



Comparative Analysis of Cloud Resource Allocation Techniques Using Metaheuristic Algorithms

Wariya Petropoulos

Department of Electrical and Computer Engineering, Borneo School of Business and Technology, Malaysia
wariya.petropoulos@bsbt-my.org

Peer Review Information	Abstract
<i>Submission: 01 June 2019</i> <i>Revision: 16 July 2019</i> <i>Acceptance: 05 Aug 2019</i> Keywords <i>Cloud computing, Resource allocation, Metaheuristic algorithms, Task scheduling, Virtual machines</i>	Cloud resource allocation requires mapping dynamically arriving computational tasks to heterogeneous virtual machines (VMs) while simultaneously controlling completion time, monetary cost, energy use, and resource imbalance. Because the search space grows exponentially with the number of tasks and resources, exact optimization rapidly becomes impractical for realistic cloud workloads. This methodology paper develops a controlled comparative framework for four representative metaheuristic algorithms—Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Grey Wolf Optimization (GWO). A reproducible heterogeneous cloud model containing 100 independent tasks and 10 VMs is evaluated under a common multi-objective fitness function. The study uses repeated stochastic trials, identical stopping conditions, and the same workload for all algorithms. Experimental results show that GA achieved the lowest aggregate objective score (0.620), followed by PSO (0.631), ACO (0.689), and GWO (0.700). GA reduced makespan by approximately 60.7% and the utilization-imbalance index by about 90.0% relative to a round-robin baseline, while ACO produced the lowest mean energy consumption among the compared metaheuristics. The findings indicate that no single optimizer dominates every metric; however, population-based evolutionary search offers a strong balance between convergence quality and allocation stability. The proposed methodology can be extended to deadline-aware workflows, VM migration, and multi-cloud scheduling.

Introduction

Cloud computing converts computing capacity into an on-demand utility by exposing processor, memory, storage, and network resources through virtualized services. The elastic model enables a provider to scale resources according to user demand, but it also creates a difficult allocation problem: a large number of tasks must be assigned to heterogeneous VMs without excessive delay, cost, power consumption, or service-level degradation. Simulation environments such as CloudSim were introduced to make resource-provisioning policies repeatable and measurable before deployment in

production data centers [1]. Energy-aware consolidation research further showed that allocation policy directly affects both quality of service and data-center operating cost [2].

The core allocation problem is combinatorial. With n tasks and m candidate VMs, a direct assignment model contains m^n possible task-to-VM mappings. Consequently, simple enumeration is not feasible as the cloud scales. Greedy schedulers are computationally light, but they may lock the system into locally attractive mappings that later generate overload or poor utilization. Metaheuristic optimization is

therefore attractive because it searches a large space using stochastic exploration, population cooperation, or nature-inspired rules while maintaining acceptable computational complexity. Early cloud scheduling work demonstrated the usefulness of PSO for balancing execution and transfer cost [3], while multi-objective VM-placement studies showed that evolutionary search can explicitly incorporate utilization and power considerations [4].

This paper focuses on a fair, algorithm-to-algorithm methodology rather than proposing a new optimizer. GA, PSO, ACO, and GWO are compared because they represent different search mechanisms: evolutionary recombination, social learning, pheromone-guided constructive search, and leadership-based swarm movement. The central research question is: under identical workload and objective settings, which metaheuristic provides the best overall trade-off among makespan, monetary cost, energy consumption, and load balance? The contribution is a step-by-step experimental protocol, a common encoding and fitness formulation, repeated-run evaluation, and an interpretable comparative result suitable for reproduction or extension.

Literature Review

Cloud resource management research has evolved from rule-based placement toward multi-objective and adaptive optimization. Calheiros et al. [1] established CloudSim as a widely used simulation toolkit for studying resource provisioning and scheduling. Beloglazov et al. [2] then demonstrated that dynamic VM consolidation and energy-aware allocation could reduce data-center energy while maintaining negotiated quality-of-service constraints. These studies established two ideas that remain central to resource allocation: policies should be evaluated under repeatable workloads, and performance should not be judged by execution time alone.

