



Cloud Computing Resource Allocation Using Hybrid Optimization Techniques

Oisin Balasingam

Department of Electrical and Computer Engineering, Coral Reef School of Systems Management, Mauritius
oisin.balasingam@crssm-mu.net

Peer Review Information

Submission: 23 Aug 2020

Revision: 09 Sep 2020

Acceptance: 24 Sep 2020

Keywords

Cloud Computing, Resource Allocation, Hybrid Optimization, Genetic Algorithm, Particle Swarm Optimization

Abstract

Efficient cloud resource allocation requires mapping heterogeneous user tasks to virtual machines (VMs) while simultaneously controlling execution time, service cost, load imbalance, and resource under-utilization. Single metaheuristics can provide useful schedules, but they may converge prematurely or spend excessive iterations exploring weak regions of the search space. This methodology paper proposes a Hybrid Genetic Algorithm-Particle Swarm Optimization with Load-aware Repair (HGA-PSO-LR) model for task-to-VM allocation. Genetic search supplies population diversity and global exploration, particle-swarm refinement exploits promising regions, and a final repair phase relocates selected tasks from overloaded VMs when the multi-objective fitness improves. A reproducible proof-of-concept simulation with 80 heterogeneous tasks and eight VMs is used to demonstrate the evaluation procedure. The proposed model is compared with Round Robin, a greedy earliest-completion heuristic, standalone PSO, and standalone GA. The hybrid method obtains the lowest composite objective score (0.451), reduces makespan from 246.54 s under Round Robin to 137.86 s, and increases average VM utilization from 63.54% to 98.64%. These values are specific to the stated fixed-seed simulation and are presented to demonstrate the methodology rather than as universal cloud-performance claims

Introduction

Cloud computing transforms computing infrastructure into elastic, service-oriented resources that can be provisioned and released on demand. The utility-oriented model described by Buyya et al. [1] makes resource management central to both provider efficiency and user quality of service. In Infrastructure-as-a-Service environments, task scheduling and VM allocation determine how effectively processors, memory, and virtualized hosts are used. As the number of tasks and heterogeneous resources grows, the allocation problem becomes combinatorial and is commonly treated as an NP-hard scheduling problem.

CloudSim provided a repeatable environment for evaluating provisioning and scheduling policies without requiring access to production data centers [2]. Subsequent research showed that resource allocation cannot be judged using execution time alone. Energy use and performance interference are also important because aggressive consolidation can reduce power consumption while increasing service-level risk [3]. Genetic algorithms have been applied to cloud load balancing because they can search large discrete assignment spaces and outperform simple dispatching rules in selected scenarios [4]. Particle swarm optimization has similarly been used to reduce workflow execution cost under cloud constraints [5].

The central research problem addressed in this paper is therefore: how can complementary optimization mechanisms be combined to obtain a better balance between makespan, monetary cost, resource utilization, and load distribution? The objective is not to claim a universally optimal scheduler, but to define a reproducible methodology that can be implemented in CloudSim or an equivalent simulator, benchmarked against conventional algorithms, and extended with additional QoS constraints. The study is restricted to literature published up to and including 2020, as required.

The methodology is organized around four objectives. First, it defines a discrete and reproducible encoding for mapping tasks to heterogeneous VMs. Second, it constructs a multi-objective fitness function that prevents a scheduler from optimizing response time at the expense of cost or severe load imbalance. Third, it combines complementary search operators so that global exploration is followed by focused local refinement. Fourth, it specifies a benchmarking protocol in which the same workload, VM pool, random seed, and termination budget are used for every competing scheduler. These objectives lead to three research questions: RQ1, does hybrid search reduce makespan compared with simple and single-metaheuristic schedulers? RQ2, does it improve VM utilization and load balance without materially increasing execution cost? RQ3, is the methodology sufficiently transparent to be reproduced and extended in a standard cloud simulator?

Literature Review

Metaheuristic scheduling developed rapidly as cloud research moved beyond first-come-first-served and Round Robin strategies. Rodriguez and Buyya [5] used PSO for deadline-constrained scientific workflow provisioning, demonstrating that resource selection and scheduling should be jointly considered when cloud costs vary. Jena [6] extended the discussion to multiple objectives, using nested PSO to balance processing time and energy in CloudSim. These studies motivate fitness functions that combine several cloud performance indicators rather than optimizing a single measure.

