Artificial Intelligence for a Lower Energy Consumption in Cloud Computing

Ettore Tancredi Galante
Department of Computer Science
University of Milan
Milan, Italy
ettoretancredi.galante@studenti.unimi.it

March 5, 2023

Contents

1 Abstract

Ettore: Placeholder for the abstract.

2 Introduction

Ettore: Introduction of the thesis. Prelude to the problem.

The widespread adoption of *cloud computing* services in the last few years has led to a significant increase in the energy consumption of *data centers*, which has become a critical concern for cloud providers. While cloud computing offers many benefits, such as cost savings and scalability, the energy consumption associated with it has negative implications both from an environmental and financial perspective. As the demand for cloud services continues to grow, the energy consumption of data centers is expected to increase exponentially, further exacerbating the issue.

The optimization of resource allocation in *cloud computing environments* is a complex problem that involves a large number of variables and constraints. The main objective is to minimize energy consumption while maintaining or improving the performance of the services. Achieving this goal is challenging due to the scale of cloud computing infrastructures, the dynamic nature of workloads, and the need to provide 24/7 availability. Given these challenges, cloud providers face the dilemma of how to maintain high-quality service delivery while reducing energy consumption. [**?**]

Artificial Intelligence (AI) techniques, such as *neural networks* and *genetic algorithms* [**?**, **?**, **?**, **?**], have been widely studied as a method for addressing this problem.

Genetic algorithms are a type of optimization algorithm that mimic the process of natural evolution and use heuristic techniques to find near-optimal solutions for complex problems. They are well suited for solving problems that involve a large number of variables and constraints, such as the resource allocation problem in cloud computing. They can explore a large search space in a relatively short amount of time, enabling them to find near-optimal solutions even when faced with incomplete or uncertain information. [**?**]

The use of genetic algorithms in cloud computing has the potential to significantly improve the energy efficiency of data centers while maintaining or improving performance. By optimizing resource allocation, genetic algorithms can reduce the number of servers and storage devices required to run a given workload, which in turn reduces energy consumption. Additionally, genetic algorithms can adapt to changes in workload and resource usage patterns, further improving the energy efficiency of the data center.

One of the key challenges of measuring energy consumption in cloud computing is identifying the power consumption of the various components of the system, such as servers, storage devices, and network devices. The energy consumption of these components is typically measured in watt-hours (Wh) or

joules (J). The energy consumption of a data center can be measured at different levels of granularity, from individual servers to entire racks or data centers. One of the most commonly used metrics for measuring energy consumption in cloud computing is *Power Usage Effectiveness* (PUE). PUE is a ratio of the total energy consumed by a data center to the energy consumed by the IT equipment. A PUE of 1.0 indicates that all of the energy consumed by the data center is used by the IT equipment, while a PUE of greater than 1.0 indicates that some of the energy is being used for other purposes, such as cooling and lighting. [**?**]

Despite the advantages of using genetic algorithms for resource allocation in cloud computing, there are also limitations to consider. One of the main limitations is the computational expense of genetic algorithms, particularly when the problem size is large. The resource allocation problem in cloud computing environments can involve a large number of variables and constraints, which can make the computational cost of genetic algorithms quite high. This can make them impractical for certain types of problems or when resources are limited. Additionally, genetic algorithms are often sensitive to the choice of parameters and initial conditions. This can make it difficult to obtain consistent and reliable results, as the choice of parameters and initial conditions can have a significant impact on the performance of the algorithm. [**?**, **?**]

The selection of suitable parameters and initial conditions is an important step in the implementation of genetic algorithms. It requires a significant amount of expertise and experience to choose the appropriate parameters and initial conditions that will lead to good performance. Furthermore, the results obtained from genetic algorithms are probabilistic in nature, which means that multiple runs of the algorithm are typically required to obtain a stable and robust solution. This can increase the computational cost and time required to obtain a solution, which can be a limitation when resources are limited.