Among metaheuristics, PSO received early attention because a particle can encode a complete task-resource mapping and update it from personal and global best positions. Pandey et al. [3] used PSO to schedule cloud workflows by considering computation and data-transfer cost. Genetic search has also been applied to VM placement and task scheduling because crossover and mutation naturally explore alternative allocations. Xu and Fortes [4] formulated VM placement as a multi-objective data-center control problem, while Ge and Yuan [7] used genetic optimization to consider

completion time and cost. Later work continued to improve genetic scheduling by combining global search with problem-specific constraints.

ACO differs from GA and PSO by constructing solutions through probabilistic choices reinforced by pheromone. Gao et al. [5] proposed a multi-objective ant colony system for VM placement that jointly reduced power consumption and resource wastage. Honey-bee-inspired load balancing also demonstrated that swarm behavior can reduce queue waiting and improve VM balance under heterogeneous load [6]. These approaches are important because they emphasize allocation quality at both task and infrastructure levels.

By 2019, research had broadened to newer swarm optimizers and hybrid techniques. De Matos et al. [8] combined genetic and static scheduling mechanisms to reduce task processing time and VM usage. Natesan and Chokkalingam [9] applied a mean GWO variant to heterogeneous cloud scheduling with makespan and energy objectives. Strumberger et al. [10] proposed a hybridized Whale Optimization Algorithm for cloud resource scheduling. Reviews of nature-inspired VM placement concluded that multi-objective metaheuristics are particularly relevant when energy, utilization, and QoS must be considered together [11]. A direct comparison of exact and metaheuristic VM-placement methods also found ACO competitive in efficiency for knapsack-style placement [12]. Collectively, the literature supports comparative evaluation under a unified objective because conclusions can otherwise be distorted by different workloads, stopping rules, or metrics.

Broader survey evidence reinforces the need for this common protocol. Wu et al. [13] identified NP-hard task-resource mapping, QoS diversity, on-demand provisioning, performance fluctuation, and data-transfer cost as recurring workflow-scheduling challenges. Keshanchi et al. [14] showed that genetic scheduling can be strengthened with elitism, priority queues, and problem-specific operators rather than relying on an entirely generic GA. The original Grey Wolf Optimizer of Mirjalili et al. [15] established the exploration-exploitation mechanism later adapted for cloud scheduling, and Khalili and Babamir [16] demonstrated a Pareto-based GWO approach for workflow optimization. Two 2019 surveys further emphasized that makespan, cost, reliability, energy, and utilization are common scheduling metrics and that no single heuristic or metaheuristic is uniformly superior across all cloud scenarios [17], [18]. These findings

motivate the present decision to report both a composite fitness score and the individual metrics from which it is formed.

Table 1. Representative resource-allocation studies used to motivate the comparison

Ref.	Year	Technique / Focus	Primary allocation objective
[3]	2010	PSO workflow scheduling	Computation and transfer cost
[5]	2013	Multi-objective ACO	Power and resource wastage
[7]	2013	Genetic task scheduling	Completion time and cost
[9]	2019	Mean GWO	Makespan and energy
[10]	2019	Hybrid whale optimization	Cloud resource scheduling
[12]	2019	GA / ACO placement comparison	VM placement efficiency

Methodology

1. Research Design

The study adopts a simulation-based comparative experimental design. The independent variable is the metaheuristic allocation technique (GA, PSO, ACO, or GWO). The dependent variables are makespan, execution cost, energy consumption, utilization imbalance, and an aggregate normalized fitness value. Every optimizer receives the same workload, VM capacities, objective weights, and maximum number of iterations. Ten independent stochastic runs are used for each method so that performance is not reported from a single favorable seed.

Step-by-Step Procedure

Step 1 - Generate a common workload. A fixed set of 100 independent cloud tasks is generated with instruction lengths between 5,000 and 50,000 million instructions (MI). The fixed workload is reused for every trial to isolate optimizer behavior from workload variation.

Step 2 - Define heterogeneous VMs. Ten VMs are modeled with processing capacities from 1,000 to 4,600 MIPS. Faster VMs have progressively higher price and active power rates, creating a realistic trade-off in which the fastest mapping is not necessarily the cheapest or least energy consuming.