Hybridization was introduced to compensate for weaknesses of individual search algorithms. Dordaie and Navimipour [7] combined PSO with hill climbing to improve makespan, using the population-based optimizer for broad search and local search for refinement. Manasrah and Ba Ali [8] directly combined GA and PSO for workflow scheduling and reported improvements in execution time, cost, and load balancing over several standalone approaches. Gao et al. [9] later combined GA, artificial bee colony optimization, and a decoding heuristic for multi-objective workflow scheduling in IaaS clouds. Mansouri et al. [10] integrated modified PSO, evolutionary operators, and fuzzy reasoning to account for task length, CPU speed, RAM, and execution time. Finally, Pasdar et al. [11] demonstrated in 2020 that hybrid scheduling can also operate across private and public cloud resources, where cost, data movement, and changing execution conditions interact.

Table 1. Representative studies informing the proposed methodology

Ref.	Year	Technique	Primary objectives	Methodological implication
[4]	2013	GA load balancing	Makespan, response time	GA is suitable for discrete task-to-resource search.
[6]	2015	Nested PSO	Time, energy	Multi-objective evaluation is preferable to a single metric.
[7]	2018	PSO + hill climbing	Makespan	Local refinement can improve population-based search.
[8]	2018	Hybrid GA-PSO	Time, cost, load balance	GA and PSO have complementary exploration/exploitation behavior.
[9]	2019	GA + ABC + heuristic	Makespan, cost	Hybrid multi-objective workflow scheduling is feasible.
[10]	2019	Modified PSO + fuzzy operators + GA	Makespan, efficiency	Resource attributes can be embedded in fitness evaluation.

[11]	2020	Static + dynamic hybrid scheduling	Cost, execution time	A hybrid policy can adapt to heterogeneous cloud conditions.
------	------	------------------------------------	----------------------	--

Methodology

1. System and Allocation Model

The cloud model contains a broker, a set of independent tasks $T = \{t_1, t_2, \dots, t_n\}$, and a set of heterogeneous virtual machines $V = \{v_1, v_2, \dots, v_m\}$. Each task has a computational length expressed in million instructions (MI). Each VM is characterized by processing capacity in MIPS and a cost rate per second. A candidate solution is an integer vector X of length n in which X_i identifies the VM assigned to task t_i . Every task must be assigned to exactly one VM. The execution time of task i on VM j is estimated as $ET_{ij} = \text{Length}_i / \text{MIPS}_j$. VM completion time is the sum of the execution times of the tasks assigned to that VM.

2. Multi-Objective Fitness Function

Four normalized criteria are minimized: makespan, execution cost, utilization penalty, and load imbalance. Makespan is the maximum VM completion time. Cost is the sum of task execution time multiplied by the VM cost rate. Resource utilization is the ratio of total busy time to the product of the number of VMs and makespan; therefore, $(1 - \text{utilization})$ is minimized. Load imbalance is defined as $(\text{maximum VM load} - \text{minimum VM load}) / \text{mean VM load}$. The composite fitness used in the proof-of-concept simulation is: $F = 0.45M + 0.20C + 0.20(1-U) + 0.15I$, where M , C , U , and I are normalized measures. Higher weight is assigned to makespan because completion time is normally the dominant scheduling objective, while the remaining terms prevent a schedule from becoming fast only by overusing costly or unbalanced resources.

3. Hybrid GA-PSO-LR Procedure

Step 1 - Initialization: create a population of random task-to-VM schedules and inject one greedy earliest-completion schedule to improve initial feasibility. *Step 2 - GA exploration:* evaluate schedules with the multi-objective fitness, retain elite chromosomes, apply tournament selection, uniform crossover, and low-rate mutation for a fixed number of generations. *Step 3 - PSO refinement:* convert the best GA schedules to particle positions, update velocity using inertia, cognitive, and social components, round positions to valid VM indices, and retain personal/global best solutions. *Step 4 - Load-aware repair:* identify the VM with the highest accumulated execution time, test migration of

selected tasks to alternative VMs, and accept a move only when the composite fitness improves. *Step 5 - Termination:* return the best allocation after the iteration limit and calculate all performance metrics. This architecture follows the exploration-exploitation logic supported by hybrid scheduling literature [7]-[10].

