

Constraint-Aware Resource Allocation: A Computational Framework for Practical Optimization

Aiswarya Gurram
Independent Researcher, USA.

Abstract - Computational systems, where there is limited capacity, competing workloads and multi-operational constraints, require efficient allocation of resources. The study suggests a constraint-aware computational model to evaluate the resources state and aid feasible resource-allocation choices. The framework puts into consideration the CPU, memory, bandwidth, workload nature, task priority, service requirements, deadlines, budget, energy consumption and fairness. Resource-pressure indicators are constructed to measure the demand in comparison with available capacity. The Python analytical workflow that includes preprocessing, feature engineering, exploratory, and supervised machine learning is executed. Accuracy, Precision, Recall, F1-score, ROC AUC and confusion matrices are used to evaluate the performance of the Logistic Regression, Random Forest and Gradient Boosting classifiers. The best overall performance was achieved by the Logistic Regression with 96.6% accuracy, a weighted F1-score of 0.9663, and a ROC-AUC of 0.9994. The feature-importance analysis also showed that CPU, memory and bandwidth pressure are significant predictors of constraint severity. The results show that resource-pressure evaluation in conjunction with predictive classification can be viable in differentiating the various constraints states and also offer a computational foundation to allocate, monitor, and resource-management solutions of constraints in scalable computing settings.

Keywords - Constraint-Aware Resource Allocation, Resource Optimization, Machine Learning, Resource Pressure, Logistic Regression, Random Forest, Gradient Boosting, Computational Resource Management, Python

I. INTRODUCTION

A. Research Background

Resource allocation is a special fundamental optimization problem in computing, communications networks, cloud computing, manufacturing, and other systems that are resource-constrained [1]. The available resources have to be allocated effectively among a number of competing activities, in addition to meeting the operational requirements then effective resource allocation becomes a challenging task [2]. Conventional optimization methods, including linear and integer programming can be used to deliver mathematically optimal solutions of the problem

with clear-cut conditions [3]. Real-world situations tend to encompass reduced levels of resource capacity, responsibility, different needs, equality standards, and time constraints in calculations [4]. All these sentiments give rise to the necessity of constraint-aware solution, addressing the allocation quality and computational efficiency and scalability.

B. System Model and Constraints.

The system under consideration is composed of several tasks, competing over a limited amount of computational resources. The amount of resources required per task, and the priority of each task are specified, and the entire system is limited to capacity requirements [5]. The allocation model takes into account the constraints regarding the resource availability, minimum task needs, priority satisfaction and equitable allocation [6]. A feasibility of an allocation is one that satisfies the requirements without surpassing the amount of resources available. The model allows computation of the different workload sizes and different resource-availability conditions.

C. Aim and Objectives

Aim: To design and assess a constraint-oriented compute paradigm to effectively and scaled deploy resources when operating in realistic conditions.

The objectives are:

- To develop a resource allocation model with capacity, demand, priority and fairness constraints.
- To apply resource allocation methods of optimization and heuristics in Python.
- To compare performance of the approaches in terms of allocation efficiency, execution time, usage of resources and constraint satisfaction.
- To test the scalability and the resilience of the allocation methods to different working conditions in terms of workload and resource availability.

D. Research Contribution

The study offers a computational model relating resource optimization and a constraint-based allocation. It largely contributes to experimental comparisons of the exact and computationally efficient allocation strategies under controlled resource and workload settings. The simulations are created in Python to produce quantitative information about solution quality, computational

cost, constraint satisfaction and scalability, giving us some idea about the trade-offs when choosing resource allocation strategies to use in the real world.

II. RELATED WORK

A. Conventional Optimization-Based Resource Allocation

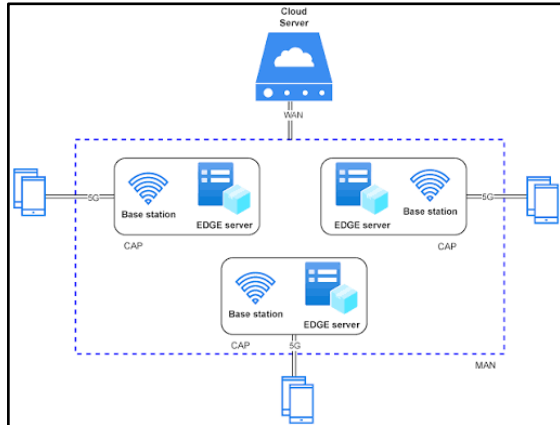


Fig. 1. Conventional Optimization-Based Resource Allocation

Mathematical optimization methods that have been widely used in traditional resource allocation research include Linear Programming (LP), Integer Linear Programming (ILP), Mixed-Integer Linear Programming (MILP), and convex optimization [7]. The techniques model resource allocation as an objective, which has specific capacity and operational limitations [8]. They can gain their first advantage, the capability to get optimal solutions in case the problem is mathematically well-defined. These methods have been used in the scheduling of processors, scheduling bandwidth in networks, scheduling cloud computing resources, and Planning production [9]. The computational complexity could dramatically rise with the increase in the number of resources, tasks, and constraints [10]. Moreover, certain assumptions like deterministic resource demand and unchanging system conditions may not give a sufficient reflection of real world environment [11].

