




ISSN: 2617-6548

URL: [www.ijirss.com](http://www.ijirss.com)



## Optimized security-aware VM placement for enhanced intrusion tolerance and resilience in IaaS clouds using MILP

 Mahmoud Aljawarneh<sup>1</sup>,  Qais Al-Na'amneh<sup>2</sup>,  Rahaf Hazaymih<sup>3</sup>,  Ayoub Alsarhan<sup>4\*</sup>,  Khalid Hamad Alnafisah<sup>5</sup>, Nayef H. Alshammari<sup>6</sup>,  Sami Aziz Alshammari<sup>7</sup>

<sup>1,2</sup>Faculty of Information Technology, Applied Science Private University, Amman, Jordan.

<sup>3</sup>Dept. Computer Science, Jordan University of Science and Technology, Irbid, Jordan.

<sup>4</sup>Department of Information Technology, Faculty of Prince Al-Hussein of Information Technology, The Hashemite University, Zarqa, Jordan.

<sup>5</sup>Department of Computer Sciences, Faculty of Computing and Information Technology, Northern Border University Rafha, Saudi Arabia.

<sup>6</sup>Department of Computer Science, Faculty of Computers and Information Technology, University of Tabuk, Tabuk, Saudi Arabia.

<sup>7</sup>Department of Information Technology, Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia.

Corresponding author: Ayoub Alsarhan (Email: [khalid.alnafisah@nbu.edu.sa](mailto:khalid.alnafisah@nbu.edu.sa))

### Abstract

Infrastructure-as-a-Service (IaaS) clouds offer unparalleled flexibility but introduce complex security challenges, particularly concerning Virtual Machine (VM) placement. Security-oblivious VM allocation can lead to catastrophic failures if a physical server is compromised, as all co-resident VMs become vulnerable, diminishing service resilience and escalating the potential damage (blast radius). This research proposes a novel, user-driven framework for security-aware VM placement that leverages Mixed Integer Linear Programming (MILP) to enhance intrusion tolerance and service resilience while managing operational costs. The framework allows administrators to define granular security policies, including VM criticality, service compositions, mandatory VM separation, service diversity requirements, and anti-affinity rules. These policies are integrated into the MILP model alongside traditional objectives like energy, latency, and provisioning cost minimization, governed by user-configurable weights. Through comprehensive simulations based on 60 VMs and 20 servers for weight analysis, and scaling up to 120 VMs for performance evaluation, we demonstrate the framework's ability to significantly reduce security risks, such as minimizing the potential blast radius and ensuring service component dispersion. For instance, increasing criticality weight ( $W_{crit}$ ) from 0 to 2.0 reduced the maximum blast radius from 20-22 to 13-15, though with an increase in the security-focused objective value. Ten comparative analyses illustrate the impact of various security postures on overall system performance and cost.

**Keywords:** Cloud security, IaaS, Intrusion tolerance, Mixed integer linear programming (MILP), Resilience, Resource allocation, Security-aware optimization, VM placement.

**DOI:** 10.53894/ijirss.v8i5.8601

**Funding:** This work is supported by Northern Border University, Arar, Kingdom of Saudi Arabia (Grant number: NBU-FFR-2025-3555-04).

**History:** Received: 29 May 2025 / Revised: 3 July 2025 / Accepted: 7 July 2025 / Published: 17 July 2025

**Copyright:** © 2025 by the authors. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Competing Interests:** The authors declare that they have no competing interests.

**Authors' Contributions:** All authors contributed equally to the conception and design of the study. All authors have read and agreed to the published version of the manuscript.

**Transparency:** The authors confirm that the manuscript is an honest, accurate, and transparent account of the study; that no vital features of the study have been omitted; and that any discrepancies from the study as planned have been explained. This study followed all ethical practices during writing.

**Publisher:** Innovative Research Publishing

## 1. Introduction

The pervasive adoption of Infrastructure-as-a-Service (IaaS) cloud computing has fundamentally reshaped the landscape of IT resource provisioning, offering unprecedented agility, scalability, and potential cost efficiencies [1]. However, this paradigm shift introduces complex security challenges, particularly stemming from the shared, multi-tenant nature of cloud infrastructures and the abstraction layers inherent in virtualization [2]. Among these challenges, the strategic placement of Virtual Machines (VMs) onto physical servers (PSs) emerges as a critical yet often underestimated factor in an organization's overall security posture [1]. Traditional VM placement algorithms have overwhelmingly focused on optimizing operational metrics such as resource utilization and energy consumption [3], Quality of Service (QoS) [4] and monetary cost [5]. While vital, such security-oblivious approaches can inadvertently create high-risk configurations. A single compromised PS, for instance, can lead to the compromise of all co-resident VMs, an event with a potentially catastrophic "blast radius" that severely diminishes service resilience and can lead to significant data breaches and financial losses [6].