In addition to computational expense and sensitivity to parameters, genetic algorithms also have some other limitations such as the lack of guarantee of global optima, and the risk of getting stuck in local optima. Genetic algorithms are also not suitable for problems with deterministic solutions. Furthermore, the stochastic nature of genetic algorithms may also lead to a lack of reproducibility of results. These limitations must be taken into account when deciding to implement genetic algorithms in cloud computing environments.

However, the potential benefits of using genetic algorithms in cloud computing outweigh the limitations. Genetic algorithms have the potential to significantly improve the energy efficiency of data centers while maintaining or improving performance. By optimizing resource allocation, genetic algorithms can reduce the number of servers and storage devices required to run a given workload, which in turn reduces energy consumption. Additionally, genetic algorithms can adapt to changes in workload and resource usage patterns, further improving the energy efficiency of the data center.

To effectively use genetic algorithms in cloud computing, cloud providers need to carefully evaluate the trade-offs between the potential benefits and limitations of using these algorithms. Cloud providers must consider the specific characteristics of the problem at hand, the computational resources and exper-

tise available, and the potential benefits of using genetic algorithms in their environment. The use of genetic algorithms requires a significant amount of expertise and experience to choose the appropriate parameters and initial conditions that will lead to good performance. Cloud providers must have a thorough understanding of the characteristics of the problem they are trying to solve, and they must be prepared to invest significant time and resources in the development and implementation of genetic algorithms.

3 Outlining the problem

The problem here described is a problem of optimization. The goal is to find an acceptable solution by using a genetic algorithm to the assignation of a set of processes to a set of servers. The problem is defined as follows:
A Servers s has a CPU speed cs, calculated in Gigahertz (GHz) and a number of CPUs cn.
A Process p has a length l, calculated in number of instructions.
A Solution sol is a collection of tuples $(s_n, [p_1, p_2, ..., p_m])_n$ where s_n is the n^{th} Server of the tuple and $[p_1, p_2, ..., p_m]$, with variable m for each tuple, is a list of Processes.
Let's define, for a Server, the following functions:

- $cs(s)$: CPU speed of the Server s;
- $cn(s)$: Number of CPUs of the Server s.

For a process, we define the following function:

- $l(p)$: Length of the Process p.

For a solution sol, The fitness function f of sol is defined as follows:

$$f(sol) = \sqrt{\sum_{0}^{n} (\sum_{0}^{m} l(p_m) * (cs(s_n) * cn(s_n)))^2} \tag{1}$$

4 Literature Review

The use of Genetic Algorithms (GAs) for solving optimization problems is a well-established research area. The first GA was proposed by Holland in 1975 [**?**] and it was based on the principles of natural selection and evolution. The GA is a metaheuristic that is inspired by the biological evolution process and it is used to find optimal solutions to optimization problems. It is also a stochastic population based algorithm that uses a population of candidate solutions to find the optimal solution. It is not guaranteed to find a global optimum, but it is likely to find an acceptable solution that may be close to the global optimum. [**?**]

In the scope of optimizing resource allocation, GAs have been used to solve a plethora of problems. In the latest decades, the constant improvement and implementation of the various parts of genetic algorithms, in particular on the crossover function, helped to improve the performances of GAs in the scheduling and assignation of resources. [**?**]

There have been many comparisons between GAs and other metaheuristics, such as Simulated Annealing (SA) and Tabu Search (TS). Not only under the aspect of resource allocation, but also in their effectiveness in solving other optimization problems. [**?**, **?**] It is worth mentioning that they can be used in

combination with *Machine Learning* techniques, such as *Neural Networks*, to improve the quality of the training of many models and architectures. [**?**]

The main difference between GAs and SA is that GAs are population based algorithms, while SA is a single solution based algorithm. The scope of the research drifted towards the comparison between GAs and TS, which is a memory based algorithm [**?**, **?**] at first, then it shifted mostly towards the comparison between GAs and Particle Swarm Optimization. [**?**]