B. Metaheuristic and Heuristic Resource Allocation

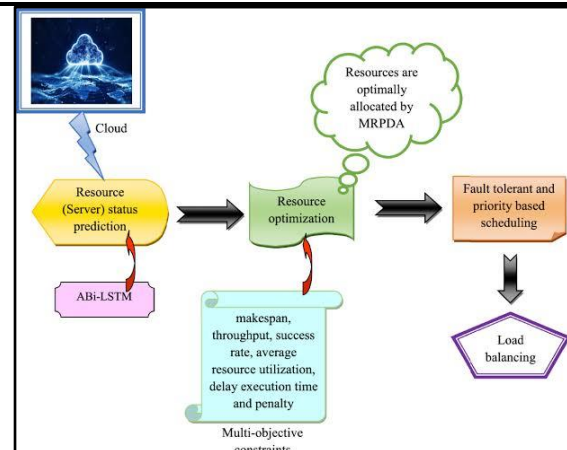


Fig. 2. Metaheuristic Resource Allocation

Heuristic and metaheuristic algorithms have become useful in the allocation of resources with problems in resource allocation that may necessitate project computation that is as exact as possible [12]. Complex resource management situations have been studied using greedy allocation, priority-based scheduling, Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and other evolutionary methods [13]. In contrast to the exact optimization approaches, they do not necessarily aim to find global optimality, but as a general rule, acceptable solutions within reasonable calculation times [14]. Heuristic algorithms and metaheuristic algorithms may include application knowledge, and metaheuristic methods will cover more of the solution space, and may be able to optimize nonlinear objectives and multiple constraints [15]. These features render them applicable to large-scale cloud, network and distributed computing environments [16]. However, algorithmic parameters, initialization, stopping criteria and workload characteristics may affect their performance heavily. A solution, which works well under certain resource condition, might not give similar results under a different resource condition [17].

C. Resource Management Conscious of Constraints

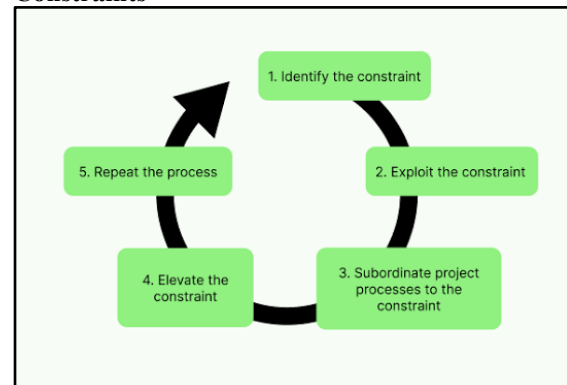


Fig. 3. Resource Management Conscious of Constraints

Increasingly, recent resource management methods put into account several practical constraints, rather than just targeting to reduce cost or maximize utilization [18]. Simultaneously, allocation decisions can be affected by capacity limitations, task priorities, deadlines, quality-of-service requirements, fairness, energy consumed, and service-level requirements [19]. Resource management, through constraint aware, tries to make sure that efficient allocation of resources occurs in a manner that meets satisfactory service levels to competing tasks [20]. Cloud and edge computing are of particular interest to these approaches and are characterized by the small number of computational resources and scalable changes in workload demand [21]. Priority based constraints could be used to guarantee that important tasks are allocated enough resources whereas fairness constraints deter overallocation to a few tasks. The use of several constraints makes the allocation problem more complex and can lead to the conflict of competing goals [22].

D. Evaluation by Computation and Simulation

Computational simulation has become a significant technique of assessing resource allocation algorithms prior to implementation in actual systems [23]. The limitation of workload, available resources, and distributions of demand, task priorities and amount of constraints allow through simulation the researcher to repeatedly test the allocation strategy under similar conditions and modify the research work variables to the desired level [24]. Python offers an appropriate computational platform since numerical analysis, optimization, analysis of data, and visualization can be combined into a reproducible experimental workflow. Numerical computation and optimization libraries enable researchers to use mathematical and heuristic allocation algorithms, and data-analysis and visualization software enable quantitative comparisons. The metrics that can be used to analyze performance include execution time, allocation cost, resource utilization, fairness, task satisfaction, and constraint-violation rate. Although these benefits exist, the design of work and working environments of experimental outcomes can be very reliant on the evaluation conditions [25].

E. Research Gap

Available literature presents solid bases of optimization-based resource allocation, heuristic allocation, and constraint-aware resource allocation. Nevertheless, much of the research considers single methods in particular workloads or concerns optimality in solutions without always considering computational cost, scalability,

fairness, resource use, and constraint satisfaction. It could still be useful to have a cohesive experimental framework comparing precise optimization and computationally efficient allocation methods in the conditions of constant practical constraints and different workload levels.

