Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Multi-Objective Optimization for Optimal VM Allocation using Ali Baba & the Forty Thieves Algorithm

Utpal Chandra De[1], Rabinarayan Satapathy[2], Sudhansu Shekhar Patra[1*], Bibhuti Bhusan Dash[1]

[1]School of Computer Applications, KIIT Deemed to be University, Bhubaneswar, India
[2]Faculty of Engineering and Technology, Sri Sri University, Cuttack, India
Email: sudhanshupatra@gmail.com

Utilizing cloud computing to harness its advantages by optimizing different parameters to accommodate evolving needs is a formidable challenge. The efficient allocation of tasks to virtual machines (VMs) and VMs to physical machines (PMs), referred to as VM placement, is essential for enhancing energy efficiency and resource utilization. In earlier works, the allocation of VMs into PMs has been studied through various nature-inspired algorithms, decision-based algorithms, ML-driven algorithms, etc. Over the years, new algorithms are constantly being developed to enhance the quality of the optimization task, whether the objective is single or multi-objective. This paper compared Ali Baba & the Forty Thieves (AFT) algorithm with previously implemented techniques for multi-objective optimization for VM placement and allocation.

Keywords: VM Allocation, Optimization, Cloud Computing, Ali Baba & the Forty Thieves Algorithm, Optimal Placement.

1. Introduction

Appropriate allocation of Virtual Machines (VMs) in a cloud server is not a new challenge today. Numerous researchers have worked towards creating an optimal allocation strategy where the intention has been to maximize the utilization of resources and cost savings with minimum power consumption. These strategies can be exact algorithms such as mathematically modeled or decision-based algorithms. They eventually turned out to be ineffective when the number of constraints increased over time. The objectives of higher dimensions could not be solved using these systems of optimization. Hence, people started moving towards approximate algorithms and today most of the work for multi-objective optimizations (MOO) is being carried out using these approximate algorithms, where we try to find a near-optimal solution within a reasonable period. Fig.1 shows an allocation of VM allocation to the PMs where there are 5 PMs $<s_1, s_2, s_3, s_4, s_5>$ where s_1 allocates $<VM_1, VM_2,$

VM_3, VM_7>, s_2 allocates <VM_4, VM_5, VM_6, VM_9> , s_5 allocates <VM_8, VM_{10}, VM_{11}, VM_{12}>, s_3 and s_4 are unused PMs (Patra et al., 2022)

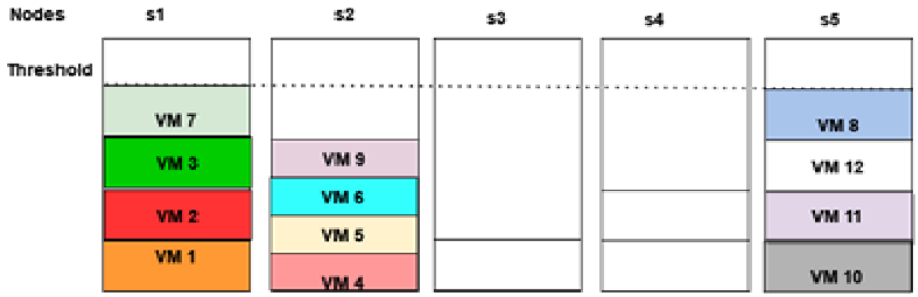

Fig. 1. Allocation of virtual machines (VMs) into Physical Machines(PMs)

Authors (De et al.,2023) have created a system that employs predictive analytics to predict the future requirements of resources and allocate the VMs based on ML techniques. In (Madhumala et al., 2019) the authors surveyed various nature-inspired algorithms for VM allocation using Ant Colony Optimization, Particle Swarm Optimization, Genetic Algorithm, Flower Pollination Algorithm, Fruit Fly Algorithm, and Honey Bee Algorithm respectively, however, there are no quantitative results discussed in this work. Authors (Mann et al., 2017) have provided us with an environment and methodology for comparison of VM allocation algorithms and have compared 7 different algorithms using the proposed scheme. Whale optimization GA has been used by the authors (Saxena et al., 2021) to carry out the VM Placement task. In (Ma et al., 2022) the authors have provided us with an exhaustive list of 511 nature-inspired algorithms from which we can have an insight into how often an algorithm is used, based on different contexts.