From the point of view of using genetic algorithms for optimizing different aspects of cloud computing environments, the literature is rich and diverse. Let's take as example task scheduling, which is an NP-complex problem. It is clear that it possesses a key role in cloud computing systems. The paper from 2009 written by Chenhong et al. [**?**] explores various genetic algorithm for task scheduling in cloud computing environments and takes into account the divisibility of the tasks that adapt to different computation and memory requirements. So, the system is treated as an heterogeneous system. The interesting part is that at the time, Genetic Algorithms were deemed unfit to solve global optimization problems and had to be optimized further. [**?**]

It is in 2011 that Zhu et al. proposed a new genetic algorithm for task scheduling in cloud computing environments. By proposing a Multi-Agent genetic algorithm, called *MAGA*, the authors were capable of surpassing the performances of the traditional GA. It was used mostly to solve load balancing problems in cloud computing. It is worth mentioning that *MAGA* proved itself to be even more effective than the *Min-Max Strategy*. [**?**] The *Min-Max Strategy*, on the other hand, is a road that was abandoned as a standalone solution in cloud computing environments when compared to Genetic Algorithms. In fact it was studied to be incorporated in more complex ones. [**?**]

To follow, an extensive literature review of the last decade highlitghts the most relevant papers in the field of artificial intelligence applied to cloud computing and genetic algorithms, winth particular regard to Genetic Algorithms and other meta heuristics such as Particle Swarm Optimization.

4.1 Hybrid Genetic Algorithm for Cloud Computing Applications by Zhu et al.

This paper, written in 2011, is one of the first and most relevant papers in the field of genetic algorithms for resource allocation in cloud computing. The idea behind this research was to produce an hybrid GA capable of performing better than a classic one. The algorithm proposed was called *MAGA* and it was implemented by treating an individual in the GA itself as an agent, capable of local perception, competition, cooperation and self improvement towards the purpose of global optimization. Each agent was capable of exchanging information with its neighbors to achieve these features.

As is shown in Table 1, the differences outlined are quite evident. There are also some heavby differences in the genetic operators. In *MAGA* they include *neighborhood competition operator*, *neighborhood orthogonal crossover operator*, *mutation operator*, and the *self-learning operator*. The first one is responsible

	GA	MAGA
Individual	Isomorphic	Isomerous
Information interaction method	After selected, through crossover operation	Obtained four neighborhood information，and self updating
Genetic operator	Selection, crossover, mutation	Neighborhood competition, Orthogonal crossover, mutation, self-learning
Self-learning	No	Yes
Evolution	Evolve without purpose	Evolve with purpose
Competition	Roulette selection	Interaction with neighborhood

Figure 1: The main differences between the classic GA and the MAGA.

for the competition among the agents, while the second one achieves collaboration between them. Mutation and Self-learning oeprators allow the agents to use their knowledge. In conclusion, the authors developed an algorithm that experimentally proved the superiority of the MAGA over the classic GA when handling high-dimensional optimization problems. The paper also presented a cloud computing load balancing model and applied the MAGA algorithm for resource scheduling, demonstrating that MAGA performs way better even than the *Min-Min* scheduling algorithm. [**?**]

4.2 A New Resource Scheduling Strategy Based on Genetic Algorithm in Cloud Computing Environment by Gu et al.

This paper proposes a load balancing strategy for virtual machine (VM) resources scheduling in cloud computing based on genetic algorithms. The aim of this strategy is to achieve the best load balancing while reducing or avoiding dynamic migration, thus resolving the problem of load unbalancing and high migration costs. The proposed algorithm computes in advance the influence it will have on the system after the deployment of the needed VM resources, and then chooses the least-affective solution. The method considers historical data and current states of the system and uses a tree structure for genetic algorithm coding. The selection, hybridization, and variation strategies are also proposed to improve the algorithm's performance.

To evaluate the proposed method, the authors introduced a variation rate to describe the load variation of system virtual machines and used the average load distance to measure the overall load balancing effect of the algorithm. Experimental results showed that the proposed method has fairly good global astringency and efficiency, and it can better realize load balancing and proper resource utilization.