III. PROPOSED CONSTRAINT-AWARE RESOURCE ALLOCATION FRAMEWORK

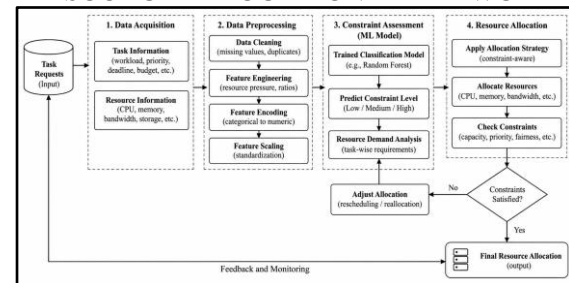


Fig. 4. Proposed Framework

The constraint based resource allocation scheme is proposed which gives a systematic method of allocation of computational resources taking into account workload demands and resource constraints. The framework starts with the data acquisition, which gathers the information in relation to the task including workload type, priority, deadline, and budget alongside the resource data including CPU, memory, bandwidth, and storage availability. These data are handed in to the preprocessing phase, which cleans data, engineers features, encodes features as categories, and scales numerical features before processing them [26]. The relationship between the task demand and resources available is also created and gives a travel of resource-pressure indicators.

This information is then processed and fed into the constraint assessment phase, where a trained machine-learned classifier provides an answer whether the operating condition establishes a Low, Medium or High constraint level. To identify task-specific requirements, resource demand is then examined to understand the resource requirements. The resource allocation phase, based on the predicted constraint condition, uses a highly sensitive to constraint strategy of resource allocation and allocates CPU, memory, bandwidth and other available resources [27]. The allocation that arises is compared to the capacity, priority, minimum-resource, and fairness constraints. In case of nonfulfillment of the requirements, resources are readjusted by reallocation or rescheduling.

IV. RESOURCE ALLOCATION ALGORITHMS

In the proposed framework, resource allocation algorithms are used to assist with the efficient decision-making in various constrained conditions. Low, Medium and High resource-constraint levels

are identified with the help of Logistic Regression that is applied as a baseline classification algorithm.

Random Forest is used to model nonlinear interactions between resource variables (CPU, memory, bandwidth, workload, priority, etc.) and the features and gives features-importance data.

Gradient Boosting is also used to enhance classification using sequential ensemble learning. A constraint-aware allocation strategy is used to allocate resources based on the available capacity, lowest task requirements, priority, and fairness, which are based on the predicted constraint level. These algorithms allow systematic comparison based on the accuracy, precision, recall, F1-score and ROC-AUC [27].

V. EXPERIMENTAL METHODOLOGY

A. Research Design

This research will take an *experimental research design* that utilizes computational research design to explore the allocation of resources in varying degrees of resource constraints. The analysis is conducted to determine the presence or absence of a computational system to operate at low, medium, or high-constraint conditions of the requirements of a given task, the resources available, the nature of the workload, and the operational parameters.

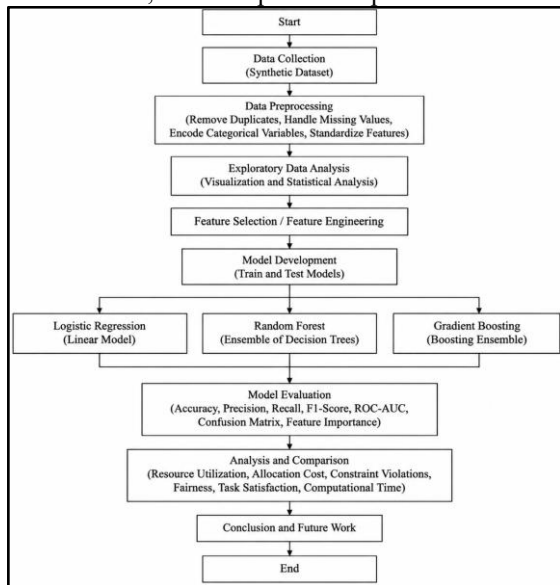


Fig. 5. Flow Diagram

Controlled synthetic environment has been chosen since it allows varying the resource demands and resource capacities in a systematic way and does not expose the analysis to unavailable or confidential operational data. It is also easy to replicate the experiment design and the design also allows deep unity in comparison of the computational models.

B. Development of Dataset

In order to reflect real-world resource-allocation circumstances, a synthetic dataset of 5,000

observations of tasks and 20 initial attributes is created. Every observation is associated with a single computational task that uses a combination of CPU, memory, bandwidth, and storage resources. The categories of workloads included: Web Service, Data Processing, AI/ML, Database, and Batch Job. Tasks were also given Low Medium High of Critical priorities and service-level classes. Variables that are resource dependent consist of CPU demand, memory demand, bandwidth demand, storage demand, minimum CPU and memory requirement, and available CPU, memory, and bandwidth. Operational variables consist of deadline, hourly budget, estimated energy usage, fairness weight, arrival time and predicted execution time. The variable to be tested, which is Constraint Level, is continuous (Low, Medium, and High constraint conditions). The obtained data set is composed of 719 low, 2,678 medium and 1, 603 high constraint databases.

