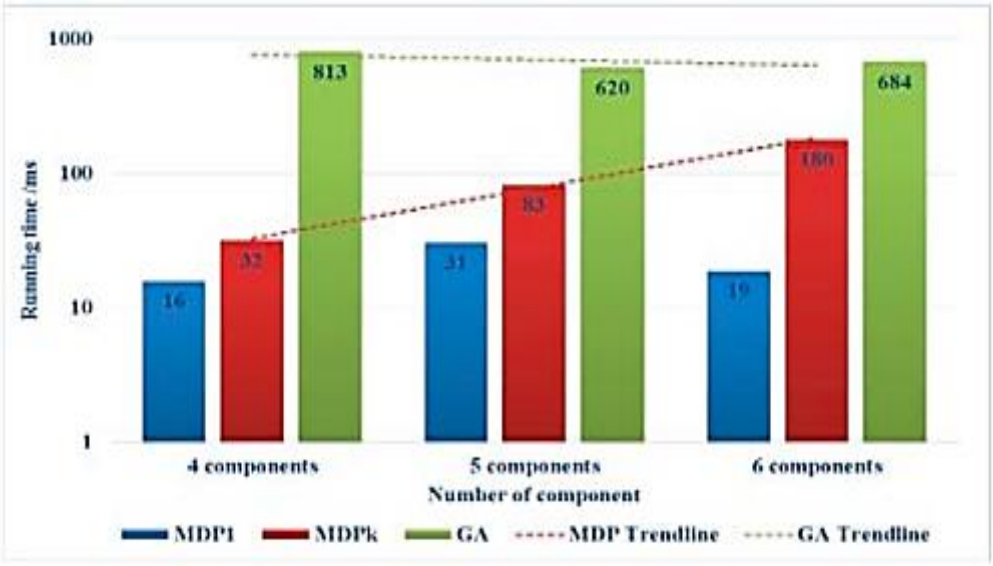

Figure 1: Total running time with different number of NFV Component

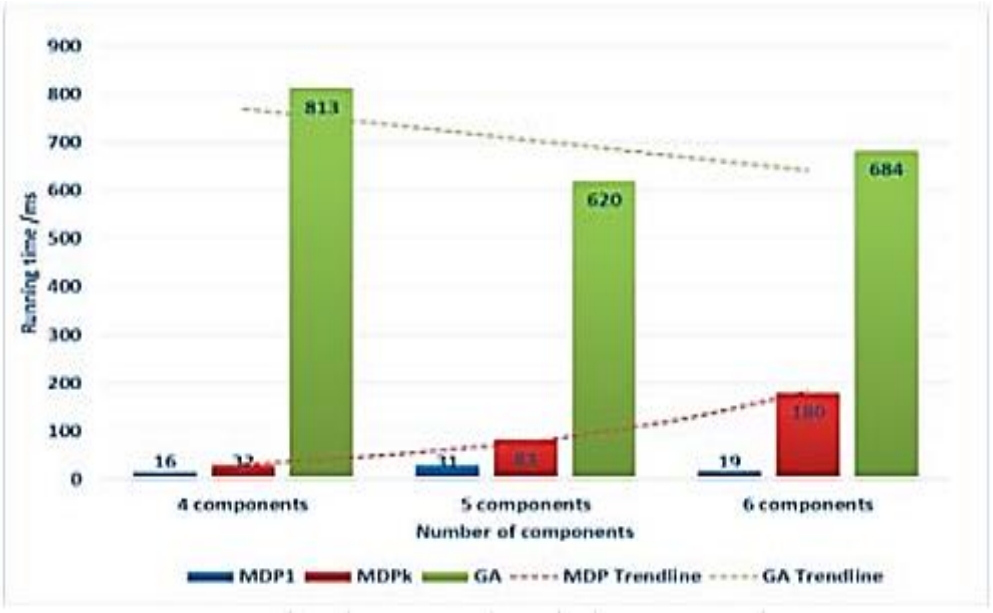

Figure 2: Solution stage running time with different number of NFV Component

Experimental findings for recording method running time while establishing a constant number of NFV component jobs and modifying the numbers of virtual machines (VMs) are shown in Figure 3. Although GA has a higher running time than other approaches, it is impossible to predict how long it will take since the GA program continues to run until either stopped or interrupted at a certain point in time. Meanwhile, GA isn't able to provide the best possible outcome on a global scale. Whenever the random seed is not consistent, the outcomes will vary. When all parameters, including the number of people and the number of generations, crossover rate, and mutation rate, remain constant, GA may provide a better allocation plan at a lower time cost, particularly for complex NFV processes and VM circumstances.

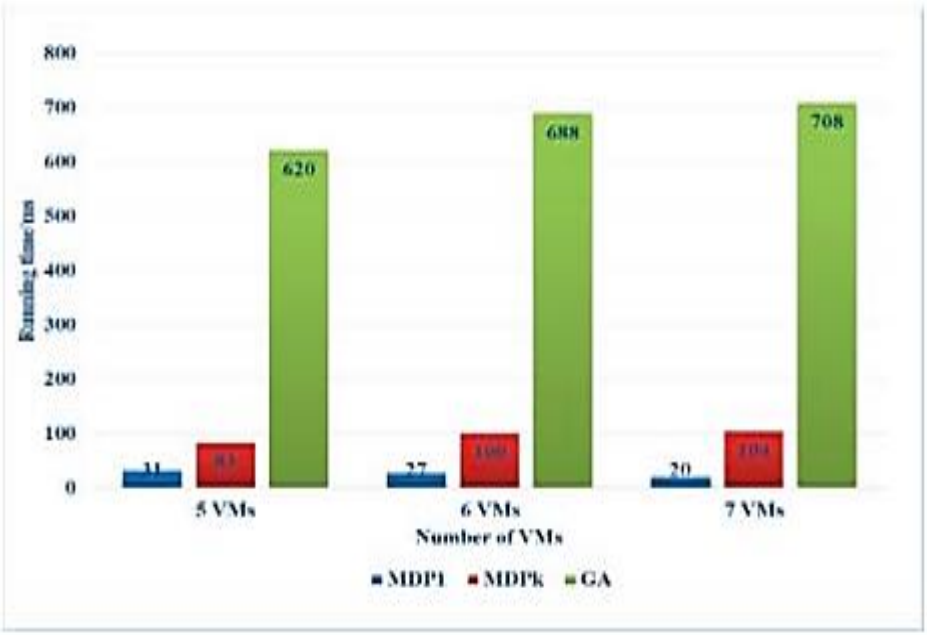

Figure 3: Running time with different number of VMs

With a considerable rise in the number of components, MDPk and MDP1 take up more time than GA on the total running time trend line, with the majority of that time being spent during startup. When compared to GA, the time required for the solution stage of MDPk and MDP1 is much lower. One explanation is that compared to GA, MDP is more likely to do floating point calculations. GA lacks expected value computation, in contrast to MDP.

A constant number of virtual machines (VMs) and a variable number of NFV component jobs are used to illustrate our experimental findings (Fig. 4). Figure 5 displays the outcomes of our experiments when the number of VMs is varied but the number of NFV component jobs remains constant. When changing NFV component tasks or virtual machines (VMs), MDPk might achieve the global optimum. The GA result is comparable to the MDPk result. In theory, the K-horizon MDP can discover the optimal solution in any feasible state. Some allocation plans will be excluded when the deadline component is taken into account, however, since the deadline partition is backward-designed while the technique to determine the total cost is forward-looking.