Authors (Li et al., 2020) have used particle swarm optimization (PSO) for adaptive management and MOO of VM in cloud computing. Authors (Braiki et al., 2018) have also used the PSO technique giving us more optimized results for the number of PMs, their balanced utilization, and energy consumption. A hybrid model of both PSO and Flower Pollination Optimization (FPO) has been leveraged by the authors (Mejahed et al., 2022) to address the same VM allocation task. PSO has been compared with Genetic Algorithms (GA) (Choudhary et al., 2022) where the authors concluded PSO to be superior to GA in VM placement. Using logistic mapping sequences in the solution, an updated version of PSO named Chaos PSO has been used (Xu et al., 2018) to address the multi-objective optimization problem.

Having discussed the drawbacks of exact algorithms and the benefits of approximate ones as their alternative, the approximate algorithms can be classified into heuristic and meta-heuristic algorithms. Heuristic algorithms are designed and used for specific purposes and hence by nature are rigid when it comes to application in other domains. On the other hand, meta-heuristic algorithms are generic ones that are by nature flexible and designed to be used in any given context. Most of the modern nature-inspired algorithms are considered to be meta-heuristic algorithms (Askari et al., 2020). Now based on the number of solutions at each

iteration, these generic algorithms can be further classified into individualistic or single solution-based meta-heuristics such as integer programming (Glover, 1986) , Guided Local Search (Voudouris et al., 2010) etc., and collective or population solution-based meta-heuristics such as genetic algorithms (Holland, 1075) differential evolution algorithm (Storn et al., 1997), etc. The population-based- metaheuristics can be further classified into physics/chemistry-based algorithms, evolutionary algorithms, swarm-based algorithms, etc. These swarm-based algorithms can then be further classified into human-related and non-human-related algorithms. This classification can go on endlessly based on certain criteria, but that's not what we are concerned about over here.

In this work, we have discussed one such meta-heuristic optimization algorithm – the Ali Baba & the Forty Thieves (AFT) (Braik et al., 2022) and compared that with previously

implemented algorithms in the context of multi-objective optimization for the optimal placement of VMs.

The format of article is organized in this manner. Section 1 serves as an introduction; Section 2 provides Ali Baba & The Forty Thieves algorithm. The mathematical modeling is covered in Section 3. The Multi-objective problem formulation and the AFT and multi-objective optimization is presented in Section 4 and section 5 respectively. The results and discussion is presented in section 6. Finally section 7 concludes with future research.

2. Ali Baba & The Forty Thieves Algorithm (AFT Algorithm)

This algorithm was initially developed for addressing single objective optimization tasks. Inspired by the tale of Ali Baba and the Forty Thieves (Braik et al., 2022) , this aims to find the position of Ali Baba inside his cave. This search is iteratively performed by a group of 40 Thieves, where the solution found in the i^{th} iteration is reinforced after the $i+1^{th}$ iteration. This reinforcement is carried out based on the collective actions of the thieves (also referred to as the population). A counter agent called Morgiana tries to limit this population from finding Ali Baba. The search space corresponds to the place where Ali Baba resides.

2.1. The Tale

As per the tale, the assistant leader of the population disguises himself as a tourist and somehow manages to identify the house of Ali Baba and marks his door with a cross sign. As a counter to this, Morgiana marks the door of all the neighboring houses with similar crosses. Similar successive attempts were made to identify the house of Ali Baba and kill him to acquire back the treasure (Mansour, 2008). This analogy demonstrates the exploration and exploitation aspects of the work in consideration, in which attempts are being made to enhance the previously discovered solutions.