System Model and Resource Constraints

For N computational tasks sharing limited resources, total allocation should not exceed available system capacity. The capacity constraint is represented as:

$$\sum_{i=1}^N X_{ij} \leq C_j, \forall j$$

where X_{ij} represents the amount of resource j allocated to task i , and C_j represents total available capacity.

Each task should also receive at least its minimum required resource:

$$X_{ij} \geq M_{ij}$$

Where M_{ij} represents the minimum resource requirement.

The general allocation objective can be expressed as:

$$\max Z = \sum_{i=1}^N w_i U_i(x_i)$$

where w_i represents task priority or fairness weight and $U_i(x_i)$ represents the utility obtained from resource allocation.

C. Data Preprocessing

Preprocessing of data is carried out prior to the development of the model to allow consistency and appropriateness to computational analysis. Identification and removal of duplicate observations are done. Task_ID is not involved in predictive modeling since it is a unique identifier, and carries no useful explanatory factors. Numerical variables are checked on missing values and intended to be addressed by mediated imputation in cases where there are such values exist [27]. The gaps in categorical values were also filled with the most frequent category. The Z-score normalization method is:

$$Z_i = \frac{X_i - \mu}{\sigma}$$

The sample size has numeric and nominal variables. The variables on different scales were transformed via the StandardScaler to standardize numerical representations such that the variables are measured on a similar numeric scale. Categorical items, such as workload type, priority and SLA class, were coded with the one-hot encoding. These transformations were implemented in a preprocessing pipeline to be sure that the same kind of transformations were used when training and testing the model.

D. Feature Engineering

There were other variables that were derived to state the correlation between requested and available resources more directly. The computation of CPU pressure was the ratio of CPU demand to CPU capacity available. Equivalent actions were devised to memory and bandwidth. A pressure values above one means that the resources being requested are more than the currently available capacity hence a possible shortage of resources [28].

E. Exploratory Data Analysis and Visualization

Exploratory Data Analysis (EDA) is conducted to know how distributions, relationships and resources within the dataset appeared. Numerical attributes are achieved using descriptive statistics, frequency distributions analyzed when there were categorical variables.

Five main visualizations are taken into account. To begin with, a bar chart was used to analyze the distribution of the Low, Medium, and High levels of constraint. Second, the mean CPU and memory requirements are compared among workload groups. Third, the correlation between the CPU demand and CPU capacity available is analyzed with the help of a scatter plot. Fourth, box plots have been used to compare the CPU resource pressure under the constraint levels. Lastly, the correlation matrix is created to look at the relationships between the numerical resource and operating variables.

F. Training and Testing Strategy

The strategy related to training and testing will be developed as follows:

To achieve predictive modeling, Constraint_Level was treated as dependent variable, and the rest of the relevant attributes and engineered features were defined as independent variables. To split dataset into train and test, an 80:20 train-test split was used, which yielded around 4,000 observations and 1,000 observations respectively, to build the model and use independent testing respectively.

The split used stratified sampling in order to maintain Low, Medium, and High proportion in both subsets in the original proportion. A constant

random state of 42 was employed to favor reproducibility.

G. Model Development and Fitting

Three of these classification algorithms are compared Linear Regression, Random Forest, and Gradient Boosting. To provide an interpretable baseline classifier, Logistic Regression is chosen. Due to the ability to model nonlinear relationships and interactions among resource variables, as well as estimate feature-importances, Random Forest is used. Gradient Boosting is added as an ensemble method that can adaptively decrease errors in its classification and simplify decision boundaries. The training of each model is carried out on the identical training dataset and preprocessing design in order to promote methodological similarity.

H. Model Evaluation

The independent test dataset was used to evaluate model performance in terms of accuracy, weighted precision, weighted recall, weighted F1-score, and multiclass ROC-AUC. The weighted measures accommodate variations in the number of observations in the constraint classes. Confusion matrices were also created to determine the errors between the classes.

$$\begin{aligned} \text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN} \\ \text{Precision} &= \frac{TP}{TP + FP} \\ \text{Recall} &= \frac{TP}{TP + FN} \\ F1 &= 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \end{aligned}$$

Comparison of the models is largely done based on weighted F1-score as it measures both precision and recall and takes into consideration the frequencies of classes. The Random Forest model of feature-importance analysis is performed to determine the resource and operational variables with an impressive impact.

