Optimising workflow execution for energy consumption and performance

Mehul Warade[1], Kevin Lee[1], Chathurika Ranaweera[1], and Jean-Guy Schneider[2]
[1]*School of Information Technology, Deakin University, Geelong VIC 3220, Australia.*
[2]*Faculty of Information Technology, Monash University, Clayton VIC 3800, Australia.*
mehul.warade@research.deakin.edu.au

***Abstract*—Optimizing computation for energy consumption and performance requires consideration of the following three key factors: the energy consumption of the resources used; the performance of the workflow; and the time it takes to complete the computation. The increasing complexity of today's computing systems makes it essential to ensure that the resources allocated to a workload are used in the most effective and cost-efficient manner. In this paper, the importance of optimal scheduling solutions for scientific workflow computation is presented. A generic framework is proposed that takes into consideration the user constraints, performance, and factors that affect the energy consumption of different computations to develop an optimal schedule for workflow execution. The aim of the framework will be to improve the energy consumption and performance of the computation. The study also aims to motivate research in energy-efficient optimized scheduling.**

***Index Terms*—Energy, Optimize, Workflow, Heuristics**

I. Introduction

Scientific workflows are an increasingly popular way to streamline and automate complex scientific processes. By combining a series of tasks into a single workflow, scientists can save time and energy while ensuring that all steps are completed in the correct order. Scientific workflows are used to automate tedious manual processes and help researchers to focus on more important tasks such as analysis and interpretation of results. This makes it easier to repeat experiments, collaborate with colleagues, and share results with the scientific community.

A scientific workflow is a set of instructions that define how data is collected, processed, and analysed [1]. This may include steps such as retrieving data from a database, cleaning and formatting the data, executing statistical tests, and producing visualizations. Workflows can be executed on a single computer, or distributed across multiple computers and networks.

Large computation capacity is needed to meet the increasing demand for computation. These data centers face multiple challenges such as energy consumption (for computation, cooling, and power delivery systems), execution time, heat generation and consequently cooling infrastructure, CO_2 emission, and high operational costs. Due to the scale of the installed systems and their resource usage, a small improvement in any of the given challenges can have huge environmental and financial benefits.

In the past, researchers almost exclusively targeted the performance of computation. With an increasing global energy crisis [2], there is a need to consider the energy consumption as well while optimizing the computation, resulting in a multi-objective optimization problem where energy consumption and run-time performance need to be optimized according to users' needs [3]–[5]. Effective optimization in energy consumption and performance can heavily reduce the costs associated with wasted resources and can help maximize performance by streamlining processes and eliminating inefficiencies.

When optimizing for energy consumption and performance in workflow computation, it is important to consider the different elements that can affect the energy consumption and performance of the computation [6]. This includes the hardware and software used, the algorithms and data structures employed, and the overall architecture of the system. It is also important to consider the type of workloads that will be processed, as this can have a significant impact on the cost and performance of the computation.

The paper aims to motivate research in different optimizing approaches for developing effective scheduling solutions based on energy consumption and performance for scientific workflow computations. The current state-of-the-art techniques used to find the optimal scheduling solution for energy consumption and performance are presented. It is anticipated that a system built using optimization techniques to balance performance and energy consumption of computation will improve the costs of performing the computation and assist in better utilization of resources.

The key contributions of this paper are as follows: (i) the current state of energy-aware computation optimization techniques; (ii) a survey of potential optimization techniques suitable for this problem; and (iii) a conceptual framework for optimal scheduling of workflow execution for energy-consumption and/or performance.

The remainder of this paper is as follows: Section II provides an overview of scientific workflow and related work in the domain of optimization that focuses on energy consumption and the performance of computation. Section III presents a survey of different optimization techniques that can optimally schedule scientific workflow execution. Section IV presents a conceptual framework that develops optimal scheduling solutions for scientific workflow computation. Finally, Section V provides conclusions and future work.

II. Background

Scientific workflows are increasingly being used to automate complex processes and data analysis, such as data mining, machine learning, and natural language processing. Understanding how to optimize scientific workflows can help to reduce errors, increase the reliability of the results and improve the performance or cost of computation. This section provides a background on (i) scientific workflows, (ii) optimization techniques focusing on energy consumption, (iii) optimization techniques focusing on performance, and (iv) optimization techniques focusing on both the cost and performance of computations.