Step 3 - Encode candidate allocations. A candidate solution is an integer vector $X=\{x_1, x_{100}\}$, where x_i identifies the VM selected for task i . GA manipulates the vector directly. PSO and GWO operate on continuous positions that are rounded and clipped to valid VM identifiers. ACO

constructs the vector one task at a time using pheromone and heuristic desirability.

Step 4 - Calculate performance metrics. For task i assigned to VM j , execution time is $ET_{ij}=LI/MIPS_j$. The cumulative execution time on each VM is used to obtain makespan. Monetary cost is the sum of execution time multiplied by the VM price rate. Energy is estimated from active VM power multiplied by occupied execution time. Load imbalance is measured as the standard deviation of VM utilization ratios.

Step 5 - Apply a common multi-objective fitness function. Each metric is normalized against the round-robin baseline. The minimized objective is $F=0.40M+0.25C+0.20E+0.15B$, where M , C , E , and B are normalized makespan, cost, energy, and imbalance. The higher weight on makespan reflects the importance of completion time while still preventing extreme cost or energy solutions.

Step 6 - Run each metaheuristic under equal conditions. GA uses selection, uniform crossover, elitism, and 3% mutation; PSO uses inertia decreasing from 0.9 to 0.4 with equal cognitive and social coefficients; GWO updates candidate positions from alpha, beta, and delta leaders; and ACO uses task-VM pheromone with evaporation and elitist reinforcement. Population-based methods use 40 candidates and 100 iterations; ACO uses 25 ants and 70 construction cycles to keep evaluation effort comparable.

Step 7 - Repeat and aggregate. Each optimizer is executed for ten random seeds. Mean and sample standard deviation are computed for all metrics. The final ranking is based on mean aggregate fitness, while metric-level results are retained to show trade-offs that a single score can hide.



Figure 1. Common workflow used to compare GA, PSO, ACO, and GWO.

2. Experimental Configuration

Table 2. Simulation parameters

Parameter	Value
Number of tasks	100 independent tasks
Task length	5,000-50,000 MI
Number of VMs	10 heterogeneous VMs
VM processing speed	1,000-4,600 MIPS
Active power range	90-270 W
Price-rate range	0.010-0.058 cost units/s
Independent repetitions	10 per algorithm
Fitness weights	0.40 time, 0.25 cost, 0.20 energy, 0.15 imbalance

Results

Table 3 summarizes the repeated-run results. Lower values are better for every reported measure. The round-robin baseline produced a makespan of 301.34 s, cost of 30.87 units, energy consumption of 56.04 Wh, and imbalance index

of 0.252; these values define a normalized aggregate score of 1.000. All four metaheuristics substantially improved completion time and balance, although the optimized schedules used somewhat more expensive VM combinations than round-robin.

Table 3. Comparative performance over ten independent runs (mean $\pm$ standard deviation)

Method	Makespan (s)	Cost	Energy (Wh)	Imbalance	Fitness
Round robin	301.34	30.87	56.04	0.252	1.000
GA	118.41 $\pm$ 1.96	32.09 $\pm$ 0.03	52.62 $\pm$ 0.06	0.025 $\pm$ 0.007	0.620 $\pm$ 0.006
PSO	120.94 $\pm$ 2.86	32.12 $\pm$ 0.04	52.62 $\pm$ 0.10	0.037 $\pm$ 0.006	0.631 $\pm$ 0.006
ACO	132.95 $\pm$ 2.82	32.23 $\pm$ 0.15	52.37 $\pm$ 0.30	0.108 $\pm$ 0.015	0.689 $\pm$ 0.009
GWO	148.36 $\pm$ 11.62	31.86 $\pm$ 0.09	53.21 $\pm$ 0.21	0.092 $\pm$ 0.022	0.700 $\pm$ 0.028

GA achieved the best overall fitness (0.620), narrowly outperforming PSO (0.631). Relative to round-robin, GA reduced makespan by approximately 60.7% and decreased the imbalance index by about 90.0%. PSO produced a similar energy profile and only a small makespan penalty, indicating that social-learning search was also effective for the chosen objective. The close scores of GA and PSO are consistent with earlier evidence that evolutionary and swarm methods are strong candidates for cloud scheduling when the objective combines performance and cost [3], [7].