I. Pseudo Code

```
Step 1: Load task and resource
information
Step 2: Preprocess data:
    - Remove duplicates
    - Handle missing values
    - Encode categorical
features
    - Normalize numerical
features
Step 3: Generate resource
pressure features
Step 4: Split data into training
and testing sets
Step 5: Train classification
models:
    - Logistic Regression
```

- Random Forest
- Gradient Boosting

Step 6: Predict constraint level:

Low / Medium / High

Step 7: Apply allocation strategy:

- Check capacity constraints
- Consider priority and fairness
- Allocate available resources

Step 8: Evaluate performance: Accuracy, F1-score, utilization, constraint violations, and execution time

The combination of these processes gives a computational method that can be repeated to assess the severity of constraints and aid further constraint sensitive allocation decisions.

VII. RESULTS AND ANALYSIS

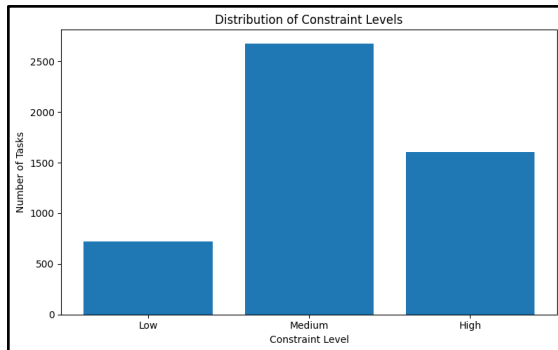


Fig. 6. Constricted Level Distribution

Constricted level tasks are centered in medium-constraint tasks which come in huge numbers compared to high constraint and low constraint. This allocation means that the resources are unevenly allocated but moderate in terms of class imbalance in the three groups.

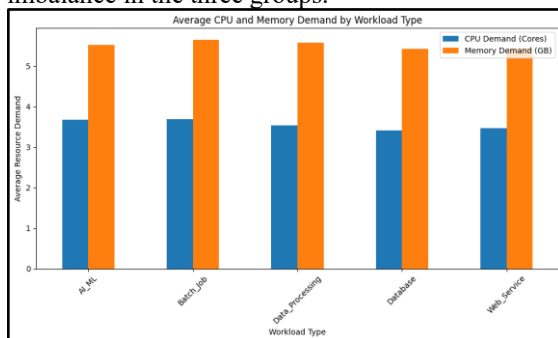


Fig. 7. Mean CPU and Memory demand by workload type

CPU and memory demands are relatively unaffected by workload types. The average demand is a bit higher with batch jobs, and the

computational resources requirements of database and web-service workload are relatively lower.

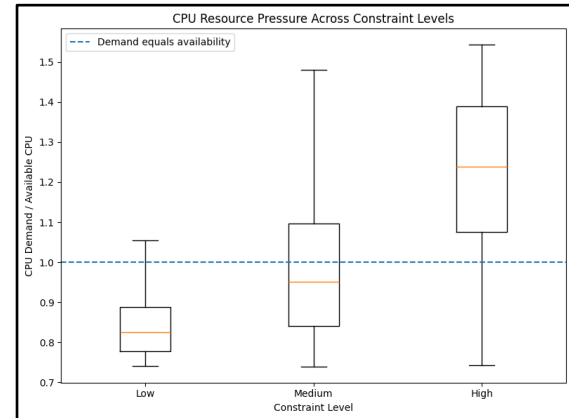


Fig. 8. CPU Resource Pressure At Each Level of Constraint

CPU pressure is high at the high levels of constraint to low constraint. Tasks that include high-configure tasks mostly surpass the demand-to-availability level of one, establishing the resource scarcity that increases in the allocated constraint categories.

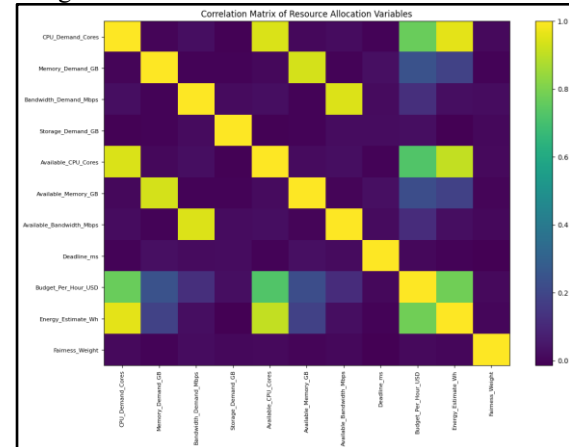


Fig. 9. Correlation Matrix

The correlation table shows a strong association between the demand and measure of the availability of the resources. There is also a significant relationship between CPU demand and energy use and budget, and very little between CPU demand and fairness.

LOGISTIC REGRESSION				
Accuracy : 0.966				
Precision: 0.9692				
Recall : 0.966				
F1 Score : 0.9663				
ROC-AUC : 0.9994				
Classification Report:				
	precision	recall	f1-score	support
High	0.9610	1.0000	0.9801	320
Low	0.8727	1.0000	0.9320	144
Medium	1.0000	0.9366	0.9672	536
accuracy			0.9660	1000
macro avg	0.9446	0.9789	0.9598	1000
weighted avg	0.9692	0.9660	0.9663	1000

Fig. 10. Logistic Regression Performance

Accuracy and weighted F1-score of the Logistic Regression models were 96.6 and 0.9663 respectively and the ROC-AUC of 0.9994 was almost perfect. It categorized Low constraints and High constraints as ideal and gave a low number of Medium-class errors.