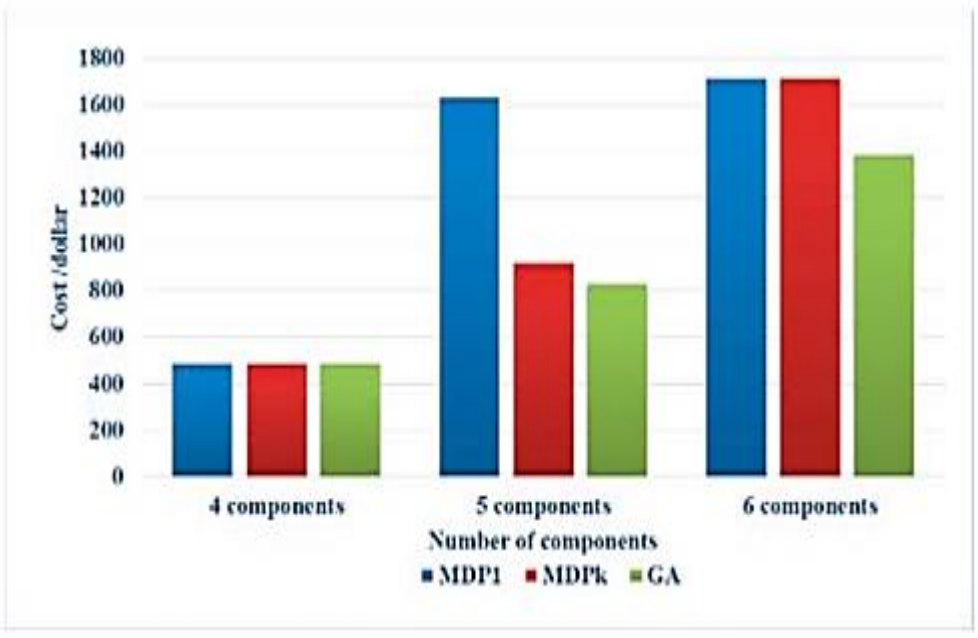

Figure 4: Cost with different number of NFV Component

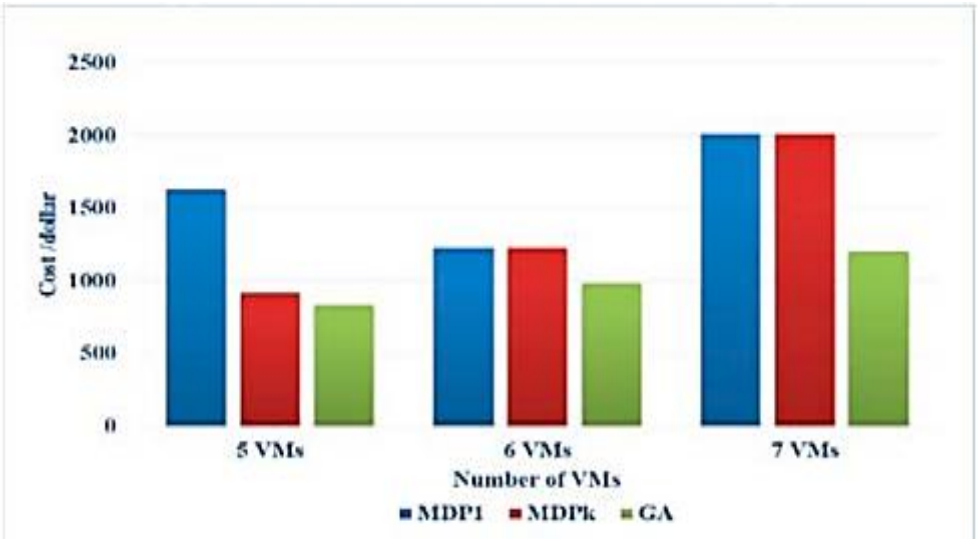

Figure 5: Cost with different number of VMS

To further investigate the potential impact of the deadline on the outcome, we devised an experiment with several due dates for recording the total cost. Figure 6 displays the outcomes. Taking the deadline into account, the GA approach may sometimes provide a more optimal allocation plan. Because it skips over one step and picks the best node in that stage, MDP1 performs the poorest. What this means is that it just reflects local optimization in the near term.