In an era of increasingly sophisticated cyber threats [7] and a growing reliance on cloud platforms for hosting mission-critical and sensitive applications, enhancing system survivability through proactive measures is paramount. This necessitates a move towards security-aware VM placement strategies designed to bolster intrusion tolerance and service resilience. Intrusion tolerance, as defined by Gill et al. [8] is the ability of a system to continue providing essential services, possibly at a degraded level, despite parts of the system being compromised by an attack. Service resilience, in the cloud context, refers to the capacity of distributed applications (often architected as microservices [6]) to maintain functionality even when underlying physical or virtual components fail or are attacked [9]. Achieving these security objectives requires a deliberate and optimized distribution of VMs based on their individual security characteristics (e.g., criticality, sensitivity), their interdependencies within logical services, and the overarching security policies and risk appetite of the cloud user or provider. This proactive approach to security can significantly reduce the potential impact of security incidents and improve the overall dependability of cloud services.

Current approaches to VM placement often fall short in addressing these multifaceted security demands. Heuristic-based approaches, while scalable, typically offer no guarantees of optimality and may struggle to satisfy a complex web of interdependent security constraints [10]. While some research has begun to explore security aspects in VM placement, such as VM isolation [11], mitigation of side-channel attacks [8], or general risk minimization [12] a comprehensive framework that allows administrators to define and weigh a rich set of granular security policies against operational costs is largely missing. This paper addresses this gap by proposing a novel, user-driven framework leveraging mixed-integer linear programming (MILP) [13]. MILP provides a robust mathematical foundation for finding provably optimal solutions (given model accuracy and solver time limits) to complex combinatorial optimization problems.

Our framework empowers cloud administrators or security-conscious tenants to define and integrate diverse security policies directly into the VM placement optimization process. These policies include:

- **VM Criticality/Sensitivity Classification:** Assigning distinct importance levels to VMs based on the data they process or the function they serve.
- **Logical Service Composition:** Grouping VMs that collaboratively deliver a specific application or service.
- **Mandatory VM Separation:** Enforcing that specified VM pairs (e.g., redundant components, VMs in different security tiers) are never co-located on the same physical server.
- **Service Component Diversity:** Ensuring that the constituent VMs of a critical service are distributed across a minimum number of distinct PSs to prevent single points of failure.
- **Configurable anti-affinity rules:** prohibiting the co-location of VMs possessing conflicting security attributes (e.g., a VM processing highly sensitive data should not share a host with an internet-exposed, potentially vulnerable VM).

These security considerations are translated into mathematical constraints or weighted components within a multi-objective MILP formulation. Users can explicitly define the relative importance of these security goals (e.g., minimizing blast radius proxy, maximizing service diversity) against traditional operational cost factors (energy, latency, provisioning), thus enabling a flexible, policy-driven approach to VM placement. The interaction with the framework is envisioned via

configuration mechanisms where these policies and their corresponding weights are specified by the user. This research makes the following primary contributions:

- 1) A novel MILP-based optimization framework specifically designed for security-aware VM placement, with an explicit focus on enhancing intrusion tolerance and service resilience within IaaS environments.
- 2) The integration of a granular, user-configurable suite of security policies encompassing VM criticality, service composition, mandatory separation, service diversity targets, and attribute-based anti-affinity rules into a unified multi-objective optimization model.
- 3) A comprehensive experimental evaluation, based on simulations with up to 60 VMs on 20 servers for weight analysis and scaling up to 120 VMs for performance assessment. This evaluation demonstrates the framework's capability to significantly improve security posture (e.g., achieving a 30-35% reduction in the maximum blast radius metric by tuning criticality weights) and provides a quantitative analysis of the inherent trade-offs against operational costs and resource utilization under diverse security policies.
- 4) Ten distinct comparative graphical analyses that meticulously illustrate these security-efficiency trade-offs and overall system behavior, offering actionable insights for cloud administrators.

The remainder of this paper is organized as follows: Section II provides a detailed review of existing literature on VM placement and cloud security. Section III elaborates on the system model, defines the security parameters, and formally presents the MILP problem formulation. Section IV describes the proposed optimization framework and its operational workflow. Section V outlines the simulation setup, defines the performance and security metrics, and discusses the empirical results. Finally, Section VI summarizes the findings and suggests avenues for future research, including the development of scalable heuristics informed by the insights from this MILP-based approach.

