IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/45AI/2025/4539825/
Vol. 2025 (20), Article ID 4539825

Concept of Resource Allocation Optimizer in Systems with MLOps Architecture*

Mateusz MILCZAREK and Adrian P. WOZNIAK

Military University of Technology, Warsaw, Poland

Correspondence should be addressed to Mateusz MILCZAREK, Mateusz.Milczarek.wat@gmail.com

* Presented at the 45th IBIMA International Conference, 25-26 June 2025, Cordoba, Spain

Abstract

With the growing importance of MLOps methodologies in the deployment and maintenance processes of machine learning models, there is an increasing demand for efficient and scalable management of computational resources in such environments. Despite numerous studies on tools supporting MLOps, relatively limited attention has been devoted to the optimization of task and component allocation regarding available system resources. This article presents the concept of a method for optimizing component allocation in a system with MLOps architecture, using a mathematical model, a genetic algorithm, a MLOps environment simulator to evaluate solutions, and multi-criteria evaluation methods. The proposed approach incorporates critical infrastructure parameters, including CPU and GPU processing capabilities, memory usage, and network bandwidth, alongside the operational characteristics of MLOps-specific components.

Keywords: optimizer, MLOps, machine learning, resource allocation

Introduction

Today's continuous advances in artificial intelligence and machine learning make it increasingly important to have effective and efficient methods of implementing these systems. In recent years, the MLOps model, which despite its growing popularity, still poses various difficulties in the effective implementation and maintenance of models. In a paper by D. Kreuzberger et al. [2], the authors provide a comprehensive overview of the principles, components, and roles involved in MLOps, emphasizing the need to standardize and automate ML processes to increase their reliability and scalability. This paper presents a conceptual model of a system for optimizing component allocation in MLOps environments. The motivation for undertaking research in this direction is the apparent research gap found in this area during a 2024 literature review of the MLOps topic (Wozniak et al. [1]). The conducted literature review revealed gaps in the literature in allocation of components executing the various stages of processes in systems based on the MLOps architecture, focusing particularly on assigning them to the appropriate computing resources. Previous work mainly focuses on optimizing container scheduling on specific platforms, such as Kubernetes, without proposing to formalize the entire system in the form of a mathematical model. In addition, very few works have treated MLOPs systems holistically, and considered all parameters such as

Cite this Article as: Mateusz MILCZAREK and Adrian P. WOZNIAK, Vol. 2025 (20) "Concept of Resource Allocation Optimizer in Systems with MLOps Architecture " Communications of International Proceedings, Vol. 2025 (20), Article ID 4539825, https://doi.org/10.5171/2025.4539825

processor computing power, RAM, GPU computing power, and network bandwidths taking into account features of such systems such as feature store. Therefore, it was decided to present and subsequently implement a resource allocation method for MLOps systems, based on an optimizer that uses a genetic algorithm, a simulator for solution evaluation, and multi-criteria optimization to identify the optimal allocation. The remainder of this paper is organized as follows. The next chapter contains a literature review of the subject, which lists the works closest to the presented concept, and is supplemented by articles published after the earlier review, up to the present day, in order to identify possible related studies. The third chapter focuses on presenting the concept of the optimizer. It includes a description of the designed mathematical model, followed by a discussion of the optimization mechanism itself. The mathematical model in the optimizer introduces input data in the form of matrices and vectors. It formulates constraints and optimization criteria and defines the vector objective function, which is the basis for optimizing the system. The fourth chapter concludes the article.

Related Works

Numerous studies in literature address tools that support MLOps practices, including continuous integration and deployment (CI/CD) for machine learning models and workflow orchestration frameworks. However, relatively little research has focused on optimizing task and resource allocation within MLOps workflows. This applies both to the infrastructure layer, such as assigning tasks to specific computational nodes, and to planning the order and priority of ML operations to be performed, e.g. training, inference or re-training.