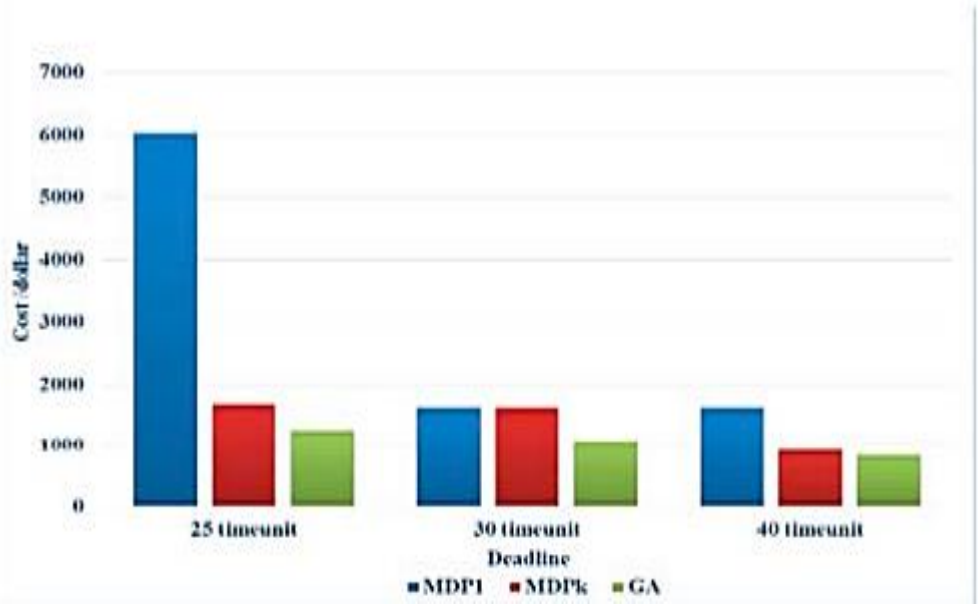

Figure 6: Cost with different deadlines

For this reason, MDP1 should not be used first in cases when time is of the essence. Because MDP1 can't predict the whole cost and the cost looks to be cheaper for certain selections at some phases, it can't strike a balance between the short and long term. As the deadline draws nearer, MDP1 must decide whether to invest more resources to save time and succeed or fail.

CONCLUSION

When comparing MDPk, MDP1, and Genetic Algorithm (GA), it becomes clear that MDPk achieves global optimization and has the fastest solution stage running time. This is particularly true when dealing with different NFV component jobs, virtual machines, and deadlines. The exhaustive state traversal in MDPk's initialization step increases the time cost, but the solution stage is more efficient and reliable than GA, which has inconsistent results and can't ensure global optimality. Although it would take a relatively long period, our MDPk may achieve the global ideal. When considering the time cost, GA is the best way for solving the problem. But GA won't always come up with the optimal answer. When faced with a tight timeline, GA can still identify a better answer. Only when dealing with somewhat unique challenges may MDP1 save time

REFERENCES

[1]. H. Jafari Kaleybar, M. Davoodi, M. Brenna, and D. Zaninelli, "Applications of Genetic Algorithm and Its Variants in Rail Vehicle Systems: A Bibliometric Analysis and Comprehensive Review," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2023, doi: 10.1109/ACCESS.2023.3292790.

[2]. Z. Peng, P. Pirozmand, M. Motevalli, and A. Esmaeili, "Genetic Algorithm-Based Task Scheduling in Cloud Computing Using MapReduce Framework," *Mathematical Problems in Engineering*, vol. 2022, no. 4, pp. 1–11, 2022, doi: 10.1155/2022/4290382.

[3]. Z. Peng, P. Pirozmand, M. Motevalli, and A. Esmaeili, "Genetic Algorithm-Based Task Scheduling in Cloud Computing Using MapReduce Framework," *Mathematical Problems in Engineering*, vol. 2022, no. 4, pp. 1–11, 2022, doi: 10.1155/2022/4290382.