## **2. Related Work**

The Virtual Machine Placement (VMP) problem has been a focal point of cloud computing research for over a decade, driven by the need to optimize resource utilization and operational efficiency. Initial efforts predominantly centered on minimizing energy consumption through VM consolidation [14, 15], reducing costs related to resource provisioning and licensing [16, 17] and ensuring Quality of Service (QoS) parameters such as performance and availability [18, 19]. Many of these works employ heuristic algorithms like First-Fit Decreasing (FFD), Best-Fit Decreasing (BFD) [20] or meta-heuristics such as Genetic Algorithms (GA) [21, 22] Simulated Annealing (SA) [23] Ant Colony Optimization (ACO) [24] [25] and Particle Swarm Optimization (PSO) [26, 27]. While these methods offer scalability for large data centers, they typically provide suboptimal solutions and may not adeptly handle complex, interdependent constraints, especially when multiple conflicting objectives are present. Mixed Integer Linear Programming (MILP) has also been applied to VMP [28-30], often yielding optimal solutions for specific objectives such as energy or cost, but historically with less emphasis on comprehensive, user-defined security policies.

As cloud adoption has matured, security considerations in VMP have gained increasing attention. Early security-focused research often addressed specific threats or vulnerabilities. For instance, several studies have proposed placement strategies to enhance VM isolation and prevent malicious co-residency [31, 32], aiming to create stronger logical or physical separation between VMs of different tenants or trust levels. Other works have focused on mitigating side-channel attacks [29, 33] where information can leak between co-resident VMs on the same physical host, by developing placement algorithms that avoid locating potential attacker and victim VMs together or by placing VMs on hardware with specific isolation features. Risk-based VMP approaches have also been proposed, which typically involve assessing the risk associated with placing a VM on a particular server based on server vulnerability, VM sensitivity, and potential attack impact, then optimizing placement to minimize overall system risk [34-36]. Some frameworks integrate security monitoring and intrusion detection (IDS) aspects with placement, though often these are reactive rather than proactive design choices embedded in the initial allocation [7].

The concepts of intrusion tolerance and service resilience are particularly relevant to security-aware VMP. Research in this area often emphasizes the strategic use of diversity and redundancy [20, 37]. For example, distributing replicas of a service or components of a distributed application across different physical hosts, racks, or even data centers (fault domains) can significantly improve the service's ability to withstand failures or targeted attacks [38]. However, achieving such diversity often comes at the cost of increased communication latency or higher resource consumption, necessitating a careful balance.

While these prior works have made valuable contributions to specific facets of secure VMP, a gap exists for a holistic framework that allows administrators to: (a) define a comprehensive and granular set of security policies (criticality, separation, diversity, anti-affinity); (b) integrate these policies directly into an optimization model alongside traditional operational objectives; (c) explicitly control the trade-offs between security and efficiency through configurable weights; and (d) obtain provably optimal placements based on these multi-faceted criteria. Many existing multi-objective VMP solutions [39-41] might combine a few objectives, but often lack the specific focus on intrusion tolerance metrics (like blast radius proxies) and the fine-grained user control over a broad range of security policy inputs that our MILP-based framework provides. Heuristic approaches, in particular, may struggle to consistently satisfy all such complex and potentially conflicting security constraints simultaneously while also optimizing for cost or performance.

Our proposed research directly addresses this gap by developing an MILP framework that explicitly models various security requirements as either hard constraints or weighted objectives. This approach enables a proactive security posture by embedding intrusion tolerance and resilience considerations into the initial VM placement decisions. The use of MILP

facilitates a rigorous, quantitative analysis of the "price of security," allowing administrators to make informed decisions based on clear trade-offs, a capability often less apparent with purely heuristic solutions. This work extends the traditional application of MILP in VMP by foregrounding a user-configurable, multi-objective security paradigm. Table 1 provides a comparative summary.

**Table 1.**

Comparative Summary of Related Work Approaches and Proposed Framework.

Feature / Aspect	Heuristic VMP	Isolation/Risk-based VMP	Resilience-focused VMP	Our Proposed MILP Framework
Optimization Goal(s)	Often single (e.g., energy, cost) or limited multi-objective	Primarily, security isolation or risk score minimization	Primarily, service availability/diversity	Multi-objective (Operational Costs + Multiple Tunable Security Goals)
Solution Optimality	Sub-optimal, no guarantee	Varies, often heuristic	Varies, often heuristic	Optimal (within model scope & solver time limits)
User-defined Policy Granularity	Low to Moderate	Moderate (e.g., risk levels, separation rules)	Moderate (e.g., diversity targets)	High (VM Crit., Services, $D_s^{min}$ SepPairs, AntiAffinRules, Weights)
VM Criticality Handling	Typically not explicit	Sometimes via risk scores	Indirectly via replication	Yes, explicit $C_i$ and $W_{crit}$
Service Definitions	Rare	Rare	Often implicit in replication	Yes, explicit $Svc_s$ and $VMs (Svc_s)$
Mandatory VM Separation	Hard to enforce strictly	Core focus for some	Can be a feature	Yes, via hard constraints (Eq. 5)
Service Diversity Management	Ad-hoc / Limited	Not the primary focus	Core focus	Yes, via $Dsmin$ , $zsj$ , $DivSlack_s$ , $W_{div}$ (Eqs. 6-8)
Anti-Affinity Rules Support	Limited / Custom rules	Limited	Limited	Yes, via linearized constraints (Equations 2-11)
Blast Radius Consideration	No / Indirect	Indirect (via isolation)	Indirect (via diversity)	Yes (via $W_{crit}$ proxy in objective; explicit metric evaluation)
Service Resilience Consideration	Indirect (via general HA)	Indirect	Core focus (via diversity)	Yes (via $W_{div}$ and diversity metrics)
Quantitative Trade-off Analysis	Difficult	Difficult	Limited	Yes (Core feature via weight variation experiments)
Scalability (Large Instances)	Generally High	Moderate to High	Moderate to High	Moderate (exact MILP is NP-hard)
Example Citations	Koubaa et al. [42] and Barthwal et al. [43]	Bhonde and Devane [44] and Elshabka et al. [45]	Attaoui and Sabir [46], Shah, et al. [47] and Hidayat et al. [48]	<i>This Work</i>