The literature review on the optimization of MLOps processes revealed a lack of studies addressing the holistic resource allocation to all components involved, while simultaneously formalizing the system using a mathematical model. Therefore, the presented concept of the optimization method considers such parameters as computing power of the processors (CPU), memory (RAM), computing power of the graphics cards (GPU), and network bandwidth in inter-server communication, which introduces latency. Additionally, determining optimal horizontal scaling for servers executing identical components is crucial due to the associated network latency between nodes. Although there are papers in the literature devoted to improving task scheduling in MLOps environments, for example, by modifying the container scheduling mechanism in Kubernetes, most of them focus on optimizing individual container tasks. For example, H.-M. Chen et al. [3] proposed replacing the default Kubernetes scheduler with a custom solution tailored for machine learning tasks, but their approach mainly focused on resource allocation at the level of a single container, without accounting for the structure and interdependencies among the components of the complete MLOps pipeline. In the work of I. Syrigos et al. [4] proposed extending the scheduling algorithm in the Kubernetes platform by considering the network bandwidth parameter to improve the performance of MLOps systems in scenarios involving cloud computing and edge devices in IoT environments. Their approach is tightly coupled with the Kubeflow platform, thereby limiting its generalizability to other MLOps frameworks. Additionally, the paper lacks a formal mathematical model that accurately describes the structure of the problem, and the optimization methodology used, which can make it difficult to replicate and compare results with other approaches. Nonetheless, the paper is an important contribution toward considering network parameters in the process of scheduling ML tasks in distributed environments. Another paper by A. Shridhar et al. [5] also addresses the topic of optimal resource allocation to components for training machine learning models in distributed environments. However, the authors focus only on the process of training models, are limited to cloud environments only, and consider GPU and network latency. The use of genetic algorithms for resource allocation also appears in the work of Z. Ding et al. [6], albeit strictly within the Kubernetes context, and they focus on optimizing microservices allocation without a specific MLOps context. However, this indicates that there have been attempts to successfully use this way of searching solutions in the optimal allocation of resources, which provides a solid foundation for further research in this promising direction. This is also confirmed by the article by Y. Zhang et al. [7] in which the genetic algorithm was also used in optimizing the operation of a fleet of robots whose performance improvements resulted from appropriate scheduling of tasks on Kubernetes clusters.

Optimizer Concept

The final version of the proposed method is intended to include the following components:

- formulation of an appropriate mathematical model,

- development and implementation of a genetic algorithm for exploring the solution space related to component-to-server allocation,
- preparation and implementation of the MLOps environment simulator used to evaluate the solutions obtained because of the genetic algorithm,
- selection of a solution from among the Pareto front using multi-criteria optimization methods.

Mathematical Model

The basic step toward optimizing an MLOps system is to develop a generalized mathematical model that formally describes the relationships between the processes that run on various components of the system, which run on the available servers. Such a model provides a starting point for further optimization analyses, enabling the precise definition of objective functions and constraints related to task allocation, scheduling, and resource utilization. This chapter presents the concept of a mathematical model developed for the analysis of a system based on MLOps architecture. Because of its complexity, a full, precise representation of the model is not included. A detailed version of the model with full specifications will be published in a separate article. This description presents the main assumptions, structure and functions of the various parts of the model, which enables understanding of its role in the conducted research.

The model input data are divided into three layers: processes, components and servers. The processes layer focuses on defining the sets of steps that make up the individual processes in the studied system with MLOps architecture, for example, in an organization. Examples of processes that can be considered are data collection and preparation, model training, etc. Next, each process is divided into steps that are components of individual processes. This ensures that for each step its requirements for processor power, amount of memory and network bandwidth needed to execute the process in a distributed environment are defined separately. The maximum number of parallel threads into which the calculation can be divided is also defined. Above a certain value, the benefit of splitting computation may be less than the cost of maintaining individual components. This includes delays caused by the network communication between them. The component layer in the proposed model performs an intermediate function between processes and hardware infrastructure. It executes specific steps (tasks) resulting from MLOps processes. Components are described by parameters that determine the consumption of resources required to keep the component active. Since the resource consumption for individual steps is determined by their own parameters, the use of such metrics for components is sufficient. In addition, a parameter is introduced to determine whether the component should be permanently active. An example of this would be a component that maintains, for example, a model repository or feature store, which are characteristic elements of almost every system based on MLOps architecture. The last layer of input data of the MLOps mathematical model is the layer of servers on which individual components are run. Servers for components can take the form of local physical machines dedicated to specific tasks or edge devices performing functions typical of IoT environments. They can also be resources made available through cloud computing. Factors describing individual servers are computing power, network bandwidth, price (maintenance of servers, cost of using cloud servers) and the number of servers of a given type. Decision variables guide the search for optimal system parameters. In the model in question, the decision variables are a matrix for allocating individual components to servers, a computation alignment matrix, which determines the maximum number of components on which a step can be run, and a constant run matrix, which determines which components must be run for the entire time the system is running. The limitations that the system under consideration must meet are as follows:

- the sum of the load generated by the operation of the components must not exceed the available CPU power and RAM that the server has,
- at least one instance of each component must be allocated to some server.

he final step in preparing the mathematical model is to define criteria for evaluating system performance. These criteria are:

- expected process execution time in MLOps,
- expected process execution time variance,
- expected number of time limit overruns for a specific task

- total cost of execution of all processes.

Based on the presented criteria, a vector objective function is formulated, minimization of which results in increased efficiency of the system.

Concept of operation of the optimizer

Because of the huge space of solutions, the algorithmic nature of the processes and the random variables, classical optimization methods, which require a full search of this space, prove to be computationally inefficient or even inapplicable. The time required to evaluate each variant of component allocation to resources grows exponentially with the number of components and servers, making a search-and-evaluate approach to each solution impossible.