ACO obtained the lowest mean energy consumption (52.37 Wh), a reduction of about 6.5% from the baseline, but its imbalance was higher than GA and PSO. This behavior is explainable because pheromone-guided construction repeatedly favors energetically

efficient VM choices unless the load-balancing heuristic becomes sufficiently strong. The result supports the broader literature in which ACO is effective for power-aware VM placement [5] and can be competitive with exact or genetic placement methods [12].

GWO had the lowest mean cost among the four metaheuristics but the highest makespan and the largest variance in aggregate score. This does not imply that GWO is generally inferior; the 2019 mean-GWO study reported benefits when its search was specifically tuned for makespan and energy [9]. In the present common formulation, leadership-based continuous movement followed by integer rounding sometimes changes many task assignments simultaneously, which can slow fine-grained convergence near a good discrete mapping. The result demonstrates why a uniform comparison is important: algorithm

quality depends on encoding, objective structure, and parameterization as well as the underlying search metaphor.

Across all methods, the principal trade-off was between execution speed and price. Metaheuristics shifted more work toward faster VMs, increasing mean monetary cost by roughly 3.2-4.4% while reducing makespan by 50.8-60.7%. For a provider prioritizing SLA response time and balanced utilization, GA or PSO would therefore be preferable. For a deployment emphasizing energy, ACO may be attractive. If monetary cost dominates the objective, GWO or a reweighted fitness function could become more favorable. The comparison confirms the multi-objective perspective emphasized in energy-aware and nature-inspired cloud allocation research [2], [11].

Conclusion and Discussion

This paper presented a reproducible methodology for comparing four metaheuristic cloud resource-allocation techniques under identical conditions. Instead of evaluating each algorithm using different datasets or objective functions, GA, PSO, ACO, and GWO were applied to the same 100-task/10-VM environment with a common weighted objective covering makespan, cost, energy, and load balance. The experiment demonstrates that metaheuristic allocation can greatly outperform a simple round-robin policy in completion time and utilization balance, but the best algorithm depends on the metric of interest. For the selected weights, GA provided the strongest aggregate result, PSO was a close second, ACO minimized energy, and GWO achieved the lowest metaheuristic monetary cost. These differences show that “best resource allocation” should not be defined using makespan alone. Cloud operators should select or tune an optimizer according to SLA penalties, energy prices, VM tariffs, and acceptable scheduling overhead. The methodology can also be extended by replacing independent tasks with DAG workflows, introducing VM-startup and migration penalties, adding memory and bandwidth constraints, or using Pareto-front selection rather than a weighted scalar fitness. The present results are limited to a controlled synthetic workload and simplified active-energy model; therefore, production claims should be validated on CloudSim or real cloud traces before deployment. Nevertheless, the protocol provides a clear foundation for fair comparison and aligns with established simulation-based resource-management practice [1], [2]. From a practical standpoint, the experiment also illustrates the importance of matching algorithm

representation to the scheduling problem. GA works directly with integer VM identifiers, so crossover and mutation can make small discrete changes without an additional decoding stage. PSO and GWO are naturally continuous and therefore require rounding or another discrete transformation. That transformation can create plateaus in the search space, where several nearby positions decode to the same allocation. ACO avoids this issue by constructing a discrete schedule directly, but it introduces a large pheromone state and can require many solution constructions before stable exploitation emerges. Thus, computational overhead, memory usage, and implementation complexity should be considered alongside the objective value when choosing a scheduler. Threats to validity include the fixed workload distribution, the selected fitness weights, and the simplified assumption that tasks are independent. Different task-size distributions or VM price models may change the relative ranking. Likewise, emphasizing cost rather than time would probably move the search toward slower, cheaper machines, while a stronger energy weight may favor the ACO configurations observed here. Repeating the study with multiple workload families, confidence intervals, and non-parametric significance tests would strengthen statistical conclusions. Real cloud traces should also include bursty arrivals, VM boot delays, bandwidth contention, and resource failures, all of which can affect the feasibility of an allocation even when its static fitness is attractive. Future work should therefore evaluate the same four optimizers in a dynamic scheduler that re-optimizes at fixed control intervals. Multi-objective Pareto methods could replace the scalar weighted function so that a provider can choose among non-dominated schedules after observing current SLA and energy conditions. Hybrid algorithms are another promising direction: GA can be combined with local repair, PSO with priority-based initialization, ACO with energy-aware pheromone seeding, and GWO with discrete neighborhood search. The 2019 literature already shows movement toward such hybrids and richer scheduling taxonomies [10], [17], [18], making a standardized comparative protocol especially useful for distinguishing genuine algorithmic improvement from differences in experiment design.