### 3. System Model and Security-Aware Problem Formulation

This section details the IaaS system model, introduces security-specific parameters, and formulates the security-aware VM placement problem as an MILP. Key notations are summarized in Table 2.

#### 3.1. System Infrastructure Model

The IaaS infrastructure comprises a set  $PS$  of  $n$  physical servers, each  $PS_j$  characterized by its available resources  $R_{j,r}^{avail}$  across types  $r \in R$  (CPU, Memory, Storage, Bandwidth) and base operational cost factors  $E_{base,j}$ ,  $L_{base,j}$ ,  $P_{base,j}$ . A set  $VM$  of  $k$  virtual machines needs placement, each  $VM_i$  having resource requirements  $R_{i,r}^{req}$ . Each server  $PS_j$  can host a maximum of  $V_{maxj}$  VMs.

### 3.2. Security Parameters and Definitions

Security policies are defined by several parameters:  $C_i$  denotes the criticality of  $V M_i$ . Logical services  $Svc_s \in SVC$  are composed of sets of VMs,  $V Ms(Svc_s)$ . Specific VM pairs in  $SepPairs$  must be separated. Each service  $Svc_s$  may require its VMs to span a minimum of  $D_s^{min}$  distinct physical servers.  $AntiAffinRules$  define VM categories that should not be co-located. User-defined weights  $\alpha, \beta, \gamma$  scale operational cost components, while  $W_{op}, W_{crit}, W_{div}$  balance operational cost against security objectives in the overall optimization.

### 3.3. Decision Variables

The primary decision variable is  $x_{ij}$ . Auxiliary variables  $z_{sj}$  and  $DivSlack_s$  facilitate modeling service diversity. Variables  $has Set1_j, has Set2_j$  are used in linearizing anti-affinity rules.

**Table 2.**

Notation and Parameters Used in the MILP Formulation.

Symbol	Description	Type/Domain
<b>Sets &amp; Indices</b>		
PS	Set of Physical Servers	Set
VM	Set of Virtual Machines	Set
SVC	Set of Services	Set
R	Set of Resource Types (C, M, S, B)	Set
$j \in PS$	Index for physical servers	Index
$i \in VM$	Index for virtual machines	Index
$s \in SVC$	Index for services	Index
$r \in R$	Index for resource types	Index
<b>Input Parameters – Infrastructure &amp; VM</b>		
n	Total number of physical servers, $n =  PS $	Parameter (Integer)
k	Total number of virtual machines, $k =  VM $	Parameter (Integer)
$R_{j,r}^{avail}$	Available capacity of resource r on $PS_j$	Parameter ( $R^+$ )
$R_{i,r}^{req}$	Required capacity of resource r by $VM_i$	Parameter ( $R^+$ )
$Ebase_j, Lbase_j, Pbase_j$	Base operational cost factors for $PS_j$	Parameter ( $R^+$ )
$V_{maxj}$	Maximum VMs allowed on $PS_j$	Parameter ( $Z^+$ )
<b>Input Parameters – Security &amp; Policy</b>		
$C_i$	Criticality score of $VM_i$	Parameter ( $Z^+$ )
$V Ms(Svc_s)$	Set of VM indices belonging to service $Svc_s$	Parameter (Set of $Z^+$ )
$SepPairs$	Set of VM index pairs ( $VM_a, VM_b$ ) requiring separation	Parameter (Set of Pairs)
$D_s^{min}$	Minimum diversity (distinct PSs) for service $Svc_s$	Parameter ( $Z^+$ )
$AntiAffinRules$	Set of anti-affinity rules (defined by attributes and values)	Parameter (Structured List)
<b>Input Parameters – Weights</b>		
$\alpha, \beta, \gamma$	Weights for energy, latency, and provisioning cost components	Parameter ( $\in [0,1]$ )
$W_{op}$	Weight for the overall operational cost term in the objective	Parameter ( $R^+$ )
$W_{crit}$	Weight for the criticality co-location penalty term	Parameter ( $R^+$ )
$W_{div}$	Weight for the service lack-of-diversity penalty term	Parameter ( $R^+$ )
$X_{ij}$	if $VM_i$ is allocated to $PS_j$ ; 0 otherwise	Binary Variable
$Z_{sj}$	if $PS_j$ hosts any VM of $Svc_s$ ; 0 otherwise	Binary Variable
$DivSlack_s$	Shortfall in diversity for $Svc_s$	Continuous Var ( $\geq 0$ )
$has Set1_j, has Set2_j$	Auxiliary for anti-affinity rules	Binary Variable
<b>Decision Variables</b>		
<b>Other Constants</b>		
$M_k$	Big-M constant for linearization (e.g., $k + 1$ )	Constant (Large $R^+$ )