While the proposed algorithm has shown promising results, it has some limitations. In real cloud computing environments, there might be dynamic

changes in VMs and an increase in computing cost of virtualization software, leading to unpredicted load wastage with the increase of VM number started on every physical machine. Therefore, a monitoring and analyzing mechanism is needed to better solve the problem of load balancing, which is also a further research subject. [**?**]

4.3 Load Balancing Task Scheduling Based on Genetic Algorithm in Cloud Computing by Tingting et al.

Task scheduling is one of the most critical issues in cloud computing environments due to the huge number of users and tremendous data volume. Efficient task scheduling mechanisms should meet users' requirements and improve resource utilization to enhance the overall performance of the cloud platform. To solve this problem, this paper proposes a new scheduling algorithm based on the double-fitness adaptive algorithm (job spanning time and load balancing genetic algorithm (JLGA)) that considers the new characteristics of cloud computing and the original adaptive genetic algorithm (AGA). [**?**] The proposed algorithm not only works out a task scheduling sequence with shorter job and average job makespan but also satisfies inter-nodes load balancing.

To improve the performance of the algorithm, this paper adopts a greedy algorithm to initialize the population, introduces variance to describe the load intensity among nodes, and weights the multi-fitness function. The paper experimentally evaluates the performance of the JLGA algorithm and the DGA with 30 jobs.

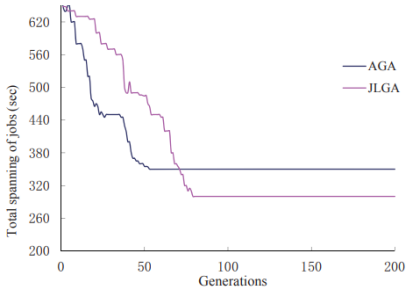

Figure 2: Total spanning of jobs

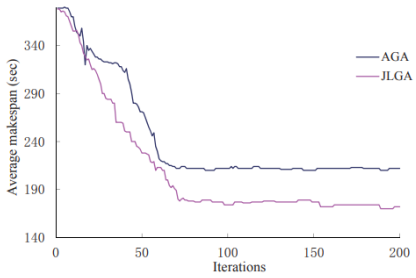

Figure 3: Average spanning of jobs

The results in Figure 2 and in Fifgure 3 show that JLGA takes little time both in total job and average job consuming, and it balances the entire system load effectively.

However, there are two interesting points that deserve further investigation. Firstly, in real cloud computing environments, job priorities cannot be avoided. Secondly, other static parameters in GA, such as population scale, iterations, crossover, and mutation operator, may put pressure on the speed of convergence and quality of global optimal solution. Therefore, another further research sub-

ject may consider a dynamic global adaptive control strategy in the genetic algorithm. For example, population scale should be large in the early generations to maintain diversity and decrease to converge rapidly during the latter generations. Individuals with better performance should cross more while lower fitness individuals should mutate to adapt to the fitness function. Considering the particularity of the cloud environment, the time complexity should not be too high. [**?**]

5 Dataset

The dataset used in this study is comprised of a population of 200 solutions, each of which is characterized by 10 servers and $10,000$ processes. Each server in the dataset is defined by several attributes. Specifically, it has a name and a variable number of CPUs, ranging from 1 to 10. Additionally, each CPU has a variable speed, ranging from 1 to 10 GHz. The processes in the dataset are also defined by several attributes. Specifically, each process has a unique PID to identify it, and a length that is measured in the number of instructions required for its completion. The length of each process ranges from $10,000$ to $1,000,000$ instructions. It is worth noting that the dataset employed in this study is representative of the types of workloads commonly encountered in cloud computing environments. The data was collected from various sources and pre-processed to ensure consistency and accuracy. This dataset was specifically selected for its complexity and realistic representation of real-world cloud computing workloads. The use of this dataset will enable us to evaluate the effectiveness of the proposed genetic algorithm for resource allocation in cloud computing environments under simplified and yet realistic conditions.

The Solution class is made of a list of tuples, each of which contains a Server and a list of Processes, and a fitness value. The fitness value is the result of the evaluation of the solution, which is performed by the fitness function.