4. Constraint Handling and Computational Complexity

Feasibility is preserved by design because every chromosome gene and every rounded particle position contains exactly one valid VM identifier. If additional constraints are introduced, such as RAM capacity, deadline, or private-cloud placement, infeasible schedules can be assigned a penalty proportional to the magnitude of the violation. For n tasks, m VMs, population size P , and G GA generations, the dominant GA evaluation cost is approximately $O(GPn)$. The PSO phase has similar order $O(SKn)$, where S is the number of particles and K is the number of PSO iterations. The repair phase evaluates only a bounded subset of migrations and is therefore controlled by an explicit move budget. The hybrid method is consequently more computationally expensive than Round Robin, but scheduling is performed at the broker level before or during task dispatch; the additional optimization time can be justified when the workload is sufficiently large or expensive to execute.

5. Validation and Reproducibility Protocol

A fair comparison requires common inputs and controlled randomness. All algorithms therefore receive the same task lengths, VM capacities, and VM prices. Randomized algorithms should normally be repeated over multiple seeds and reported with mean, standard deviation, and a non-parametric significance test when enough runs are available. Gao et al. [9], for example, used statistical validation for multi-objective workflow scheduling. In this compact paper a single fixed seed is retained so that the complete numerical example can be exactly reproduced within the page limit. The recommended full experiment is at least 20-30 independent runs per workload size, followed by Wilcoxon signed-rank or Friedman testing and effect-size reporting. This separation between the compact proof-of-concept and the recommended full validation prevents accidental overstatement of a single run.

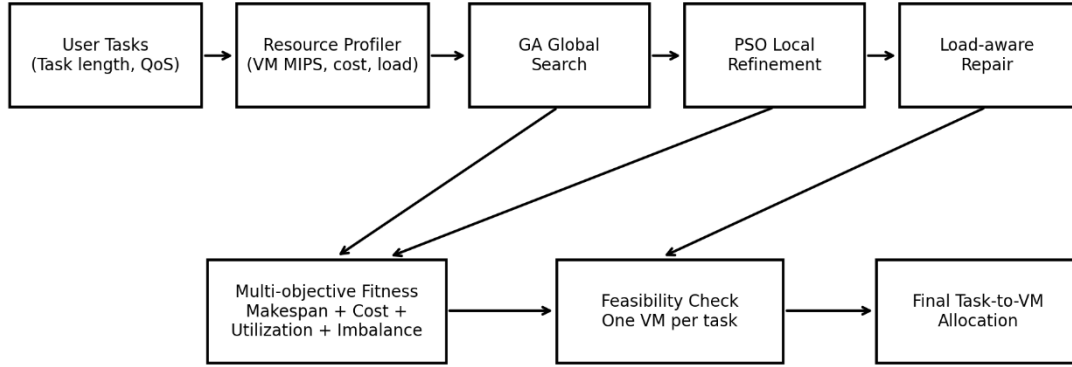


Figure 1. Proposed HGA-PSO-LR resource allocation workflow.

6. Experimental Setup

The validation is a fixed-seed proof-of-concept simulation designed to show how the methodology can be evaluated before deployment in a full CloudSim implementation. The random seed is 42. Eight heterogeneous VMs are used, with processing capacities between 1000 and 3500 MIPS and cost rates between 0.010 and 0.030 monetary units per second. Eighty independent tasks are generated with lengths uniformly sampled between 5,000 and

50,000 MI. GA uses a population of 40 schedules for 30 generations, a mutation probability of 0.035, elitism, and tournament selection. Standalone PSO uses 35 particles and 50 iterations. The hybrid phase refines the GA solution with 50 particles for 100 iterations; the inertia weight decreases linearly from 0.9 toward 0.4 and both acceleration coefficients are 1.6. The load-aware repair phase performs bounded improving moves. The same task and VM set is used for all algorithms.