### 3.4. Objective Function

The objective is to minimize a weighted sum of operational costs and security-related penalties:

$$\begin{aligned}
 \min \quad & W_{op} \sum_{i \in VM} \sum_{j \in PS} x_{ij} \cdot (\alpha Ebase_j + \beta Lbase_j + \gamma Pbase_j) \\
 & + W_{crit} \sum_{i \in VM} \sum_{j \in PS} x_{ij} C_i \\
 & + W_{div} \sum_{s \in SVC} DivSlack_s
 \end{aligned} \tag{1}$$

The second term,  $W_{crit} \sum_{i,j} x_{ij} C_i$ , serves as a linear proxy to discourage placing many highly critical VMs by penalizing the total sum of criticalities of all placed VMs. While not directly minimizing the maximum criticality sum on a single server (a more direct blast radius metric), it tends to distribute critical VMs or favor servers for less critical VMs when  $W_{crit}$  is significant. The true "Max Blast Radius" is evaluated as a separate metric post-optimization.

### 3.5. Constraints

#### 1) Standard VMP Constraints:

$$\forall j \in PS, \forall r \in R : \sum_{i \in VM} x_{ij} \cdot R_i \cdot r_{req} \leq R_j \cdot r_{avail} \quad (2)$$

$$\forall i \in VM : \sum_{j \in PS} x_{ij} = 1 \quad (3)$$

$$\forall j \in PS : \sum_{i \in VM} x_{ij} \leq Vmax_j \quad (4)$$

#### 2) Security-Specific Constraints:

- Mandatory VM Separation:

$$\forall (V M_a, V M_b) \in SepPairs, \forall j \in PS : x_{aj} + x_{bj} \leq 1 \quad (5)$$

- Service Diversity Linking ( $z_{sj}$ ):

$$\forall s \in SVC, \forall i' \in VMs(Svc_s), \forall j \in PS : z_{sj} \geq x_{i'j} \quad (6)$$

$$\forall s \in SVC, \forall j \in PS : z_{sj} \leq \sum_{i' \in VMs(Svc_s)} x_{i'j} \quad (7)$$

- Minimum Service Diversity (with Slack for Objective):

$$\forall s \in SVC : \sum_{j \in PS} z_{sj} + DivSlack_s \geq D_s^{min} \quad (8)$$

- Anti-Affinity Rules (Example Linearization): For a rule: "VMs of attribute 'Attr1=Val1' cannot co-locate with VMs of 'Attr2=Val2'." Let  $Set_1 = \{i \in VM \mid Attr1_i = Val1\}$  and  $Set_2 = \{i \in VM \mid Attr2_i = Val2\}$ . For each  $PS_j \in PS$ , introduce binary  $has\_Set1_j, has\_Set2_j$ .  $M_k = k + 1$ .

$$\sum_{i \in Set_1} x_{ij} \leq M_k \cdot has\_Set1_j \quad ; \quad has\_Set1_j \leq \sum_{i \in Set_1} x_{ij} \quad (9)$$

