International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(13s) 2026 pp. 683-687
https://doi.org/10.70917/ijcisim-2026-3370
Article

Swarm Intelligence Algorithms for Resource Optimization: A Comparative Experimental Analysis and Hybrid Approach

Mandeep Kaur

Department of Electronics and Communication Engineering, Punjabi University, Patiala, India (147002)
ermandeep0@gmail.com

Abstract: Resource optimization is an essential factor in many areas, such as manufacturing, logistics, cloud computing, wireless sensor network and energy systems. Conventional optimization methods can be impractical when utilized in large, nonlinear, or dynamic optimization problems. Swarm Intelligence (SI) algorithms are based on collective behavior in nature, which provide alternatives, which are decentralized, flexible, and efficient. This paper provides a detailed experimental analysis of three traditional swarm algorithms Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Artificial Bee Colony (ABC) to an allocation of resources and scheduling. The performance of each algorithm was determined using a benchmark problem, which modeled the problem of minimizing costs, resource utilization, convergence behavior, the complexity of computation and robustness to dynamic problems. Furthermore, a hybrid PSO-ACO algorithm is suggested with the utilization of the fast convergence of PSO and the global search of ACO. The observed experimental results can confirm that the hybrid model is more effective in cost optimization and utilization by 12-18 percent than other algorithms. The paper ends by projecting the research directions in the future such as adaptive hybridization, multi-objective optimization, and real-world applications.
Keywords: Swarm Intelligence, Resource Optimization, Particle Swarm Optimization, Ant Colony Optimization, Artificial Bee Colony, Hybrid Algorithms, Metaheuristics and Dynamic Resource Allocation.

1. Introduction

Logistic, energy, computing and manufacturing systems in modern systems are increasingly characterized by large numbers of heterogeneous and interdependent resources. The operational efficiency, cost and environmental sustainability are directly influenced by the successful allocation and planning of these resources. Resource optimization problems tend to have the following characteristics:

- High dimensionality: Good resources and tasks make it possible to have a combinatorial explosion of the possible allocations.
- These are nonlinearity and nonconvexity: The functions of cost may include nonlinear interactions, rendering the conventional gradient-based methods inappropriate.
- Dynamic environments: The real world systems represent dynamic demand, supply of resources, and operational constraints.
- Multiple conflicting aims: Trade-offs may be amidst cost, time, utilization, and quality.

Linear Programming (LP), Integer Programming (IP), and Dynamic Programming (DP), are classical methods of optimization, and are effective at small-scale or structured problems, but are poorly scaled to dynamic and nonlinear problems. Heuristic and metaheuristic algorithms offer useful alternatives to the precise algorithm by compromising speed and flexibility [1]. The most common of them are Swarm Intelligence (SI) algorithms, which are inspired by real-world collective phenomenon, such as bird flocking, ant foraging, or bee colony dynamics, and have proven to be very effective in solving large-scale optimization problems.

This paper is an experimental study of three popular SI algorithms: PSO, ACO and ABC and a hybrid PSO-ACO based approach to optimizing our resources. Compared to most comparative studies which only look at a static setting, we consider dynamic settings, sensitivity of algorithm parameters and the benefits of hybridization and hence can be generally applicable to real-world applications including the dynamic allocation of cloud resources, smart grid energy distribution, and planning of industrial production process.

2. Literature Review

Swarm Intelligence is related to the early papers of Particle Swarm Optimization by Kennedy and Eberhart (1995) [3], Ant System by Dorigo (1996)[1] and Artificial Bee Colony by Karaboga (2005) [2]. These algorithms have been expanded and used in other areas in the last twenty year. PSO is a mathematical optimization algorithm known to be simple to use and converge fast, and it has been applied in continuous optimization, such as power load dispatch, parameter tuning in neural networks, and scheduling tasks in clouds [4,5]. PSO is however susceptible to premature convergence in multimodal search spaces.ACO was created to address discrete optimization, including the Traveling Salesman Problem (TSP), ACO is effective in combinatorial optimization, including vehicle routing, job shop scheduling, and resource allocation in grid computers. It has a pheromone updating mechanism that allows exploration at the cost of increased computation time. ABC algorithm models the behaviour of honeybees during foraging in terms of exploration and exploitation via used, observer and scout bees. ABC is especially applicable in dynamic settings [5], when the space of solutions changes with time. The use of hybrid swarm intelligence algorithms has become popular lately. To overcome the individual limitations, researchers have come up with ACO -PSO hybrids [6], ABC -PSO hybrids, and hybrid evolutionary methods. Indicatively, PSO-ACO hybrids capitalize on the convergence of PSO to perfect the world following ACO venturing into prospective directions. Although these processes have caused development, there are few comparative experimental processes under dynamic situations and most of the hybrid solutions are domain specific. This paper addresses this gap by analytically examining and integrating PSO and ACO towards a generalized resource optimization model.

3. Theoretical Framework

Swarm intelligence algorithms share a common framework: A population of agents explores the search space, updates positions based on collective rules, and iteratively improves solutions.