A. Scientific Workflows

Scientific workflows are a series of steps or tasks that are executed in a specific order to accomplish a scientific or research goal. These steps may include data collection, data processing, analysis, and visualization. Scientific workflows can be used in a variety of fields, including genomics, bioinformatics, and climate modeling. They allow for the automation of repetitive or complex tasks, and can also facilitate collaboration and reproducibility in scientific research. A workflow consists of a number of separate jobs that can each be executed separately. The tasks in a workflow are often interdependent, meaning that the output of one task is used as input for another task. Some examples of scientific workflows include Montage Workflow and Bioinformatics workflow [1].

As the scale and complexity of these workflows continue to grow, so too does the demand for computational resources. This increase in demand for resources can lead to high energy consumption and poor performance, which can negatively impact the overall efficiency and effectiveness of the scientific research process [7].

To address the issue of ever-increasing demand for compute resources, several optimization techniques have been developed to find the optimal scheduling solution that takes into consideration the energy consumption and performance of computation. These techniques can be broadly categorized into two groups: those that focus on reducing energy consumption and those that focus on improving performance.

B. Optimising for energy consumption

A mathematical model is developed to reduce the energy consumption of machines that takes into consideration different attributes of machines such as idle time, launch time for computation, and time it takes to power off and on [8]. A genetic algorithm is used to obtain 'near-optimal' solutions to achieve the lowest energy consumption using different attributes [8].

An energy loss optimization scheduling modeling method based on the multi-objective fuzzy algorithm has been developed that focuses on the energy cost and scheduling time of the equipment in an IoT environment [9]. This model focuses on a single-target energy loss problem and searches for idle time of the device and schedules jobs accordingly to make complete use of resources and to reduce the overall energy consumption.

A mixed meta-heuristic algorithm comprising of Whale Optimization Algorithm (WOA) and Evolutionary Algorithm (EA) is developed to improve the energy consumption of IoT devices in a network by considering factors such as workloads, temperatures, remaining energy, and the target energy consumption [10], [11]. The meta-heuristic algorithm is then contrasted with widely popular optimization techniques such as the Artificial Bee Algorithm (ABA), Neural Network (NN), and Simulated Annealing (SA) to evaluate its performance.

A heuristic method to consolidate virtual clusters has been developed that focuses on the energy consumption of servers by analyzing the server duration and utilization. The algorithm aims to consolidate virtual clusters on physical servers to save energy while guaranteeing job service-level agreements (SLAs) [12].

C. Optimising for performance

An optimization model is developed using mixed integer programming focusing on improving performance and minimizing deadline misses. A heuristic approach using Generic Algorithm (GA) is used to develop a scheduling solution using the model in the cloud computing environment [13].

A Hybrid Optimization approach towards Tensor computation on heterogeneous systems has been undertaken that performs schedule exploration and optimization to improve the performance of Tensor computation. The approach makes use of different Heuristic methods and Machine learning methods to develop different schedules specific to the hardware requirements [14].

An enhancement to the Whale Optimization Algorithm has been introduced focusing on addressing the issue of resource scheduling for performance in cloud environments. The new algorithm falls under the category of swarm intelligence metaheuristics and has been evaluated with different data, simulations, and other algorithms for its achieved performance and resource scheduling [15].

A cloud scheduling policy based on an Ant Colony Optimization (ACO) has been developed that focuses on minimizing the makespan of the given task set by adapting the scheduling strategy to the changing environment. The algorithm uses a random optimization search approach to allocate and schedule incoming jobs to virtual machines [16]. ACO is also used to reduce the energy, cost, and time of resource scheduling in cloud computation [17].

D. Optimising for performance and energy

A Whale Optimization Algorithm is used to identify the optimal trade-off between performance and energy consumption of mobile cloud computing [18]. The algorithm considers task execution position, task execution sequence,

and operating voltage and frequency in order to maximize performance, efficiency, and operational cost of computation.