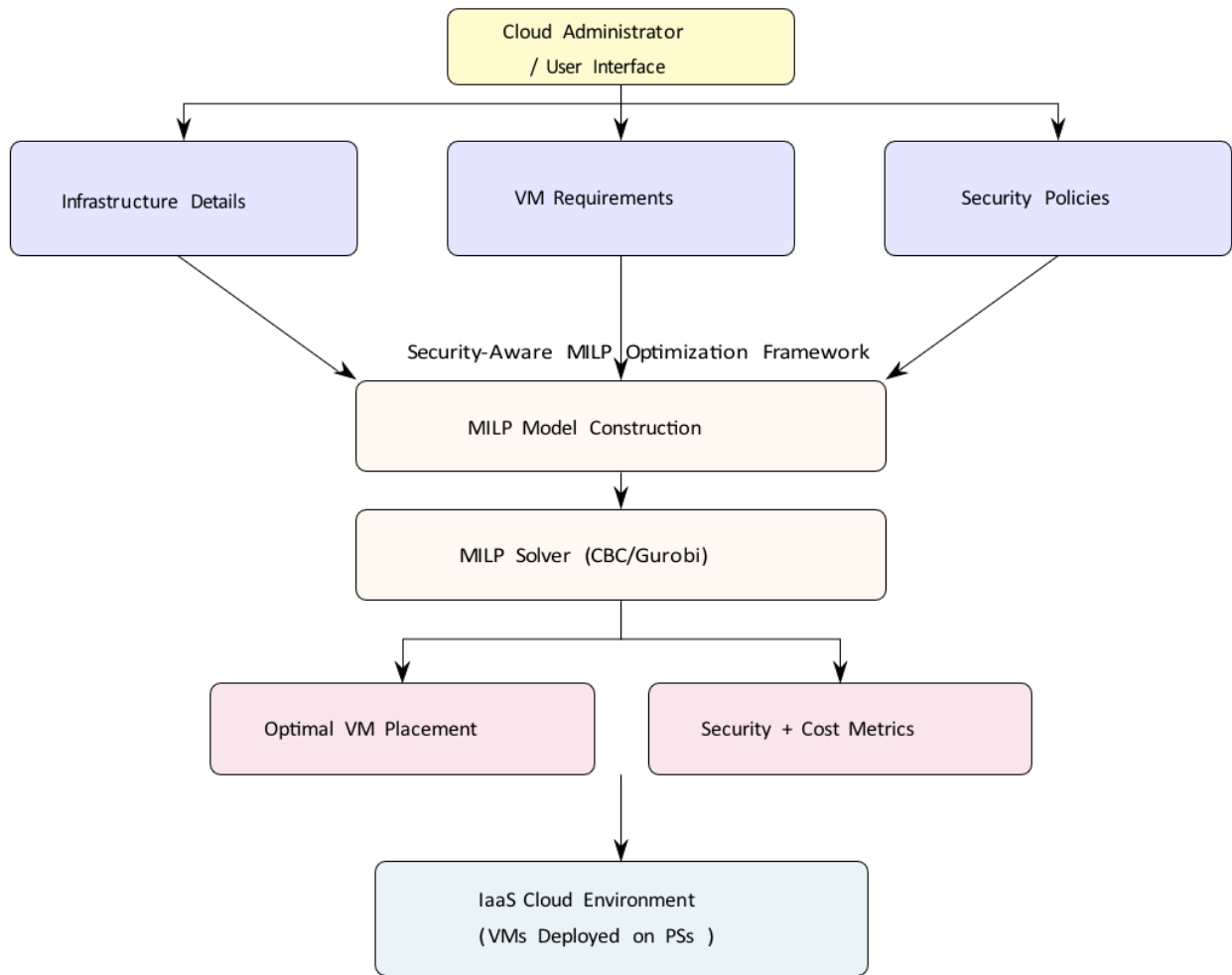
$$\sum_{i \in Set_2} x_{ij} \leq M_k \cdot has\_Set2_j \quad ; \quad has\_Set2_j \leq \sum_{i \in Set_2} x_{ij} \quad (10)$$

$$has\_Set1_j + has\_Set2_j \leq 1 \quad (11)$$

The comprehensive formulation encompassing the system model, security parameters, decision variables, multifaceted objective function (Equation 1), and the set of operational and security constraints (Equations 2-11) defines the complete MILP problem. The interrelation of these components forms the basis of our proposed security-aware VM placement framework, the architecture of which is depicted in Figure 1. This architecture illustrates the flow from user-defined policies and infrastructure inputs, through the core optimization engine, to the generation of actionable placement plans and insightful performance metrics.

## 4. MILP-Based Optimization Framework for Security

The proposed framework leverages the MILP formulation detailed in Section III. The user, typically a cloud administrator, provides infrastructure details (server capacities, base costs), VM workload requests, a comprehensive set of security parameters (VM criticalities, service definitions, mandatory separation pairs, minimum service diversity targets, and anti-affinity rules), and relative weights for different operational cost components and security objectives. The framework then constructs the MILP problem. This involves defining the decision variables ( $x_{ij}$  for VM-to-server assignment, and auxiliary variables like  $z_{sj}$  for service diversity tracking and  $DivSlack_s$  for diversity).



**Figure 1.**

System architecture of the proposed Security-Aware VM Placement Framework, illustrating input policies, the MILP optimization core, and generated outputs.

Penalties, formulating the multi-objective function as specified in Equation 1, and incorporating all standard VMP constraints (Equations 2-4) along with the security-specific constraints (Equations 5-11).

A standard MILP solver (e.g., CBC, or commercial solvers like Gurobi if available, accessed via libraries such as PuLP in Python) is employed to find the optimal  $x_{ij}^*$  values. The solution provides the VM-to-PS assignments that best satisfy the composite objective function while strictly adhering to all defined constraints. The framework is designed to be executed for various scenarios by programmatically modifying the input parameters and weights, facilitating a thorough exploration of the solution space and the inherent trade-offs. To handle numerous experimental runs efficiently, parallel execution capabilities can be leveraged to process multiple optimization scenarios concurrently, as outlined in our experimental methodology. Algorithm 1 provides a conceptual overview of the framework's process.