In the final attempt, the chief of the population himself visits and identifies the house and observes the door of the house in detail so that he won't miss identifying it next time. Having done this, he buys 40 empty barrels along with 40 mules and visits Ali Baba during the night as an oil trader seeking shelter. He fills one barrel with oil and the rest of the barrels are occupied by the thieves and put behind his house. On observing a finger ring, similar to the ones in treasure in the hands of the trader she secretly investigates the barrels and finds out the

thieves inside them. This leads her to empty the oil barrel, heat the oil, and pour it onto the other barrels consisting of thieves to kill them and counter this attempt as well. This attempt signifies persistence as a trait for an efficient search algorithm.

2.2. AFT Algorithm

Imitating the tale described in the previous subsection, the aim is to mathematically model human actions into computation tasks based on the following assumptions:

1. n thieves work in a group to determine the position of the residence of Ali Baba, where n = 40.

2. An initial distance needs to be covered by these n thieves, to locate the house of Ali Baba.

3. An agent, Morgiana acts as an agent to divert the thieves and prevent them from reaching Ali Baba.

The meta-heuristic algorithm is thus framed by mapping the behavior of the thieves and Morgiana to an objective function, which needs to be optimized in our case.

3. Mathematical Modelling

The concept and ideas discussed in the previous section start with an initialization similar to that of other optimization techniques, followed by iterations and updations.

3.1. Initialization

The position matrix t, of n thieves is randomly initialized in d-dimensional space.

$$T = \begin{bmatrix} T_1^1 \; T_2^1 \; ....T_d^1 \\ T_1^1 \; T_2^1 \; ....T_d^1 \\ \quad \text{M O} \quad \text{M} \\ T_1^n \; T_2^n \; ....T_d^n \end{bmatrix} \tag{1}$$

The initial position of thieves is generated as

$$T^i = l_j + r(u_j - l_j) \tag{2}$$

Where,

- t^i is the position of the i^{th} thief.
- l_j & u_j are lower & upper bounds.
- r is a random number that ranges from 0 to 1.

The wit level matrix of Morgiana - m is initialized as

$$m = \begin{bmatrix} m_1^1 \; m_2^1 \; ....m_d^1 \\ m_1^1 \; m_2^1 \; ....m_d^1 \\ \text{M O M} \\ m_1^n \; m_2^n \; ....m_d^n \end{bmatrix} \tag{3}$$

3.2. Fitness Function

The fitness function for each thief is defined as

$$f = \begin{bmatrix} f_1(T_1^1 \; T_2^1 \; ....T_d^1) \\ f_2(T_1^1 \; T_2^1 \; ....T_d^1) \\ \text{M O M} \\ f_n(T_1^n \; T_2^n \; ....T_d^n) \end{bmatrix} \tag{4}$$

While simulating this algorithm, the quality of each solution corresponding to each thief is evaluated and updated, if the current solution is better than the previous one.

3.3. Cases

Three cases may fundamentally occur during the search process. We need to consider two factors for this, first is the movement of thieves and second is the diversion created by Morgiana.

1. Case 1: Ali Baba is successfully tracked down by the thieves. The updated locations can be expressed as,

$$T_{t+1}^i = g_t^{best} + \left[Td_t(best_t^i - y_t^i)r_1 + Td_t(y_t^i - m_t^{a(i)})r_2 \right] \text{sgn}(rand - \frac{1}{2}) \tag{5}$$

Where,

- g^{best} represents the current best position.
- Td is the tracking distance.
- Pp is the perception potential of the thieves.
- $m^{a(i)}$ is the Morgiana's intelligence.
- $a = \lceil (n-1)rand(n,1) \rceil$
- $r_1 \geq 0.5$
- $r_2 > Pp_t$

$$m_t^{a(i)} = \begin{cases} T_t^i, & if\ f(T_t^i) \geq f(m_t^{a(i)}) \\ m_t^{a(i)}, & if\ f(T_t^i) < f(m_t^{a(i)}) \end{cases} \quad (6)$$

$$Td_t = \alpha_0 e^{-\alpha_1 (t/t_{\max})^{\alpha_1}} \quad (7)$$

$$Pp_t = \beta_0 \log(\beta_1 (t/t_{\max})^{\beta_0}) \quad (8)$$

Where,

- $\alpha_0 = 1$
- $\beta_0 = 0.1$
- $\alpha_1 = \beta_1 = 2$

2. Case 2: In this case, the thieves may come to know that they got deceived and start searching randomly in the search space. The updated locations can be expressed as,

$$T_{t+1}^i = Td_t \left[(u_j - l_i) rand + l_i \right] \quad (9)$$

3. Case 3: Considers search in other positions than those obtained using Eqn. 5 from Case 1, just that $r_1 < 0.5$ & $r_2 \leq$ Pp.