A new method of optimization, named Ant Mating Optimization (AMO), has been created to identify the best balance between the time it takes for a system to complete a task and the amount of energy used by resources in fog computing. [19]. Ant Mating Optimization (AMO) improves a task-scheduling algorithm that aims to decrease energy consumption and increase the efficiency of the fog computing platform.

Longest job to fastest processor (LJFP) and minimum completion time (MCT) techniques are used in the improved initialization of particle swarm optimization (PSO) employing heuristic algorithms. For the makespan, performance, and energy consumption measures in cloud infrastructure, the algorithms' performance is assessed [20].

The current research in energy-aware optimization techniques is static and computation specific. These techniques cannot be applied to any generic computation. Also, they do not take advantage of the constantly changing state of the computation and the resources. There is a need for a generic framework that makes use of different optimization techniques to dynamically identify the optimal scheduling solution for a computation based on energy consumption, performance, and resource state. The framework takes into account different combinations of cluster configuration and computation characteristics to find the optimal scheduling solution.

III. Optimization techniques

Optimization techniques are used to find the optimal set of parameters for a model that minimizes the error or maximizes the performance. It is important to understand the different optimization techniques available in order to be able to choose the most appropriate method for a specific problem or dataset. In this Section, a survey of various high-level optimization concepts and their examples is presented. The purpose of this survey is to provide an overview of various optimization techniques that are commonly used.

A. Discrete Optimizations

Discrete optimization is the process of finding the optimal solution from a finite set of discrete options.

1) Linear Programming: This is a method used to optimize a linear objective function, subject to constraints represented by linear equations or inequalities [21].

2) Simplex Method: This is a popular algorithm for solving linear programming problems, which iteratively improves a solution by moving to adjacent vertices of the feasible region [22].

3) Interior-Point Methods: This is a family of algorithms that solve linear programming problems by finding a solution that is close to the central point of the feasible region [23].

4) Gradient Descent: This is an optimization algorithm that iteratively improves a solution by moving in the direction of the negative gradient of the objective function [24].

B. Heuristic Approaches

Heuristics optimization is a method of solving problems by using practical approaches or rules of thumb, rather than relying on a systematic, theoretical solution.

1) Hill Climbing: Hill climbing is an optimization method that begins with an initial solution and gradually makes small adjustments to it in order to enhance the solution. This technique is commonly used to locate a nearby optimal solution [25].

2) Greedy Algorithm: A greedy algorithm constructs a solution piece by piece, always opting for the next piece that provides the most obvious and immediate benefit [26].

3) Beam Search: Beam search is a method of searching for solutions that uses heuristics to examine a small subset of the most likely solutions at each step, rather than exploring all potential solutions [27].

4) Randomized Local Search: Randomized Local Search is a heuristic optimization algorithm that starts with a random initial solution and iteratively applies random changes to it in order to find an improved solution [28].

C. Metaheuristics Optimisation

Metaheuristics optimization is a method of solving complex problems by using high-level procedures, or heuristics, that guide the search for an approximate solution.

1) Genetic Algorithm: A genetic algorithm (GA) is a type of optimization algorithm inspired by natural selection. It uses techniques of reproduction, mutation, and selection to find the best solution for a given problem [29].

2) Grasshopper Algorithm: The Grasshopper algorithm (GHA) is an optimization algorithm that uses the concept of swarm intelligence and it is inspired by the movement of grasshoppers. It uses a population of potential solutions that move randomly, and then converge to the optimal solution through a process of selection and imitation [30].

3) Particle Swarm Optimization (PSO): Particle Swarm Optimization (PSO) is a population-based method that is modeled after the collective behavior of birds and fish such as flocking and schooling. It uses a population of particles that move in the search space to find the optimal solution through a process of exploration and exploitation [31].

4) Cuckoo Search (CS): Cuckoo Search (CS) is an optimization algorithm inspired by the behavior of cuckoos, it can be used to find an optimized scheduling solution by iteratively updating the position of each cuckoo in the population based on the energy consumption and performance of the computation [32].