Table 2. Proof-of-concept simulation configuration

Parameter	Value	Purpose
Tasks	80; 5,000-50,000 MI	Heterogeneous workload
Virtual machines	8; 1000-3500 MIPS	Heterogeneous compute capacity
VM cost rate	0.010-0.030 units/s	Cost-aware scheduling
GA	40 population; 30 generations; pm=0.035	Global exploration
PSO	35 particles; 50 iterations	Standalone comparison
Hybrid PSO phase	50 particles; 100 iterations	Local refinement of GA solution
Fitness weights	0.45, 0.20, 0.20, 0.15	Time, cost, utilization, imbalance
Random seed	42	Reproducibility

Evaluation uses five metrics: makespan (seconds), execution cost (monetary units), average VM utilization (%), load-imbalance index, and throughput (tasks per second). Round Robin and greedy earliest-completion scheduling provide deterministic baselines. GA and PSO isolate the contribution of the component optimizers, while HGA-PSO-LR tests the complete hybrid method. Because the experiment is a compact simulation, the numerical results should be interpreted as reproducible evidence for the proposed procedure, not as a substitute for benchmarking on production traces such as Google or PlanetLab.

7. Recommended Sensitivity Scenarios

For a complete experimental campaign, the workload should be varied along at least three dimensions. Workload size can be increased from

tens to several hundred or thousand tasks to test scalability. Heterogeneity can be changed by narrowing or widening the VM MIPS range, which reveals whether the scheduler benefits mainly from obvious fast-machine choices or from genuinely balanced allocation. Finally, the objective weights can be varied to produce time-prioritized, cost-prioritized, and balanced policies. A robust hybrid method should not depend on one narrow setting; instead, it should retain competitive performance across these scenarios. Additional stress tests can introduce bursty task arrival, VM failure, bandwidth delay, and a deadline penalty. Such sensitivity analysis would make the method suitable for direct comparison with energy-aware consolidation [3], workflow provisioning [5], and hybrid-cloud scheduling [11].

Results

Table 3 reports the simulation output. Round Robin produces the largest makespan and the lowest utilization because it does not account for VM speed. Greedy scheduling substantially improves both measures by considering predicted completion time. PSO and GA further improve the balance of work, with GA performing better under the selected parameters. The complete HGA-PSO-LR method achieves the best

composite score of 0.451, a makespan of 137.86 s, utilization of 98.64%, and the smallest imbalance index (0.031). Relative to Round Robin, the hybrid schedule reduces makespan by approximately 44.1% and raises utilization by about 35.1 percentage points. Relative to standalone GA, the improvement is smaller but consistent: makespan decreases by about 2.0%, utilization increases by about 2.0 percentage points, and imbalance falls from 0.069 to 0.031.

Table 3. Comparative performance of allocation strategies

Algorithm	Makespan (s)	Cost	Utilization (%)	Imbalance	Objective
Round Robin	246.54	19.783	63.54	1.048	0.845
Greedy	144.84	19.575	92.52	0.168	0.492
PSO	144.84	19.602	94.01	0.142	0.486
GA	140.72	19.589	96.64	0.069	0.465
Hybrid GA-PSO-LR	137.86	19.590	98.64	0.031	0.451

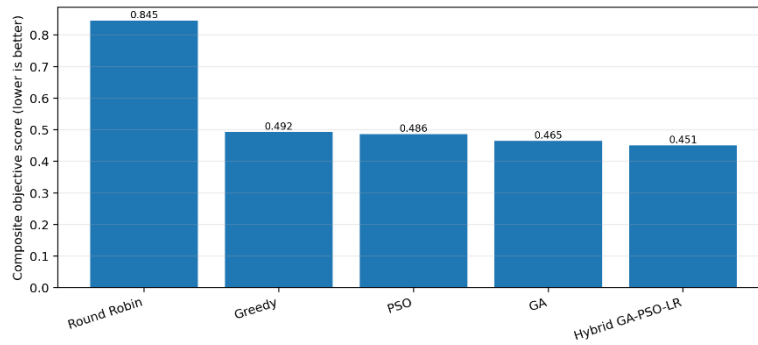


Figure 2. Composite objective score for the evaluated schedulers (lower is better).