RANDOM FOREST				
Accuracy : 0.912				
Precision: 0.9177				
Recall : 0.912				
F1 Score : 0.9108				
ROC-AUC : 0.9827				
Classification Report:				
	precision	recall	f1-score	support
High	0.9590	0.8781	0.9168	320
Low	0.9820	0.7569	0.8549	144
Medium	0.8758	0.9739	0.9223	536
accuracy			0.9120	1000
macro avg	0.9390	0.8697	0.8900	1000
weighted avg	0.9177	0.9120	0.9108	1000

Fig. 11. Random Forest Performance

Random Forest had 91.2-percent accuracy, and 0.9108 weighted F1-score. The High and Medium classes performance was still high, but the Low-class recall is lower, which suggests that finding low constraints is more difficult.

GRADIENT BOOSTING				
Accuracy : 0.95				
Precision: 0.9514				
Recall : 0.95				
F1 Score : 0.9494				
ROC-AUC : 0.9951				
Classification Report:				
	precision	recall	f1-score	support
High	0.9714	0.9563	0.9638	320
Low	0.9835	0.8264	0.8981	144
Medium	0.9309	0.9795	0.9545	536
accuracy			0.9500	1000
macro avg	0.9619	0.9207	0.9388	1000
weighted avg	0.9514	0.9500	0.9494	1000

Fig. 12. Gradient Boosting Performance

Gradient Boosting found accuracy, weighted precision, and weighted F1-score of 95.0, .9514, and 0.9494%, respectively. Good classification at all the levels of constraint indicates good non-linear

modeling to the relationships between resources and pressure.

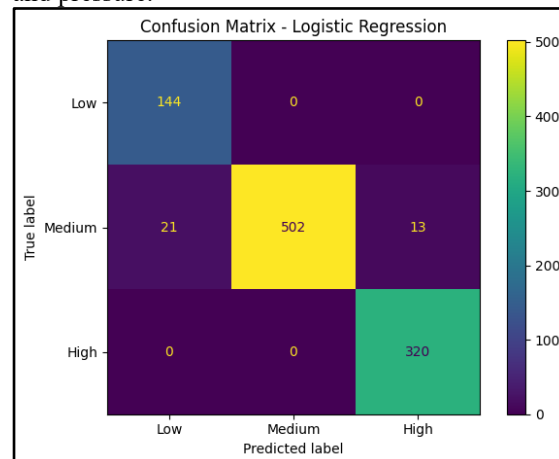


Fig. 13. The best Model Logistic Regression Confusion Matrix

The confusion matrix indicates a perfect classification of both arguments, 144 Low and 320 High. Out of 536 cases of the Medium Answers, 502 were in the right category with an overall 34 misclassifications.

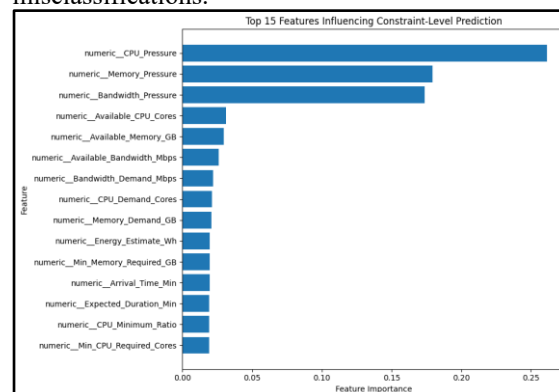


Fig. 14. Importance Analysis of the features

CPU pressure is the most influential predictor; this is followed by memory and bandwidth pressure. This proves the fact that demand-availability relationships add a significant value to the classification of constraints as compared to the single operational properties.

AVERAGE RESOURCE PRESSURE			
Constraint_Level	CPU_Pressure	Memory_Pressure	Bandwidth_Pressure
Low	0.842	0.859	0.865
Medium	0.983	1.021	1.011
High	1.220	1.177	1.170

Fig. 15. Average Resource Pressure Analysis

The average CPU, memory and bandwidth resource pressure is on the upward trend as low and High constraint levels. High-constraint tasks are above the threshold of one in all the resources and this means that there are major shortages of resources.

Discussion

TABLE 1: Model Summary

Model	Acc ura cy	Preci sion	Rec all	F1_ Scor e	RO C_ AUC
Logistic Regressio n	0.96 6	0.969 2	0.96 6	0.96 63	0.99 94
Gradient Boosting	0.95 0	0.951 4	0.95 0	0.94 94	0.99 51
Random Forest	0.91 2	0.917 7	0.91 2	0.91 08	0.98 27

The experiment results confirm the fact that resource pressure is closely related with the severity of constraints, especially in terms of CPU, memory and bandwidth dimensions. The best classification performance was of the Logistic Regression 96.6% accuracy and a weighted F1-score of 0.9663. Gradient Boosting also exhibited competent predictive competencies too as compared to the comparatively low performance by the Random Forest. This feature analysis confirmed that CPU pressure was the most important predictor. In general, the findings provide evidence in favor of constraint-based computational assessment to manage and allocate resources efficiently.