Let the resource optimization problem be defined as:

$$\min \ F(\mathbf{x}) = \sum_{r=1}^{R} \text{Urep} \sum_{t=1}^{T} \text{u}\, C_{rt} x_{rt} - \lambda U(\mathbf{x}) \qquad (1)$$

In which xrt denotes variables of resource allocation (binary or continuous), crt are costs related to resource allocation, and U_{rep} denotes the utilization of resources. Capacity limits and task satisfaction are imposed by constraints[10].

Both algorithms use their own mechanism of updating solutions:

PSO: Simulation of social learning is done by velocity-position updates.

ACO: Searching by trails of probabilistic pheromones and by heuristic visibility.

ABC: The fitness (quality of food sources) regulates the behaviors of bees to forage and exploit.

Hybridization is a synthesis of the path exploration of ACO and convergence of PSO which is velocity-driven and results in a balance of the global and local search.

4. Methodology

4.1 Algorithms Implemented

PSO: using inertia weight w=0.7, cognitive/social c1=c2=1.5 and population size = 40.

ACO: 50 ants with pheromone evaporation rate ρ=0.1, and α =1, 2, and heuristic visibility that is applied to task-resource mapping.

ABC: There are 50 bees, 50 people around, and the limit parameter of scout activation = 100.

Hybrid PSOACO: ACO gets pheromone matrix as the starting point of proposal of solutions: initial solutions are constructed; best solutions are inoculated as seeds of the particles of PSO to be refined through iteration.

4.2 Experimental Setup

Resources (R): 50, Tasks (T): 100.

Dynamic conditions: task demand varies in every 10 iteration with the variations of ±10-20.

Evaluation Metrics:

- Cost minimization
- Resource utilization (%)
- Convergence iterations
- Computation time
- Perturbative stability Stability to dynamic perturbations.

Tests were performed with Python 3.11 in Intel i7, 16 GB of memory. The number of runs in each experiment averaged 30 to guarantee the reliability of a statistical experiment.

4.3. Experimental Dataset and Implementation

To ensure reproducibility and transparency of the experiments, a synthetic dataset was generated to simulate realistic resource optimization scenarios.

Dataset:

- 50 resources with capacities ranging between 60 and 160 units.
- 100 tasks with random demand (1–8 units) and priority weights between 0.5 and 1.5.
- A 50 × 100 resource–task cost matrix with costs uniformly sampled between 1.0 and 10.0.

The dataset is provided in three CSV files:

- resources.csv – resource IDs and capacities
- tasks.csv – task IDs, demands, and priorities
- cost_matrix.csv – cost values for each resource–task pair

4.4Implementation All algorithms were implemented in Python 3.11. The baseline greedy algorithm was used to establish initial performance levels. Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), and a PSO–ACO Hybrid algorithm were then applied to the dataset.Algorithm parameter settings are stored in algorithm_params.json. The main experiment script si_resource_opt_code.py loads the dataset, runs the algorithms, generates convergence plots, and produces summarized performance tables.

Availability: The entire experimental package is provided in the supplementary file swarm_resource_opt_package.zip, which contains all datasets, parameter files, Python code, figures, and result summaries to enable full replication.

5. Discussion and Results of the Experiments

5.1 Convergence Analysis

PSO improved with an average of 45 iterations and sometimes stagnated. ACO was less aggressive (in terms of its exploration) but converged more gradually (~95 iterations). The convergence rate of ABC was moderate (around 70 iterations). The hybrid PSO-ACO converged after 60 iterations and cost was lower.

5.2 Static and Dynamic Environments

In stagnant environments, the cost was cheapest with ACO, whereas PSO was quicker.ABC was more adaptable in dynamic situations, as a result of scouts. The hybrid ensured almost optimal performance during varying demands.

5.3 Parameter Sensitivity

Optimizing ACO by increasing its β parameter improved exploitation at the expense of convergence. The weight of PSO when it comes to exploration was a major influence. The hybrid had less sensitivity to tuning of the parameters implying strength.

5.4 Scalability

When R=200 and T=500, hybrid PSOACO scaled linearly and was still able to optimize, but ACO was computationally costly and PSO had poor quality.

5.5 Algorithmic Complexity

PSO: O(NP×D)

ACO: O(Nants×D^2)

ABC: O(Nbees×D)

Hybrid: ACO init + PSO exploit ⋄ O(D 2 + NPxD) but gives better results.

6. Conclusion

The article was a comparative experiment on PSO, ACO, and ABC and a hybrid between PSO and ACO on the aspect of resource optimization in both the dynamic and the non-dynamic environment. Key findings includes that PSO is effective to the continuous problems which can be easily converged within the shortest time; but it is vulnerable to local minimum.ACO can be applied in discrete optimization where the quality of solution is of high standard.ABC is a better product in the aspects of dynamism due to exploration strategies. The hybrid PSO-ACO was the best one in the aspect of finding a balance between exploration and exploitation and better cost and utilization of 12-18 percent. In future, the work can be extended on Phase reinforcement learning using ACO/PSO Phase hybridization. Further, Multi-objective optimization can be done in term of the inclusion of the sustainability, energy usage or latency in the cost. Furthermore, Parallel, distributed large-scale cloud/IoT implementations and Smart grids, cloud data centers and supply chain networks on real-life testing can be done using optimization algorithms.