Meta-heuristic Optimisations include population-based algorithms and are powerful optimization techniques, but do not guarantee finding the global optimum. This indicates that the achieved policy or output might not translate or always work optimally with other workflow executions. To mitigate this problem some variants such as the Artificial Bee Algorithm (ABA) have been developed to find the global optimum [33]. These variants can be used to find generalized

scheduling techniques that might work with different computations and resources.

D. Machine Learning

Machine learning optimization is the process of finding the best set of parameters for a model to minimize its error on a given dataset.

1) Reinforcement Learning: Reinforcement learning can optimize scheduling decisions by gaining insight from the system's feedback. By analyzing the energy consumption and performance of the computation, it can continuously improve the scheduling decisions over time [34], [35].

2) Q-learning: Q-learning is a form of reinforcement learning that can be utilized to identify the best scheduling decisions by studying the energy consumption and performance of the computation [34], [35].

3) Neural Networks: Neural networks can be utilized to depict the correlation between scheduling decisions and energy consumption and performance of computation. They can be trained on past data to anticipate energy consumption and performance of new scheduling decisions [34], [35].

4) Decision Trees: Decision trees can be used to depict the association between scheduling decisions and energy consumption and the performance of computation. Using historical data can enhance the accuracy of Decision Tree predictions [34], [35].

E. Multi-Objective optimization

Multi-objective optimization techniques are used to optimize scheduling solutions by taking into account multiple objectives, including energy consumption, accuracy, and performance. These techniques can find a set of optimal solutions that balance the various objectives and make trade-offs between them.

1) Non-dominated Sorting Genetic Algorithm (NSGA): NSGA is a genetic algorithm that is capable of dealing with multiple objectives. It can be used to find optimal solutions with minimal or no trade-offs between the different objectives [36].

2) Multi-objective Particle Swarm Optimization (MOPSO): MOPSO is a particle swarm optimization algorithm that can handle multiple objectives simultaneously and can be used to identify the trade-offs between energy consumption and the performance of the computation [37].

3) Strength Pareto Evolutionary Algorithm (SPEA): SPEA is an evolutionary algorithm based on the concept of Pareto Optimality, which states that a solution is considered optimal when no other solution can improve one of its objectives without compromising at least one of the other objectives [38].

4) Multi-objective Cuckoo Search (MOCS): MOCS combines the Cuckoo Search algorithm with NSGA-II (Non-dominated Sorting Genetic Algorithm II) to handle multiple objectives in optimization [39].

It's important to note these algorithms depend on multiple factors such as the characteristics of the problem, the available resources, and the desired level of accuracy and computational time. Some algorithms may work better for certain types of problems or settings than others, and it may be necessary to experiment with different algorithms to find the best solution.

IV. Optimization-based Scheduling Framework for Energy and Performance Optimization

The previous section presented the different optimization techniques that can be used to identify scheduling solutions that consider the energy consumption and performance of a computation. In this section, a framework that makes use of these optimization techniques and focuses on the energy consumption and performance of workflow computations has been presented.

Figure 1 illustrates several proposed adaptive optimization approaches for workflow computation, with a focus on energy efficiency and performance. There are five main parts of the computation process:

A. User Input

User Input is the input data from the users to the system. Data such as the computation and a list of user constraints or objectives will be provided. The computation can include any scientific workflow and the user constraint includes the objectives that the system will try to achieve such as a fixed cost budget for computation, the accuracy of results, the targeted performance, maximum heat generation threshold, the energy consumption of the resources, etc.

B. Inherent Static Information

This is the information that does not come from the user but is inherently required by the optimization algorithms to make decisions. This information can contain data such as the available computing resources and their configuration, the current costs of consuming energy, the current networking setup of the nodes, any special characteristics of the nodes, etc.

C. Optimization Approaches

This is the core of the system. There are multiple approaches identified and will be implemented in the system. Based on the user input, the complexity of the workflow, user constraints and objectives, and inherent information, a specific optimization technique will be used. This optimization approach will identify the optimal scheduling solution that satisfies the objectives of the system. The original scheduling policy of the workflow will be overwritten by the newly formed optimized schedule. The optimization approach used also needs to consider information such as data dependencies, the complexity of individual jobs, bottlenecks, etc. in the computation.