The relatively small cost variation in Table 3 is expected because all algorithms execute the same total workload and the VM price range is narrow; improvements therefore come mainly from shorter makespan and better distribution of busy time. This illustrates an important methodological point: weights and cost models must be selected to reflect the deployment context. The literature similarly shows that hybrid methods are most useful when multiple conflicting objectives are present. Manasrah and Ba Ali [8] reported that GA-PSO can improve workflow load balance, while Gao et al. [9] and Mansouri et al. [10] demonstrated that hybrid search can address multi-objective scheduling and heterogeneous resource characteristics. The present simulation is consistent with those methodological trends, although its numerical values are specific to the stated configuration.

8. Interpretation of the Hybrid Gain

The result pattern also explains why hybridization is useful. The greedy scheduler quickly reaches a strong solution because it directly considers current VM completion time, but it has no mechanism to revisit early

assignments. PSO improves the balance of the solution population, while GA obtains a stronger discrete assignment by recombining promising schedules. The final hybrid improves further because PSO explores the neighborhood of the best GA allocation and the repair operator accepts only moves that lower the complete fitness. The gain over GA is much smaller than the gain over Round Robin, which is expected: a strong standalone metaheuristic already captures most of the available improvement. In a larger or more constrained search space, the value of the hybrid should be assessed statistically rather than inferred from a single absolute percentage. This is why the proposed methodology emphasizes repeated experiments and effect sizes for future work.

Conclusion and Discussion

This paper developed a step-by-step methodology for cloud resource allocation using a hybrid optimization technique. The proposed HGA-PSO-LR framework combines GA-based global exploration, PSO-based refinement, and a load-aware repair operator under a multi-objective fitness function. A fixed-seed simulation

shows that the hybrid schedule can improve makespan, utilization, and load balance compared with Round Robin, greedy scheduling, standalone PSO, and standalone GA. The main contribution is the methodology itself: a clearly encoded allocation vector, explicit objective terms, controlled baselines, and reproducible evaluation metrics that can be transferred to CloudSim [2] or another cloud simulator. The results should be interpreted with appropriate limitations. Independent tasks, static VM capacities, and simple time-proportional prices do not capture network delay, VM boot time, migration overhead, failures, or dynamic workload arrival. Energy consumption is represented indirectly through utilization and is not modeled at the host-power level described in consolidation studies such as Beloglazov and Buyya [3]. Future work should therefore test the hybrid method using scientific DAG workflows, real workload traces, dynamic VM scaling, SLA penalties, and host-level energy models. Hybrid-cloud constraints can also be incorporated using private/public placement and data-locality concepts similar to Pasdar et al. [11].

References

- R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599-616, 2009. DOI: 10.1016/j.future.2008.12.001.
- 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 and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397-1420, 2012. DOI: 10.1002/cpe.1867.
- K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, and S. Dam, "A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing," *Procedia Technology*, vol. 10, pp. 340-347, 2013. DOI: 10.1016/j.protcy.2013.12.369.
- M. A. Rodriguez and R. Buyya, "Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 222-235, 2014. DOI: 10.1109/TCC.2014.2314655.
- R. K. Jena, "Multi Objective Task Scheduling in Cloud Environment Using Nested PSO Framework," *Procedia Computer Science*, vol. 57, pp. 1219-1227, 2015. DOI: 10.1016/j.procs.2015.07.419.
- N. Dordaie and N. J. Navimipour, "A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments," *ICT Express*, vol. 4, no. 4, pp. 199-202, 2018. DOI: 10.1016/j.icte.2017.08.001.
- M. Manasrah and H. Ba Ali, "Workflow Scheduling Using Hybrid GA-PSO Algorithm in Cloud Computing," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 1934784, 16 pages, 2018. DOI: 10.1155/2018/1934784.
- Y. Gao, S. Zhang, and J. Zhou, "A Hybrid Algorithm for Multi-Objective Scientific Workflow Scheduling in IaaS Cloud," *IEEE Access*, vol. 7, pp. 125783-125795, 2019. DOI: 10.1109/ACCESS.2019.2939294.
- N. Mansouri, B. M. H. Zade, and M. M. Javidi, "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory," *Computers & Industrial Engineering*, vol. 130, pp. 597-633, 2019. DOI: 10.1016/j.cie.2019.03.006.
- Pasdar, Y. C. Lee, and K. Almi'ani, "Hybrid scheduling for scientific workflows on hybrid clouds," *Computer Networks*, vol. 181, Article 107438, 2020. DOI: 10.1016/j.comnet.2020.107438.