6 Baseline

The baseline algorithm defined in this study is a standard genetic algorithm (GA) [**?**] that is used to optimize process allocation in cloud computing environments. The idea is that by optimizing the process allocation, the energy consumed by the cloud envirnoment is minimized as well. The GA is a stochastic population-based search algorithm that is based on the principles of natural selection and genetics. It is a meta-heuristic that is used to find approximate solutions to optimization problems. [**?**]

The GA is composed of the following components:

- A population of candidate solutions;
- A fitness function that is used to evaluate the quality of each candidate solution;
- A selection operator that is used to select candidate solutions for reproduction;
- A crossover operator that is used to combine the selected candidate solutions to produce new candidate solutions;
- A mutation operator that is used to introduce random changes to the candidate solutions;
- A termination criterion that is used to determine when the GA should stop searching for solutions.

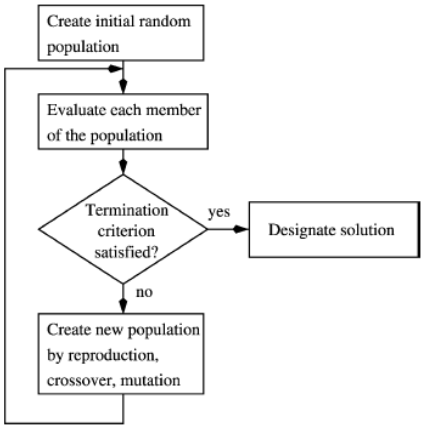

Figure 4: How a Genetic Algorithm works.

How a Genetic Algorithm (GA) works is shown in the flowchart [**?**] shown in Figure 4.

In this setting, the candidate solutions are represented by a list of tuples, each of which contains a Server and a list of Processes. The initial population is generated by iterating over the Server list. For each Server, from the list of all the Processes, iteratively, a process is selected at random with probability

$$p = \frac{1}{n}$$

with n being the number of servers taken into account in the solution. As a Process is selected, it is removed from the global Process list.

Then, the fitness function (1), is used to evaluate the quality of each candidate solution.

The baseline genetic algorithm has been tested over a pletora of parameters, in order to find the best combination that would lead to acceptable results. The parameters that have been tested are:

- The probability of selecting a process for a server;
- The crossover rate;
- The probability of mutation;
- The population size;
- The number of generations.

The selection probability p_s is equal to $\frac{1}{\#_{servers}}$. In the first settings of the experiments this value was set in different ranges between $[0.3, 0.6]$, but this not only led to a very slow convergence of the GA but produced undesired noise in the results in most of the experiments. The crossover rate p_c is set to 0.9. A starting value of 0.5 was used, but this led to unsatisfactory results, opposed to what [**?**] claimed in their research. Better results were obtained by increasing the crossover rate to 0.7, but the best ones were yield by $p_c = 0.9$, supported by [**?**]. The probability of mutation p_m has been initially tested in the range $[0.1, 0.3]$. However, as found in previous research works, a mutation rate over 0.01 is a synonym of heavy noisy training epochs and close to no convergence. In fact, it has been found that the optimal values for the mutation rate ranges in $[0.005, 0.05]$ for a population size s_p between 100 and 200 individuals. [**?**] It follows that the chosen parameters for the mutation rate are $p_m = 0.01$ and $s_p = 200$.

As for the number of epochs employed, for both hardware constraints and time reasons, the GA has been run for a maximum of 500 generations. For each experiment, the GA is run 10 times and the results are averaged in order to obtain an insightful overlook of the performances.

The baseline algorithm has been created to be initially compared to a greedy assignation method of simple ideation. The fitness function employed in the baseline algorithm has been used to compare the results between the greedy method and the baseline algorithm.