$$T_{t+1}^i = g_t^{best} - \left[Td_t (best_t^i - y_t^i) r_1 + Td_t (y_t^i - m_t^{a(i)}) r_2 \right] \text{sgn}(rand - \frac{1}{2}) \quad (10)$$

The pseudo-code for AFT algorithm is shown below:

ALGO: AFT Algorithm

01: u_j & l_j are the upper and lower bounds of the search space in the j^{th} dimension

02: Randomly initialize the position, T, of all n thieves in the search space

03: Initialize the best position ($best^i_t$) and global best position (g^{best}_t) for the thieves.

04: Initialize the intelligence degree of Morgiana with respect to all thieves

05: Evaluate the position of all thieves using a fitness function (F(T))

06: Set t ← 1

07: while ($i < t_{max}$) do

08: $Td_t = 1.0 \times e^{-2.0(t/t_{max})^{2.0}}$

09: $Pp_t = 0.1 \times \log(2.0(t/t_{max})^{0.1})$

10: for i = 1, 2, ..., n do

11: if (rand ≥ 0.5) then

12: if (rand > Pp_t) then
13: Use Eqn. 5
14: else
15: Use Eqn. 9
16: end if
17: else
18: Use Eqn. 10
19: end if
20: end for
21: for i = 1, 2, ..., n do
22: Check the feasibility of new positions
23: Evaluate & update thieves' new position
24: Update the solutions $best^i_t$ and g^{best}_t
25: Update $m_t^{a(i)}$ using Eqn. 6
26: end for
27: t = t + l
28: end while
29. END ALGO

4. MOO Problem Formulation

We intend to minimize:

1. Placement Time for the j^{th} VM (T_j)
2. Power Consumption (P_{Total})
3. Resource Wastage (W_{Total})

For n PMs and m VMs in consideration [8],

$$T_j = \sum_{i=0}^{n} t_j e_{ij} \tag{11}$$

$$P_{Total} = \sum_{i=0}^{n} x_i \times P_i^{'} \tag{12}$$

$$W_{Total} = \sum_{i=1}^{n} x_i \frac{\left| (u_i^{CPU} - \sum_{j=1}^{m} e_{i,j} c_j) - (u_i^{RAM} - \sum_{j=1}^{m} e_{i,j} m_j) \right| + \varepsilon}{\sum_{j=1}^{m} e_{i,j} c_j + \sum_{j=1}^{m} e_{i,j} m_j} \quad (13)$$

5. AFT in Multi-Objective Optimization

This is an enhanced version of the AFT algorithm. In case of Multi Objective optimizations, the newly updated state doesn't always dominate the previous state. Hence, to diversify the population an addition has been made to it [18]. In cases where the new member doesn't member doesn't have any significant impact over the other, the new member is replaced with an old one with a certain probability. This ratio is called the randomness factor (R_f) expressed as,

$$Rf_t = \theta_0 e^{-\theta_1 (t/t_{\max})\theta_1} \quad (14)$$

Where,

- θ_0 is the initial value.
- θ_1 is a constant affecting the amount of change.
- $\alpha_0 = 1.5$, $\alpha_1 = 1$
- $\beta_0 = 0.2$, $\beta_1 = 2.72$
- $\theta_0 = 0.35$, $\theta_1 = 2$

6. Results & Discussion

The algorithm was run for 150 iterations. During the initial iterations a sudden decrease in the F(T) values was observed, followed by which the decrease in values got stable. After 125 iterations we observed an increase in the F(T) value, where it reached its minimum. Due to this, we used parameter configurations corresponding to the 125th iteration. The line plot in Fig. 2 shows the fitness function values with an increasing number of iterations.