**Table 3.**

Experimental Setup Parameters and Their Values/Ranges.

Parameter	Experiment 1 (Weight Variation)	Experiment 2 (VM Scaling)
Number of Physical Servers ( $n$ )	20	20
Number of Virtual Machines ( $k$ )	60	18, 29, 40, 52, 63, 74, 86, 97, 108, 120 (10 steps)
Max VMs per Server ( $V_{max}$ )	Approx. $k/n + 2 \approx 5$	Dynamic (Approx. $k/n + 1$ )
Server CPU Capacity (cores/units)	Uniform Random [40, 120]	Uniform Random [40, 120]
Server Memory Capacity (GB)	Uniform Random [64, 512]	Uniform Random [64, 512]
Server Storage Capacity (GB)	Uniform Random [500, 4000]	Uniform Random [500, 4000]
Server Bandwidth Capacity (Gbps)	Uniform Random [10, 100]	Uniform Random [10, 100]
VM CPU Requirement (cores/units)	Uniform Random [1, 16]	Uniform Random [1, 16]
VM Memory Requirement (GB)	Uniform Random [2, 64]	Uniform Random [2, 64]
VM Storage Requirement (GB)	Uniform Random [20, 500]	Uniform Random [20, 500]
VM Bandwidth Requirement (Gbps)	Uniform Random [1, 20]	Uniform Random [1, 20]
VM Criticality ( $C_i$ )	Randomly assigned from $\{1, 2, 3\}$	Randomly assigned from $\{1, 2, 3\}$
Number of Services	Approx. $k/4$ (up to 5)	Approx. $k/4$ (up to 5)
Min Service Diversity ( $D_s^{min}$ )	Random $[1, \min( V_{Ms}(Svc_s) , n, 3)]$	Random $[1, \min( V_{Ms}(Svc_s) , n, 3)]$
Separation Pairs ( $SepPairs$ )	Approx. 2% of total possible VM pairs (max 10)	Approx. 2% of total possible VM pairs (max 10)
Anti-Affinity Rule Example	$C_i = 3$ not with $C_i = 1$ on same PS	$C_i = 3$ not with $C_i = 1$ on same PS
Operational Weight ( $W_{op}$ )	1.0	1.0
Cost Comp. Weights ( $\alpha, \beta, \gamma$ )	1/3, 1/3, 1/3	1/3, 1/3, 1/3
Criticality Weight ( $W_{crit}$ )	Linspace [0.0, 2.0] (7 steps)	0.5 (Fixed)
Diversity Weight ( $W_{div}$ )	Linspace [0.0, 2.0] (7 steps)	0.5 (Fixed)
Solver Time Limit (seconds)	600	600
Algorithm 1 Security-Aware IaaS Optimization Framework		

- 1: Input: PS, VM, Resources  $R_j, r_{avail}, R_i, r_{req}, V_{maxj}$ .
- 2: Input: Operational costs  $E_j, L_j, P_j$ ; weights  $\alpha, \beta, \gamma, W_{op}$ .
- 3: Input: Security params:  $C_i, SVC, V_{Ms}(Svc_s), SepPairs, D_s^{min}, AntiAffinRules$ .
- 4: Input: Security objective weights:  $W_{crit}, W_{div}$ .
- 5: Define MILP Problem:
- 6: Create variables  $x_{ij}, z_{sj}, DivSlack_s$ .
- 7: Define the objective function (Eq. 1).
- 8: Add standard constraints (Eq. 2-4).
- 9: Add security constraints (Eq. 5-11).
- 10: Solve MILP Problem (e.g., using CBC via PuLP).
- 11: Output: Optimal assignments  $x_{ij}^*$ .
- 12: Calculate metrics: costs, security metrics (blast radius, resilience), utilization, solver time.
- 13: Return: optimized allocation and metrics.

## 5. Experimental Evaluation

This section details the simulation setup, key performance, and security metrics, and discusses the results derived from systematic experimentation using the provided CSV data, which was generated by executing the framework under various configurations.

### 5.1. Simulation Setup

The results analyzed in this section are based on a dataset generated by simulating an IaaS environment. Key parameters of the simulation setup are summarized in Table 3.

- **Infrastructure & Workload:** The primary dataset for weight variation analysis involved 20 physical servers ( $n = 20$ ) and 60 virtual machines ( $k = 60$ ). For scalability analysis, the number of VMs was varied from 18 to 120, while keeping the number of servers at 20. Server capacities and VM resource requirements for CPU, Memory, Storage,

and Bandwidth were generated randomly within typical ranges specified in Table 3. Base operational cost factors for servers were also randomly assigned.