D. Executing the schedule

The optimal schedule will then be executed on the compute resources and monitored for their execution. Workflows are usually managed by Workflow Management Systems such as Pegasus [40] that uses HTCondor [41] as a scheduler for the workflow.

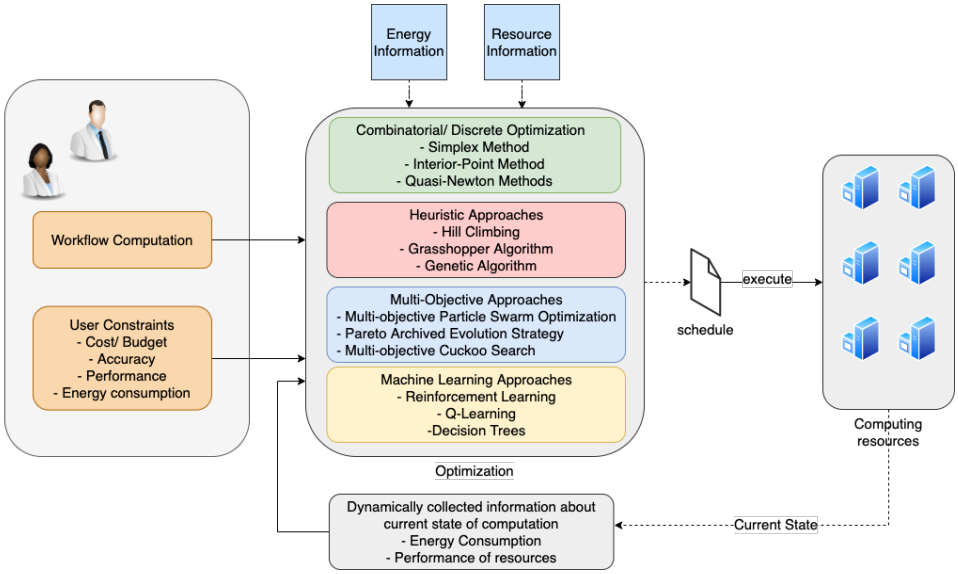

Fig. 1. Proposed adaptive optimization approaches for workflow computation focusing on energy and performance

E. Dynamically collected information

A constant feedback loop from the current execution of the computation will be used to further improve the optimal schedule dynamically. The current execution state of the computation will be constantly monitored for its energy consumption, the performance of individual jobs, the performance of the compute resources, memory usage, the temperature of the resources, etc.

The proposed framework in Figure 1 uses optimization techniques to find an optimal scheduling solution for workflow computation that focuses on energy consumption and performance. This could have several implications on the current domain of workflow scheduling optimizations.

1) The proposed framework could lead to more efficient and cost-effective solutions for large-scale workflow computation. By considering both energy consumption and performance, the framework may be able to balance the trade-offs between these two factors and find a solution that is more optimal than if either factor were considered in isolation.
2) The proposed framework may also lead to improved performance and faster completion of workflow computation tasks, as it takes into account the impact of scheduling decisions on overall performance.
3) The use of multiple optimization techniques in the framework could also make it more robust and adaptable to different types of workflow computations and changing conditions such as cloud computing, distributed computing, and edge computing.
4) The proposed framework could also have implications for industries and organizations that rely heavily on workflow computation as it could potentially lead to significant cost savings in terms of energy consumption and improved performance.

V. Conclusion

As the scale and complexity of scientific workflows continue to grow, it is increasingly important to focus on different optimizing approaches that can be used for finding an optimal scheduling solution for workflow computation, with a specific emphasis on the energy consumption and performance of the computation. The paper has presented a variety of optimization approaches that can find an optimal solution based on different objectives.

This paper also proposed an adaptive system that utilizes different optimization approaches to develop an optimal execution schedule for workflow computation. This proposed system uses the objectives provided by the user and data from different sources to develop a schedule that satisfies the objectives of the system. The proposed system is intended to be a comprehensive solution that can adapt to different scenarios and workloads.