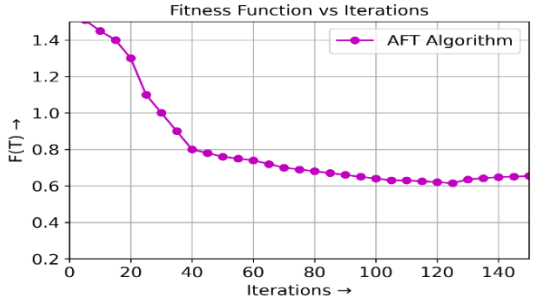

Fig. 2. F(T) vs Iterations

Compared to previously implemented algorithms, we see a decrease in the value of F(T) = 0.615 after 125 iterations. Fig. 3 shows a comparison of TSO with FPO, HPSOLF-FPO and PSOLF.

Fig. 4 shows a comparison between the average placement times (in secs) for 2000 requested VMs. Fig. 5 shows a comparison between power consumed (in kWh) for around 2100 requested VMs. We see a decrease in the values when implemented using the Multi Objective Ali Baba & the Forty Thieves Algorithm.

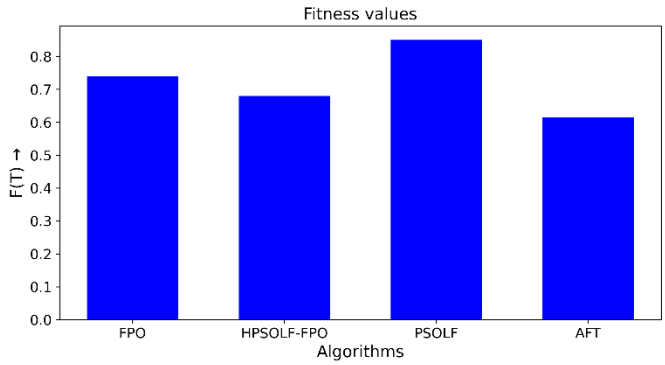

Fig. 3. Comparison of Fitness Values

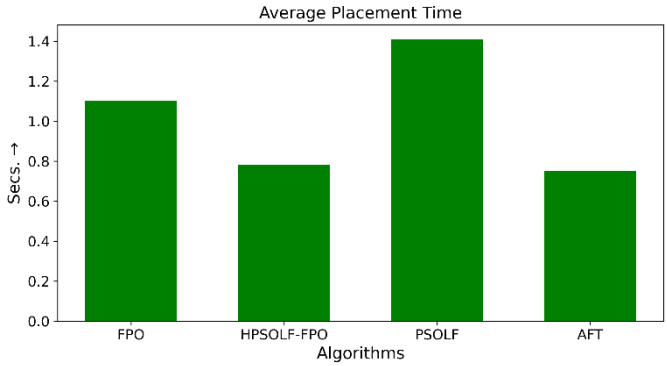

Fig. 4. Comparison of Placement Times

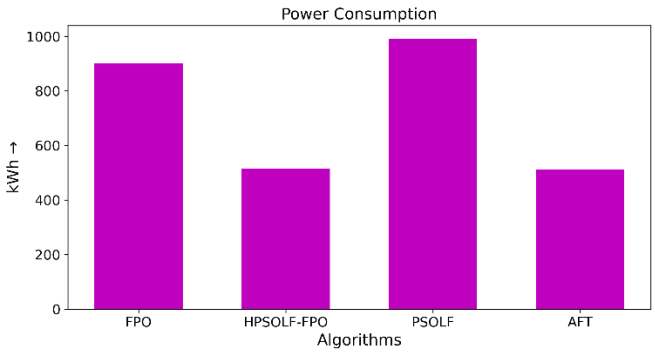

Fig. 5. Comparison of Power Consumption

7. Conclusion and Future Work

After implementing the Ali Baba & the Forty Thieves Optimization Algorithm over multiple objectives namely placement time, power consumption and resource wastage we found out that it gave us minimized values compared to previously implemented techniques. AFT gave us a placement time of 0.75 seconds for 20000 requested VMs. It also gave a minimized value of power consumption of around 512 kWh for 2100 requested VMs. Even though 25 extra iterations were needed compared to previously implemented algorithms, to find the minima, the results acquired are comparatively better.