[4]. S. Wang, Y. Wu, and R. Li, "An Improved Genetic Algorithm for Location Allocation Problem with Grey Theory in Public Health Emergencies," *International Journal of Environmental Research and Public Health*, vol. 19, no. 15, p. 9752, 2022, doi: 10.3390/ijerph19159752.

[5]. S. Jomthanachai, W. Rattanamanee, R. Sinthavalai, and W.-P. Wong, "The application of genetic algorithm and data analytics for total resource management at the firm level," *Resources, Conservation and Recycling*, vol. 161, p. 104985, 2020, doi: 10.1016/j.resconrec.2020.104985.

[6]. R. P. M. and D. M., "Efficient task allocation approach using genetic algorithm for cloud environment," *Cluster Computing*, vol. 22, no. 21, 2019, doi: 10.1007/s10586-019-02909-1.

[7]. Y. Khamayseh and R. Al-qudah, "Resource Allocation using Genetic Algorithm in Multimedia Wireless Networks," *International Journal of Communication Networks and Information Security*, vol. 10, no. 2, pp. 419–424, 2018, doi: 10.17762/ijcnis.v10i2.3280.

[8]. W. Rankothge, F. Le, A. Russo, and J. Lobo, "Optimizing Resource Allocation for Virtualized Network Functions in a Cloud Center Using Genetic Algorithms," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 1–1, 2017, doi: 10.1109/TNSM.2017.2686979.

[9]. A. Ezugwu, N. Okoroafor, S. Buhari, M. Frincu, and S. Junaidu, "Grid Resource Allocation with Genetic Algorithm Using Population Based on Multisets," *Journal of Intelligent Systems*, vol. 26, no. 1, pp. 169–184, 2016, doi: 10.1515/jisys-2015-0089.

[10]. M. Zarra-Nezhad, R. Yousefi Hajiabad, and M. Fakhernia, "Evaluation of Particle Swarm Algorithm and Genetic Algorithms Performance at Optimal Portfolio Selection," *Asian Economic and Financial Review*, vol. 5, pp. 88–101, 2015.

[11]. J. Porta, F. Parapar, F. Rivera, I. Santé, and I. Crecente, "High performance genetic algorithm for land use planning," *Computers, Environment and Urban Systems*, vol. 37, no. 1, pp. 45–58, 2013, doi: 10.1016/j.compenvurbsys.2012.05.003.

[12]. A. Haruna, N. Abba, N. Zakaria, and N. Haron, "Grid Resource Allocation: A Review," *Research Journal of Information Technology*, vol. 4, no. 2, pp. 38–55, 2012.

[13]. J. Gu, J. Hu, T. Zhao, and G. Sun, "A New Resource Scheduling Strategy Based on Genetic Algorithm in Cloud Computing Environment," *Journal of Computers*, vol. 7, no. 1, pp. 42–52, 2012, doi: 10.4304/jcp.7.1.42-52.

[14]. T. Goel, N. Stander, and Y.-Y. Lin, "Efficient resource allocation for genetic algorithm based multi-objective optimization with 1,000 simulations," *Structural and Multi-Disciplinary Optimization*, vol. 41, no. 3, pp. 421–432, 2010, doi: 10.1007/s00158-009-0426-9.

[15]. Y. Dai, M. Xie, K. Poh, and B. Yang, "Optimal testing-resource allocation with genetic algorithm for modular software systems," *Journal of Systems and Software*, vol. 66, no. 1, pp. 47–55, 2003, doi: 10.1016/S0164-1212(02)00062-6.

[16]. M. Louati, S. Benabdallah, F. Lebdi, and D. Milutin, "Application of a Genetic Algorithm for the Optimization of a Complex Reservoir System in Tunisia," *Water Resources Management*, vol. 25, no. 10, pp. 2387–2404, 2011, doi: 10.1007/s11269-011-9814-1.

[17]. J. Alcaraz and C. Maroto, "A Robust Genetic Algorithm for Resource Allocation in Project Scheduling," *Annals of Operations Research*, vol. 102, no. 1, pp. 83–109, 2001, doi: 10.1023/A:1010949931021.