Declaration of competing interest

Author declares no conflicts of interest.

References

1. Dorigo, M., & Gambardella, L. M. (1997). Ant colonies for the traveling salesman problem. BioSystems, 43(2), 73–81.
2. Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical Report-TR06, Erciyes University.
3. Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks, 4, 1942–1948.
4. Li, X., & Zhao, Q. (2023). Hybrid metaheuristic optimization for resource scheduling in cloud computing. Journal of Systems and Software, 203, 111688.
5. Zhang, Y., Wang, H., & Chen, X. (2021). Hybrid swarm intelligence algorithms for energy-efficient task allocation. Applied Soft Computing, 111, 107696.
6. Liu, J., Zhang, T., & Chen, P. (2022). ACO–PSO hybrid algorithm for logistics resource scheduling. Expert Systems with Applications, 188, 116019.
7. Xue, Y., & Zhang, Z. (2019). Adaptive PSO for large-scale resource allocation. IEEE Transactions on Evolutionary Computation, 23(2), 295–308.
8. S.Carlo, and O. Tokunbo,(2015) “FPGA implementation of LMS-based FIR adaptive filter for Real time Digital Signal Processing Applications”. Proc. IEEE International Conf. Digital Signal Processing.
9. N.S. Johannes and L. Boris,(2017) “Optimal feedforward preview control by FIR filters,” IFAC Papers OnLine , 5115–5120.
10. K.K. Parhi,(2007) VLSI Digital Signal Processing Systems Design and implementation. Interscience.
11. H.R. Lee, C.W.Jen, C.M.Liu, (1996)“New hardware-efficient architecture for programmable FIR filter,” IEEE Trans. Circuits Sys., 43, 637–644.
12. K. Azadet and C.K. Nicole,(1998) “Low-power equalizer architectures for high-speed modems,” IEEE Comm. Mag.

13. K. Khoo, Z. Yu and N.W. Alan,(2001) “Design of optimal hybrid form FIR filter,” Proc. IEEE sym.
14. H. Samueli,(1989) “An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients,” IEEE Trans. Cir. Syst., pp. 1044–47.
15. A. Chandra and S. Chattopadhyay, (2016)“Design of hardware efficient FIR filter: A review of the state-of-the-art approaches,” Engineering Science and Technology, an International Journal, 19(1), pp. 212–226.
16. N. Maheshwari and S. Sapatnekar,(1998) “Efficient retiming of large circuits,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., 6(1) 74– 83’
17. C.E. Leiserson and J.B. Saxe,(1991) “Retiming synchronous circuitry,” Algorithmica, 6, 5–35.. [11] J.R. Jiang and R.K. Brayton,(2006) “Retiming and resynthesis: a complexity perspective, “ IEEE Trans. Computer-Aided Design of Integr. Circuits Syst.,25(12), 2674–86.
18. D. Yagain and K.A. Vijaya,(2014) “Design of synthesizable, retimed digital filters using FPGA based path solvers with MCM approach: comparison and CAD tool,’ VLSI Design, Hindawi Pub. Corp., Article ID 280701.
19. P.K. Meher,(2016) “On efficient retiming of fixed-point circuits,” IEEE Trans. VLSI Syst. 4(4). [14] S. Thakral, D. Goswami and R.S. Sharma,(2016) “Design and implementation of a high speed digital FIR filter using unfolding,” Proc. IEEE 7th Int. Con., Bikaner, India.
20. C. Swati and J. Himanshu,(2015) “FIR filter designing using wallace multiplier,” International Journal of Engineering and Technical Research, 3(6), 276-78.
21. P. Pramod and T.K. Shahana,(2020) “Efficient modular hybrid adders and Radix-4 booth multipliers for DSP applications’, Microelectronics Journal, Print ISSN 0026-2692, https://doi.org/10.1016/j.mejo.2020.104701
22. S.N. Rai, B.S.P. Shree and Y.P. Meghana,(2018) “Design and implementation of 16 tap FIR filter for DSP applications,” Proc. Sec. Int. Conf. Adv. in Electron. Comp. and Comm., Bengaluru, India,pp. 9-10.
23. L. Ting, R. Woods and C. F. N. Cowan,(2005) “Virtex FPGA Implementation of a pipelined adaptive LMS predictor for electronic support measures receivers,” IEEE Trans. Very Large Scale Int.Syst., 13(1), 86-95.
24. P.K. Meher and S.Y.Park,(2014) “Critical-path analysis and low-complexity implementation of the LMS adaptive algorithm,” IEEE Trans. Circuits Syst.-I: Reg. Papers, 61(3), pp.778-788.