VIII. LIMITATIONS AND FUTURE WORK

A. Limitation

- CPU, memory, and bandwidth pressure are the main concerns of the proposed framework; other more practical considerations like network latency, hardware failures, security needs and dynamic pricing as well as task migration overhead are not explicitly reflected.
- Analysis has references to Logistic Regression, Random Forest, and Gradient Boosting to classify constraints. More neural methods, such as deep learning, reinforcement learning, and multiple objectives, were not compared, restricting the comparison between more comprehensive methods of computation.

B. Future Direction

Future study is able to bridge the suggested framework with the dynamic and real time resource circumstances such as changing workloads, network latency, energy availability and cost of migrating tasks. For the adaptive allocation decisions, advanced methods can be explored, including reinforcement learning, deep learning,

and multi-objective optimization [30]. Future studies are to combine actual cloud or edge-computing systems, and measure allocation plans by alternative measures, such as response time, power efficiency, SLA breaches, cost of operation and fairness over time.

IX. CONCLUSION

The study gave a constraint based model of computational analysis of resource allocation under different operational conditions. CPU, memory and bandwidth indicators of resource pressure were combined with machine-learning-based approaches to categorize the level of constraint. The best score was obtained using Logistic Regression which had 96.6% accuracy and weighted F1-score of 0.9663 succeeded by Gradient Boosting and Random Forest. The most significant predictor to come up during the analysis was CPU pressure, and resource pressure steadily rose with Low, Medium and High constraint levels. The results show that a resource-demand analysis plus predictive modeling can offer a successful base of uncovering resource shortage and enable systematic constraint-conscious resource-decision making.

REFERENCE

- [1] Liu, Q., Choi, N. and Han, T., 2021, November. Constraint-aware deep reinforcement learning for end-to-end resource orchestration in mobile networks. In *2021 IEEE 29th International Conference on Network Protocols (ICNP)* (pp. 1-11). IEEE.
- [2] Shin, S.Y., Brun, Y., Balasubramanian, H., Henneman, P.L. and Osterweil, L.J., 2017. Discrete-event simulation and integer linear programming for constraint-aware resource scheduling. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(9), pp.1578-1593.
- [3] Ouyang, T., Zhao, K., Zhang, X., Zhou, Z. and Chen, X., 2023, May. Dynamic edge-centric resource provisioning for online and offline services co-location. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications* (pp. 1-10). IEEE.
- [4] Wang, T., Yang, L., Wang, C., Qin, C., Deng, L., Wu, W., Wang, J., Shen, L. and Xiong, H., 2024. Towards constraint-aware learning for resource allocation in NFV networks. *arXiv preprint arXiv:2410.22999*.
- [5] Babburi, S., 2024. Explainable AI Framework for Policy-Compliant Anomaly Detection in Data Pipelines. *International Journal of Communication Networks and Information Security*.
- [6] Leipnitz, M.T. and Nazar, G.L., 2023. Constraint-aware multi-technique approximate high-level synthesis for FPGAs. *ACM*