In future work and expansion, the approach and design of the proposed system are to be formalized and implemented. The proposed system will be evaluated by conducting experiments on real-world workflow computation benchmarks. The optimized schedule will be evaluated against the standard execution of the workflow for energy, performance, and achievement of the system objectives. Another approach can include exploring the trade-offs between energy consumption and performance, and to develop new optimization techniques that better balance these competing objectives. It would be interesting to investigate how the proposed system can be applied to other types of workloads and to further evaluate the system in different scenarios.

REFERENCES

[1] M. Warade, J.-G. Schneider, and K. Lee, "Towards energy-aware scheduling of scientific workflows," in *International Conference of Green Energy, Computing and Sustainable Technology (GECOST)*. Curtin University Malaysia, 2022, (Accepted).

[2] M. V. Warade, J.-G. Schneider, and K. Lee, "Measuring the energy and performance of scientific workflows on low-power clusters," *Electronics*, vol. 11, no. 11, p. 1801, 2022.

[3] K. Pradeep and T. P. Jacob, "Cgsa scheduler: A multi-objective-based hybrid approach for task scheduling in cloud environment," *Information Security Journal: A Global Perspective*, vol. 27, no. 2, pp. 77–91, 2018.

[4] T. Prem Jacob and K. Pradeep, "A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization," *Wireless Personal Communications*, vol. 109, no. 1, pp. 315–331, 2019.

[5] A. Al-Maamari and F. A. Omara, "Task scheduling using pso algorithm in cloud computing environments," *International Journal of Grid and Distributed Computing*, vol. 8, no. 5, pp. 245–256, 2015.

[6] R. Garg, M. Mittal, and L. H. Son, "Reliability and energy efficient workflow scheduling in cloud environment," *Cluster Computing*, vol. 22, no. 4, pp. 1283–1297, 2019.

[7] M. V. Warade, J.-G. Schneider, and K. Lee, "Fepac: A framework for evaluating parallel algorithms on cluster architectures," in *2021 Australasian Computer Science Week Multiconference*, 2021, pp. 1–10.

[8] F. Shrouf, J. Ordieres-Meré, A. García-Sánchez, and M. Ortega-Mier, "Optimizing the production scheduling of a single machine to minimize total energy consumption costs," *Journal of Cleaner Production*, vol. 67, pp. 197–207, 2014.

[9] X. Ding and J. Wu, "Study on energy consumption optimization scheduling for internet of things," *IEEE Access*, vol. 7, pp. 70 574–70 583, 2019.

[10] M. A. R. Khan, S. N. Shavkatovich, B. Nagpal, A. Kumar, M. A. Haq, V. J. Tharini, S. Karupusamy, and M. B. Alazzam, "Optimizing hybrid metaheuristic algorithm with cluster head to improve performance metrics on the iot," *Theoretical Computer Science*, 2022.

[11] A. V. Dhumane and R. S. Prasad, "Multi-objective fractional gravitational search algorithm for energy efficient routing in iot," *Wireless networks*, vol. 25, no. 1, pp. 399–413, 2019.

[12] F. Teng, L. Yu, T. Li, D. Deng, and F. Magoulès, "Energy efficiency of vm consolidation in iaas clouds," *The Journal of Supercomputing*, vol. 73, no. 2, pp. 782–809, 2017.

[13] R. O. Aburukba, T. Landolsi, and D. Omer, "A heuristic scheduling approach for fog-cloud computing environment with stationary iot devices," *Journal of Network and Computer Applications*, vol. 180, p. 102994, 2021.

[14] S. Zheng, Y. Liang, S. Wang, R. Chen, and K. Sheng, "Flextensor: An automatic schedule exploration and optimization framework for tensor computation on heterogeneous system," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 859–873.

[15] I. 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, p. 4893, 2019.

[16] M. A. Tawfeek, A. El-Sisi, A. E. Keshk, and F. A. Torkey, "Cloud task scheduling based on ant colony optimization," in *2013 8th international conference on computer engineering & systems (ICCES)*. IEEE, 2013, pp. 64–69.

[17] G. Bindu, K. Ramani, and C. S. Bindu, "Optimized resource scheduling using the meta heuristic algorithm in cloud computing," *IAENG International Journal of Computer Science*, vol. 47, no. 3, pp. 360–366, 2020.