Therefore, the proposed optimizer is planned to use a heuristic genetic algorithm approach, effectively searching the solution space without fully mapping it. The analyzed system's complexity makes it virtually impossible to directly determine the evaluation function's value for each solution. Therefore, a dedicated simulator of the MLOps system will be designed, which acts as a tool for estimating the characteristics of the system's performance and thus allows approximate evaluation of solutions. A diagram of the algorithm's operation with the simulator is shown in Figure 1.

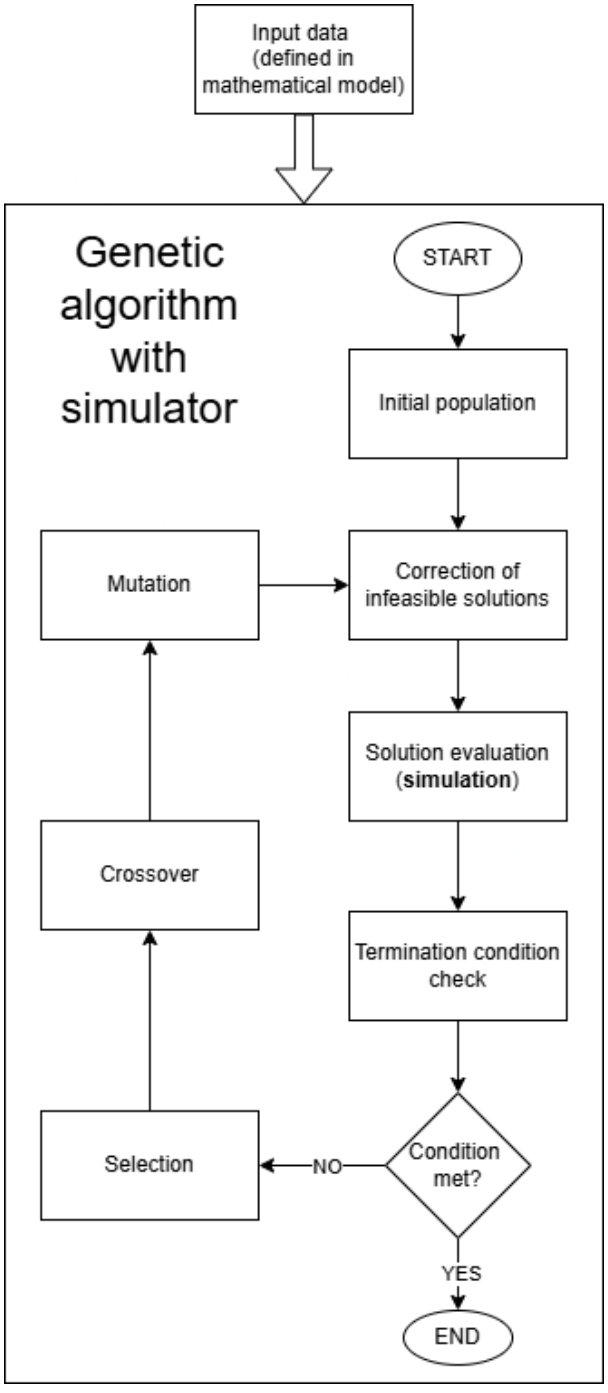

Figure 1: Diagram of the optimizer

Figure 1 presents the planned architecture of the optimizer. The genotype is a natural-valued matrix of size K×S, where K is the number of components and S is the number of servers. Each element of the matrix indicates the number of instances of a given component allocated to a specific server. Based on this matrix, the allocation strategy for components across servers is defined. A value of 0 means no allocation, while a positive integer denotes how many instances of the component are assigned to that server. In the first iteration of the optimizer operation, the value of the initial population must be determined, and then the specific allocations of components to servers must be performed. At this stage, we assume the allocations will be assigned randomly, while their number will be determined by a parameter that can be defined at the input to the simulator. The key condition at this stage of

implementation is that the allocations meet the limiting criteria defined in the mathematical model. The load required by individual component is the sum of the computing power needed to keep the component active, and the computing power needed to execute a process step. The values of the load generated by individual steps performing specific tasks should be known at the input to the system by obtaining them, for example, from performance tests of the optimized system. We assume the optimizer will be equipped with an algorithm for rejecting erroneous allocations and replacing them with successive random combinations that meet the constraints until the expected number of allocations in the initial population is got at the input of the optimizer.