- Transactions on Reconfigurable Technology and Systems*, 16(4), pp.1-28.
- [7] Wang, T., Yang, L., Wang, C., Qin, C., Deng, L., Shen, L. and Xiong, H., 2024. Towards constraint-aware learning for resource allocation in nfv-enabled networks.
 - [8] Li, J., Zhang, L.L., Xu, J., Wang, Y., Yan, S., Xia, Y., Yang, Y., Cao, T., Sun, H., Deng, W. and Zhang, Q., 2023, August. Constraint-aware and ranking-distilled token pruning for efficient transformer inference. In *Proceedings of the 29th ACM SIGKDD Conference on knowledge discovery and data mining* (pp. 1280-1290).
 - [9] Almeida, G.M., Abdel-Rahman, M.J. and Cardoso, K.V., 2024, May. Constraint-Aware Deep Reinforcement Learning for vRAN Dynamic Placement. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)* (pp. 337-350). SBC.
 - [10] Cui, H., 2024. Task Graph Modeling and Constraint-Aware Scheduling for Reliable Large Language Model Agent Execution. *Artificial Intelligence and Computing Innovations*, 4(3).
 - [11] Kou, J. and Pei, E., 2024. Constraint-aware resource matching for virtual machine placement in cloud environments. *Artificial Intelligence and Computing Innovations*, 4(1).
 - [12] Ouadi, A.E. and Abdoun, O., 2024, November. Rate-Distortion Theory for Constraint-Aware Digital Twins in Smart Warehouse Environments. In *International Conference on Big Data and Artificial Intelligence Applications* (pp. 195-207). Cham: Springer Nature Switzerland.
 - [13] Askarizade Haghighi, M., Maeen, M. and Haghparsat, M., 2019. An Energy-Efficient Dynamic Resource Management Approach Based on Clustering and Meta-Heuristic Algorithms in Cloud Computing IaaS Platforms: MA Haghighi et al. *Wireless personal communications*, 104(4), pp.1367-1391.
 - [14] Yakubu, I.Z. and Murali, M., 2023. An efficient meta-heuristic resource allocation with load balancing in IoT-Fog-cloud computing environment. *Journal of Ambient Intelligence and Humanized Computing*, 14(3), pp.2981-2992.
 - [15] Marathe, S., Neelima, T., Anusha, G. and Medishetti, S.K., 2024, December. DOA: Optimizing Resource Utilization in Cloud Environment Using Multi-Objective Meta-Heuristic Scheduling Algorithm. In *2024 International Conference on Advancement in Renewable Energy and Intelligent Systems (AREIS)* (pp. 1-9). IEEE.
 - [16] Laboni, N.M., Safa, S.J., Sharmin, S., Razzaque, M.A., Rahman, M.M. and Hassan, M.M., 2022. A hyper heuristic algorithm for efficient resource allocation in 5G mobile edge clouds. *IEEE Transactions on Mobile Computing*, 23(1), pp.29-41.
 - [17] Reddy, P., Verma, A., Verma, K., Singh, A. and Soni, A., 2023. Efficient resource allocation in cloud computing environments: A modelling perspective. *International Journal of Technology and Modeling*, 2(2), pp.99-112.
 - [18] El Tanab, M. and Hamouda, W., 2016. Resource allocation for underlay cognitive radio networks: A survey. *IEEE Communications Surveys & Tutorials*, 19(2), pp.1249-1276.
 - [19] Matsuda, N., 2023. SUSTAINABLE GREEN CLOUD INFRASTRUCTURE FOR SMART MANUFACTURING USING CARBON-AWARE RESOURCE MANAGEMENT. *International Journal of Economic Social Science and Management LAW*, 4(4), pp.124-135.
 - [20] Sun, X., Liu, Q., Zhang, K., Shen, S., Wang, F., Zhuang, Y., Zhang, Z., Gong, W., Wang, S., Yang, L. and Huo, X., 2024. Optimizing student ability assessment: a hierarchy constraint-aware cognitive diagnosis framework for educational contexts. *arXiv preprint arXiv:2412.04488*.
 - [21] Thinakaran, P., 2021. Embracing Heterogeneity in Cloud Platforms through Adaptable Resource Management Framework.
 - [22] Manoharan, D. and Vasagam, M., 2024. Devops Lifecycle Management And Cloud Migration Assessments: A Security-Driven CICD Perspective. *JOURNAL OF INTERNATIONAL CRISIS AND RISK COMMUNICATION RESEARCH*.
 - [23] Armesto, L., Moura, J., Ivan, V., Erden, M.S., Sala, A. and Vijayakumar, S., 2018. Constraint-aware learning of policies by demonstration. *The International Journal of Robotics Research*, 37(13-14), pp.1673-1689.
 - [24] Khallouli, W. and Huang, J., 2022. Cluster resource scheduling in cloud computing: literature review and research challenges. *The Journal of supercomputing*, 78(5), pp.6898-6943.
 - [25] Iqbal, N., Imran, Ahmad, S., Ahmad, R. and Kim, D.H., 2021. A scheduling mechanism based on optimization using IoT-tasks orchestration for efficient patient health monitoring. *Sensors*, 21(16), p.5430.
 - [26] Sato, T., 2021. Adaptive Resource Management in Cloud-Native Architectures Using Predictive Analytics and Reinforcement Learning Techniques. *American International*



- Journal of Computer Science and Technology*, 3(2), pp.1-10.
- [27] Sulaiman, M., Ahmadi, M., Salahuddin, M.A., Boutaba, R. and Saleh, A., 2023, May. Generalizable resource scaling of 5G slices using constrained reinforcement learning. In *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium* (pp. 1-9). IEEE.
- [28] NR, M.K., Nijaguna, G.S., Chaithra, K.N., Alsalami, Z. and Kumari, K.A., 2024, March. eMRA: A Multi-Objective Resource Allocation Technique using Genetic Algorithms for Cloud Computing. In *2024 International Conference on Distributed Computing and Optimization Techniques (ICDCOT)* (pp. 1-5). IEEE.
- [29] Gola, K.K., Singh, B.M., Gupta, B., Chaurasia, N. and Arya, S., 2023. Multi-objective hybrid capuchin search with genetic algorithm based hierarchical resource allocation scheme with clustering model in cloud computing environment. *Concurrency and Computation: Practice and Experience*, 35(7), p.e7606.
- [30] Poliakov, E., Hung, W.J. and Huang, C.C., 2023, October. Efficient constraint-aware neural architecture search for object detection. In *2023 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)* (pp. 733-740). IEEE.