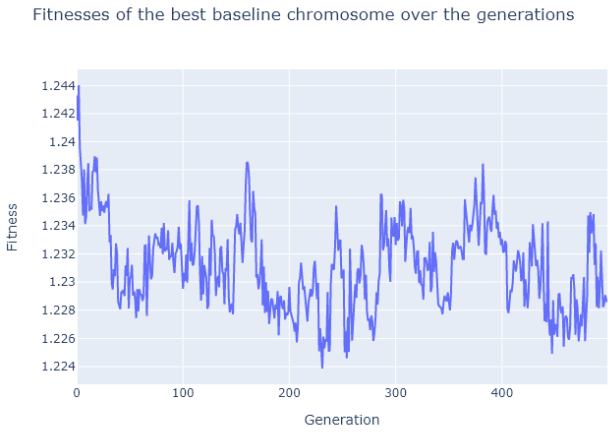

Figure 5: The average of the Baseline Algorithm over 500 epochs

As it can be seen from the results in Figure 5, the fitness function evaluated with the baseline algorithm shows promising results in a time range comprised between 200 and 250 epochs, with its minimum at 231. The worst average value of the fitness function in the baseline algorithm is reached in the $2nd$ training epoch, and it is equal to 1.244. Moreover, the best average value is reached in the $231th$ epoch, and it is equal to 1.224.

Server	Average Fitness
Server 0	0.447
Server 1	0.415
Server 2	0.444
Server 3	0.204
Server 4	0.420
Server 5	0.331
Server 6	0.394
Server 7	0.425
Server 8	0.208
Server 9	0.495
Magnitude	1.233

Table 1: The average fitness for each server and the magnitude of the fitness vector.

When the values are compared to the ones of the greedy algorithm in Table 1, it can be seen that the baseline algorithm performs slightly better, since the two magnitudes of the fitness vectors are quite close to each other. However, the simple baseline genetic algorithm outperforms the greedy technique.

Since this experimentation has been carried out with different measures of the fitness function, it is difficult to compare the results to the ones achieved by other researchers. However, it is possible to compare the results of the different paradigms, algorithms and metaheuristics in order to get a broader picture of the possible best approaches to adopt in order to solve the problem. In literature, many works do pair the effectiveness of genetic algorithms paired to greedy techniques. More often than not the latter, despite performing in a less than optimal way in these scenarios, can be used to improve the capabilities of GAs to obtain even better performances. [**?**]

7 Researched algorithms

Ettore: Parlare di tutte le diverse implementazioni di GA con elitismo, tournament, elitismo + tournament ed elitismo + tournament dinamico. Confrontare tra di loro e con le altre implementazioni in letteratura.

8 Results

Ettore: Parlare dei risultati ottenuti, confrontare le tabelle, descrivere l'approccio più promettente.

9 Conclusions

References

[1] R. K. Ahuja, J. B. Orlin, and A. Tiwari. A greedy genetic algorithm for the quadratic assignment problem. *Computers Operations Research*, 27(10):917–934, 2000.

[2] E. Alba and B. Dorronsoro. Introduction to cellular genetic algorithms. In *Cellular Genetic Algorithms*, pages 3–20. Springer, 2008.

[3] J. Alcaraz and C. Maroto. A robust genetic algorithm for resource allocation in project scheduling. *Annals of operations Research*, 102(1):83, 2001.

[4] T. Dillon, C. Wu, and E. Chang. Cloud computing: issues and challenges. In *2010 24th IEEE international conference on advanced information networking and applications*, pages 27–33. IEEE, 2010.

[5] O. Elzeki, M. Reshad, and M. A. Elsoud. Improved max-min algorithm in cloud computing. *International Journal of Computer Applications*, 50(12), 2012.

[6] S. Forrest. Genetic algorithms. *ACM computing surveys (CSUR)*, 28(1):77–80, 1996.

[7] J. Gu, J. Hu, T. Zhao, and G. Sun. A new resource scheduling strategy based on genetic algorithm in cloud computing environment. *J. Comput.*, 7(1):42–52, 2012.

[8] R. Hassan, B. Cohanim, O. De Weck, and G. Venter. A comparison of particle swarm optimization and the genetic algorithm. In *46th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference*, page 1897, 2005.