Solution Evaluation

Every population including the initial one and subsequent generations of genotypes is evaluated using a simulator after undergoing selection, crossover, mutation, and correction of invalid solutions. Each simulation is run repeatedly to estimate the expected values of the random variables describing the solution criteria. Each of the criteria based on which we evaluate the solution is presented in different scales and units. Proper evaluation of solutions across all criteria involves normalizing each criterion. This ensures that the values are comparable and can be used, for example, in calculating the Euclidean metric. Following normalization, the search space can be examined geometrically within a multi-dimensional space defined by the number of evaluation criteria. The optimizer then moves on to the next step, i.e. selection (until, of course, the stop condition is reached after evaluating the population in question). After selecting the best solutions, the optimizer will proceed to the next steps, i.e. crossover and mutation. At the current stage of design, the detailed mechanisms of both these operations have not yet been definitively established. They will be refined during further experimental research, considering the specifics of the problem and the structure of the genotype representing the allocations of MLOps components to computational resources. Essentially, crossover will combine features of selected parental solutions to generate new solutions that potentially combine the best properties of both parents. This may involve, for example, exchanging portions of the allocation matrix between individuals, but the specific type of crossover will be chosen experimentally. Mutation will be responsible for making small random changes to the genotype to ensure population diversity and avoid local minimum. This may involve, for example, changing the number of component instances assigned to a server or moving a component to another node. Again, the mutation method will be designed to match the nature of the MLOps model under consideration. The choice of appropriate crossover and mutation operators will depend, among other factors, on the results of preliminary tests, the efficiency of solution space exploration, and the quality of the solutions generated by the algorithm. In each iteration of the process, the values that will form the Pareto front will be updated. The Pareto front is a set of non-dominated solutions, meaning that no other solution in the current population is better in all criteria simultaneously. Each solution on the Pareto front is optimal in the sense that improvement in one criterion would lead to deterioration in at least one other. After the final iteration of the process, it will be possible to determine from the produced Pareto set the solutions that are optimal with respect to each criterion, as well as to select the one closest to the ideal solution. The ideal solution is a hypothetical point that simultaneously minimizes (or maximizes) all criteria to their best possible values. In practice, the final selection from the Pareto front is made by applying a decision-making strategy, such as minimizing the Euclidean distance to the ideal point, assigning weights to different objectives, or incorporating domain-specific preferences.

Summary

The paper presents a method for optimizing the allocation of MLOps process components to computing resources. The approach employs a genetic algorithm, a dedicated simulator for solution evaluation, and multi-criteria optimization to identify optimal solutions. The paper describes the concept of a mathematical model of the MLOps system, enabling an accurate representation of component allocation to servers. In the proposed optimizer, a genetic algorithm searches the solution space, and a dedicated simulator evaluates each solution according to four criteria. This allows finding a set of solutions that are non-dominated with respect to each other, known as the Pareto front. While the method avoids exhaustive search of the solution space by employing a heuristic approach, its effectiveness still depends on the accuracy of the evaluation function. This stage requires a precise mapping of the MLOps system within the simulator, which constitutes a key challenge and represents the most complex part of the entire implementation. In the future, the solution is planned to be fully developed and adapted for use in on-premises, cloud, and hybrid infrastructures.

References

- A. P. Woźniak, M. Milczarek and J. Woźniak, "MLOps Components, Tools, Process, and Metrics: A Systematic Literature Review," in *IEEE Access*, vol. 13, pp. 22166-22175, 2025
- D. Kreuzberger, N. Kühl and S. Hirschl, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," in *IEEE Access*, vol. 11, pp. 31866-31879, 2023
- H. -M. Chen, S. -Y. Chen, S. -H. Hsueh and S. -K. Wang, "Designing an Improved ML Task Scheduling Mechanism on Kubernetes," *2023 Sixth International Symposium on Computer, Consumer and Control (IS3C)*, Taichung, Taiwan, pp. 60-63, 2023
- Syrigos, N. Angelopoulos and T. Korakis, "Optimization of Execution for Machine Learning Applications in the Computing Continuum," *2022 IEEE Conference on Standards for Communications and Networking (CSCN)*, Thessaloniki, Greece, pp. 118-123, 2022
- Shridhar and D. Nadig, "Heuristic-based Resource Allocation for Cloud-native Machine Learning Workloads," *2022 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Gandhinagar, Gujarat, India, pp. 415-418, 2022
- Z. Ding, S. Wang and C. Jiang, "Kubernetes-Oriented Microservice Placement With Dynamic Resource Allocation," in *IEEE Transactions on Cloud Computing*, vol. 11, no. 2, pp. 1777-1793, 1 April-June 2023
- Y. Zhang, F. Mirus, F. Pasch, K. -U. Scholl, C. Wurll and B. Hein, "A Comprehensive Modeling and Scheduling Approach for Allocating Distributed Multi-Robot Software to the Edge/Cloud," *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Abu Dhabi, United Arab Emirates, pp. 5799-5806, 2024