[18] H. Peng, W.-S. Wen, M.-L. Tseng, and L.-L. Li, "Joint optimization method for task scheduling time and energy consumption in mobile cloud computing environment," *Applied Soft Computing*, vol. 80, pp. 534–545, 2019.

[19] S. Ghanavati, J. Abawajy, and D. Izadi, "An energy aware task scheduling model using ant-mating optimization in fog computing environment," *IEEE Transactions on Services Computing*, vol. 15, no. 4, pp. 2007–2017, 2020.

[20] S. A. Alsaidy, A. D. Abbood, and M. A. Sahib, "Heuristic initialization of pso task scheduling algorithm in cloud computing," *Journal of King Saud University-Computer and Information Sciences*, 2020.

[21] H. Rommelfanger, R. Hanuscheck, and J. Wolf, "Linear programming with fuzzy objectives," *Fuzzy sets and systems*, vol. 29, no. 1, pp. 31–48, 1989.

[22] J. Yen, J. C. Liao, B. Lee, and D. Randolph, "A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 28, no. 2, pp. 173–191, 1998.

[23] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *Journal of optimization theory and applications*, vol. 99, no. 3, pp. 723–757, 1998.

[24] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[25] S. Chinnasamy, M. Ramachandran, M. Amudha, and K. Ramu, "A review on hill climbing optimization methodology," *Recent Trends in Management and Commerce*, 2022.

[26] Z. He, S. Deng, X. Xu, and J. Z. Huang, "A fast greedy algorithm for outlier mining," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2006, pp. 567–576.

[27] S. Wiseman and A. M. Rush, "Sequence-to-sequence learning as beam-search optimization," *arXiv preprint arXiv:1606.02960*, 2016.

[28] B. Selman, H. A. Kautz, B. Cohen *et al.*, "Local search strategies for satisfiability testing." *Cliques, coloring, and satisfiability*, vol. 26, pp. 521–532, 1993.

[29] S. Sivanandam and S. Deepa, "Genetic algorithm optimization problems," in *Introduction to genetic algorithms*. Springer, 2008, pp. 165–209.

[30] S. Z. Mirjalili, S. Mirjalili, S. Saremi, H. Faris, and I. Aljarah, "Grasshopper optimization algorithm for multi-objective optimization problems," *Applied Intelligence*, vol. 48, no. 4, pp. 805–820, 2018.

[31] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[32] X.-S. Yang and S. Deb, "Cuckoo search: recent advances and applications," *Neural Computing and applications*, vol. 24, no. 1, pp. 169–174, 2014.

[33] D. Karaboga, "Artificial bee colony algorithm," *scholarpedia*, vol. 5, no. 3, p. 6915, 2010.

[34] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE transactions on cybernetics*, vol. 50, no. 8, pp. 3668–3681, 2019.

[35] S. Perrin and T. Roncalli, "Machine learning optimization algorithms & portfolio allocation," *Machine Learning for Asset Management: New Developments and Financial Applications*, pp. 261–328, 2020.

[36] K. Cao, M. Batty, B. Huang, Y. Liu, L. Yu, and J. Chen, "Spatial multi-objective land use optimization: extensions to the non-dominated sorting genetic algorithm-ii," *International Journal of Geographical Information Science*, vol. 25, no. 12, pp. 1949–1969, 2011.

[37] M. Reyes-Sierra, C. C. Coello *et al.*, "Multi-objective particle swarm optimizers: A survey of the state-of-the-art," *International journal of computational intelligence research*, vol. 2, no. 3, pp. 287–308, 2006.

[38] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," *TIK-report*, vol. 103, 2001.

[39] R. Salakhutdinov, S. T. Roweis, and Z. Ghahramani, "Optimization with em and expectation-conjugate-gradient," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 672–679.

[40] E. Deelman *et al.*, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming*, vol. 13, no. 3, pp. 219–237, 2005.

[41] P. Couvares, T. Kosar, A. Roy, J. Weber, and K. Wenger, "Workflow management in condor," in *Workflows for e-Science*. Springer, 2007, pp. 357–375.