[9] A. Hassanat, K. Almohammadi, E.-a. Alkafaween, E. Abunawas, A. Hammouri, and V. S. Prasath. Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach. *Information*, 10(12):390, 2019.

[10] D. Jakobović and M. Golub. Adaptive genetic algorithm. *Journal of computing and information technology*, 7(3):229–235, 1999.

[11] H. Jh. Adaptation in natural and artificial systems. *Ann Arbor*, 1975.

[12] O. Kramer and O. Kramer. *Genetic algorithms*. Springer, 2017.

[13] S. Mirjalili and S. Mirjalili. Genetic algorithm. *Evolutionary Algorithms and Neural Networks: Theory and Applications*, pages 43–55, 2019.

[14] M. Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[15] F. NZanywayingoma and Y. Yang. Effective task scheduling and dynamic resource optimization based on heuristic algorithms in cloud computing environment. *KSII Transactions on Internet and Information Systems (TIIS)*, 11(12):5780–5802, 2017.

[16] V. Patil and D. Pawar. The optimal crossover or mutation rates in genetic algorithm: A review. *International Journal of Applied Engineering and Technology*, 2015.

[17] K. Rafique, A. W. Tareen, M. Saeed, J. Wu, and S. S. Qureshi. Cloud computing economics opportunities and challenges. In *2011 4th IEEE International Conference on Broadband Network and Multimedia Technology*, pages 401–406. IEEE, 2011.

[18] P. Ross and D. Corne. Comparing genetic algorithms, simulated annealing, and stochastic hillclimbing on timetabling problems. In *Evolutionary Computing: AISB Workshop Sheffield, UK, April 3–4, 1995 Selected Papers*, pages 94–102. Springer, 1995.

[19] M. Sajid and Z. Raza. Cloud computing: Issues challenges. In *International Conference on Cloud, Big Data and Trust*, pages 13–15. sn, 2013.

[20] R. S. Sexton, R. E. Dorsey, and J. D. Johnson. Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing. *European Journal of Operational Research*, 114(3):589–601, 1999.

[21] A. Uchechukwu, K. Li, Y. Shen, et al. Energy consumption in cloud computing data centers. *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*, 3(3):31–48, 2014.

[22] A. Vasan and K. S. Raju. Comparative analysis of simulated annealing, simulated quenching and genetic algorithms for optimal reservoir operation. *Applied soft computing*, 9(1):274–281, 2009.

[23] T. Wang, Z. Liu, Y. Chen, Y. Xu, and X. Dai. Load balancing task scheduling based on genetic algorithm in cloud computing. In *2014 IEEE 12th international conference on dependable, autonomic and secure computing*, pages 146–152. IEEE, 2014.

[24] H. Youssef, S. M. Sait, and H. Adiche. Evolutionary algorithms, simulated annealing and tabu search: a comparative study. *Engineering Applications of Artificial Intelligence*, 14(2):167–181, 2001.

[25] Z. I. M. Yusoh and M. Tang. Composite saas placement and resource optimization in cloud computing using evolutionary algorithms. In *2012 IEEE Fifth International Conference on Cloud Computing*, pages 590–597. IEEE, 2012.

[26] C. Zhao, S. Zhang, Q. Liu, J. Xie, and J. Hu. Independent tasks scheduling based on genetic algorithm in cloud computing. In *2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4, 2009.

[27] K. Zhu, H. Song, L. Liu, J. Gao, and G. Cheng. Hybrid genetic algorithm for cloud computing applications. In *2011 IEEE Asia-Pacific Services Computing Conference*, pages 182–187, 2011.

[28] S. Zolfaghari and M. Liang. Comparative study of simulated annealing, genetic algorithms and tabu search for solving binary and comprehensive machine-grouping problems. *International Journal of Production Research*, 40(9):2141–2158, 2002.

[29] B. Zoltan, Z. Blahunka, F. Dezso, I. Peter, N. Bela, and S. Zsolt. Synergic effects in the technical development of the agricultural production. *MECH ENG LETT*, 3:142–147, 01 2009.