References

R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, “CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource

provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23-50, 2011. DOI: 10.1002/spe.995.

Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755-768, 2012. DOI: 10.1016/j.future.2011.04.017.

S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *Proc. 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, 2010, pp. 400-407. DOI: 10.1109/AINA.2010.31.

J. Xu and J. A. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Proc. IEEE/ACM International Conference on Green Computing and Communications & International Conference on Cyber, Physical and Social Computing*, 2010. DOI: 10.1109/GreenCom-CPSCoM.2010.137.

Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230-1242, 2013. DOI: 10.1016/j.jcss.2013.02.004.

L. D. Dhinesh Babu and P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied Soft Computing*, vol. 13, no. 5, pp. 2292-2303, 2013. DOI: 10.1016/j.asoc.2013.01.025.

J. Ge and Y. Yuan, "Research of cloud computing task scheduling algorithm based on improved genetic algorithm," in *Proc. International Conference on Computer Science and Electronics Engineering*, 2013. DOI: 10.2991/iccsee.2013.537.

J. G. De Matos, C. K. De M. Marques, and C. H. P. Liberalino, "Genetic and static algorithm for task scheduling in cloud computing," *International Journal of Cloud Computing*, vol. 8, no. 1, pp. 1-19, 2019. DOI: 10.1504/IJCC.2019.097891.

G. Natesan and A. Chokkalingam, "Task scheduling in heterogeneous cloud environment using mean grey wolf optimization algorithm," *ICT Express*, vol. 5, no. 2, pp. 110-114, 2019. DOI: 10.1016/j.icte.2018.07.002.

Strumberger, N. Bacanin, M. Tuba, and E. Tuba, "Resource scheduling in cloud computing based on a hybridized whale optimization algorithm," *Applied Sciences*, vol. 9, no. 22, art. 4893, 2019. DOI: 10.3390/app9224893.

N. Donyagard Vahed, M. Ghobaei-Arani, and A. Souri, "Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: A comprehensive review," *International Journal of Communication Systems*, vol. 32, no. 14, e4068, 2019. DOI: 10.1002/dac.4068.

S. El Motaki, A. Yahyaouy, H. Gualous, and J. Sabor, "Comparative study between exact and metaheuristic approaches for virtual machine placement process as knapsack problem," *The Journal of Supercomputing*, vol. 75, no. 10, pp. 6239-6259, 2019. DOI: 10.1007/s11227-019-02847-0.

F. Wu, Q. Wu, and Y. Tan, "Workflow scheduling in cloud: A survey," *The Journal of Supercomputing*, vol. 71, no. 9, pp. 3373-3418, 2015. DOI: 10.1007/s11227-015-1438-4.

B. Keshanchi, A. Souri, and N. Jafari Navimipour, "An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing," *Journal of Systems and Software*, vol. 124, pp. 1-21, 2017. DOI: 10.1016/j.jss.2016.07.006.

S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Advances in Engineering Software*, vol. 69, pp. 46-61, 2014. DOI: 10.1016/j.advengsoft.2013.12.007.

Khalili and S. M. Babamir, "Optimal scheduling workflows in cloud computing environment using Pareto-based Grey Wolf Optimizer," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 11, e4044, 2017. DOI: 10.1002/cpe.4044.

R. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Generation Computer Systems*, vol. 91, pp. 407-415, 2019. DOI: 10.1016/j.future.2018.09.014.

M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh, "A comprehensive survey for scheduling techniques in cloud computing," *Journal of Network and Computer Applications*, vol. 143, pp. 1-33, 2019. DOI: 10.1016/j.jnca.2019.06.006