New nature inspired algorithms are being developed every year, which when put to use may turn out to be beneficial over the other ones. We shall implement other latest algorithms and compare their efficacies amongst themselves in our future works with an intention to improve the overall efficiency of the system and minimize the losses.

References

1. Askari, Q., Younas, I., & Saeed, M. (2020). Political Optimizer: A novel socio-inspired meta-heuristic for global optimization. Knowledge-based systems, 195, 105709.
2. Braik, M., Ryalat, M. H., & Al-Zoubi, H. (2022). A novel meta-heuristic algorithm for solving numerical optimization problems: Ali Baba and the forty thieves. Neural Computing and Applications, 34(1), 409-455.
3. Braiki, K., & Youssef, H. (2018). Multi-objective virtual machine placement algorithm based on particle swarm optimization. In 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC) (pp. 279-284). IEEE.
4. Choudhary, R., & Perinpanayagam, S. (2022). Applications of virtual machine using multi-objective optimization scheduling algorithm for improving CPU utilization and energy efficiency in cloud computing. Energies, 15(23), 9164.
5. De, U. C., Satapathy, R., & Patra, S. S. (2023). Optimizing Resource Allocation using Proactive Predictive Analytics and ML-Driven Dynamic VM Placement. In 2023 4th IEEE Global Conference for Advancement in Technology (GCAT) (pp. 1-5). IEEE.
6. Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. Computers & operations research, 13(5), 533-549.
7. Gül, B. K., & Taşpınar, N. (2023). A New Multi-objective Ali Baba and the Forty Thieves Optimization Algorithm. In 2023 58th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST) (pp. 127-130). IEEE.
8. Holland, J. (1975). Adaptation in natural and artificial systems, univ. of mich. press. Ann Arbor, 7, 390-401.
9. Li, S., & Pan, X. (2020). Adaptive management and multi-objective optimization of virtual machine in cloud computing based on particle swarm optimization. EURASIP Journal on Wireless Communications and Networking, 2020(1), 102.
10. Ma, Z., Wu, G., Suganthan, P. N., Song, A., & Luo, Q. (2022). Performance assessment and exhaustive listing of 500+ nature inspired metaheuristic algorithms. ArXiv. /abs/2212.09479
11. Madhumala, R. B., & Tiwari, H. (2019). Virtual Machine Optimization using Nature Inspired Algorithms. International Journal for Research in Applied Science and Engineering Technology (IJRASET), 2321-9653.
12. Mann, Zoltan & Szabó, Máté. (2017). Which is the best algorithm for virtual machine placement optimization?. Concurrency and Computation: Practice and Experience. 29. e4083. 10.1002/cpe.4083.

13. Mansour, W. (2008). 'Ali Baba and the Forty Thieves': an allusion to Abbasid organised crime. Global Crime, 9(1-2), 8-19.
14. Mejahed, S., & Elshrkawey, M. (2022). A multi-objective algorithm for virtual machine placement in cloud environments using a hybrid of particle swarm optimization and flower pollination optimization. PeerJ Computer Science, 8, e834.
15. Patra, S. S., Govindaraj, R., Chowdhury, S., Shah, M. A., Patro, R., & Rout, S. (2022). Energy Efficient End Device Aware Solution Through SDN in Edge-Cloud Platform. IEEE Access, 10, 115192-115204.
16. Saxena, D., Gupta, I., Kumar, J., Singh, A. K., & Wen, X. (2021). A secure and multiobjective virtual machine placement framework for cloud data center. IEEE Systems Journal, 16(2), 3163-3174.
17. Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11, 341-359.
18. Voudouris, C., Tsang, E. P., & Alsheddy, A. (2010). Guided local search. In Handbook of metaheuristics (pp. 321-361). Springer, Boston, MA.
19. Xu, L., & Pan, D. (2018). Multi-objective optimization based on chaotic particle swarm optimization. International Journal of Machine Learning and Computing, 8(3), 229-235.