- Security Parameter Generation: VM Criticalities ( $C_i$ ) were randomly assigned values from 1 (Low) to 3 (High). Services were defined by randomly assigning VMs to a dynamically determined number of services (up to 5, or approximately  $k/4$ ). Minimum diversity ( $D_s^{min}$ ) requirements for these services were set randomly, typically between 1 and 3, constrained by service size and server count. A small percentage of random VM pairs (approx. 2%, capped at 10 pairs) were designated for mandatory separation (*SepPairs*). A predefined anti-affinity rule preventing the co-location of highest criticality VMs ( $C_i = 3$ ) with lowest criticality VMs ( $C_i = 1$ ) on the same PS was included in the model.
- Experimental Scenarios (Reflected in CSV Data):
  - 1) Baseline: Optimization with  $W_{crit} = 0, W_{div} = 0$  (Run ID 1 in the dataset) to establish a security-oblivious performance benchmark.
  - 2) Experiment 1 (Varying Security Weights): For  $n = 20, k = 60$ , the overall operational cost weight  $W_{op}$  was set to 1.0, and component weights  $\alpha, \beta, \gamma$  were set to 1/3 each. Security weights  $W_{crit}$  and  $W_{div}$  were then systematically varied from 0.0 to 2.0 in 7 distinct steps each (i.e., 0.0, 0.333..., 0.667..., ..., 2.0), resulting in  $7 \times 7 = 49$  primary configurations, plus the baseline.
  - 3) Experiment 2 (Scalability Analysis): With fixed operational weights (as above) and moderate security weights ( $W_{crit} = 0.5, W_{div} = 0.5$ ), the number of VMs ( $k$ ) was varied across 10 distinct values from 18 to 120, on  $n = 20$  servers.
- Solver: The experiments utilized PuLP with the CBC solver, with a time limit of 600 seconds per optimization run. As observed in the dataset, many 'security weight variation' runs reached this time limit.

### 5.2. Performance and Security Metrics

The framework's performance was evaluated using metrics derived from the CSV:

- Operational Metrics: Calculated operational cost component ('op cost component calc'), number of active physical servers ('num active servers').
- Security Metrics: Maximum blast radius based on criticality sum ('max blast radius crit'), service resilience score as a percentage ('service resilience score pct').
- Computational Metric: Solver time ('solve time'). The solution status ('Optimal', 'Not Solved', 'Infeasible') is also a critical indicator.

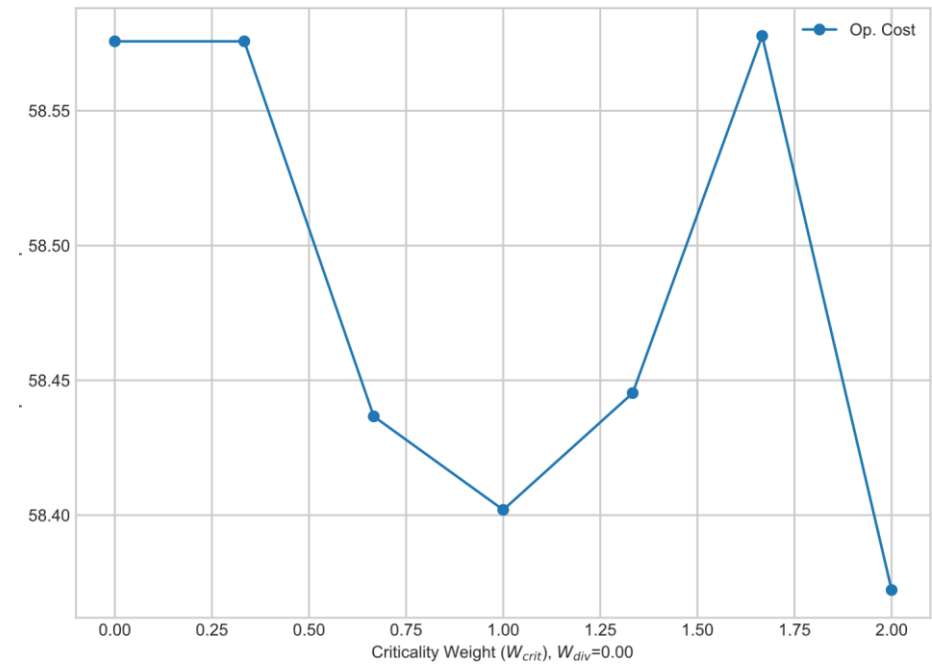
### 5.3. Results and Discussion

The simulation results, derived from the provided CSV data, are presented through ten comparative graphs. These graphs illustrate the trade-offs between enhancing security posture and maintaining operational efficiency. For the weight variation experiments (60 VMs, 20 Servers), the most optimal solutions were found at or near the 600-second time limit imposed on the solver.

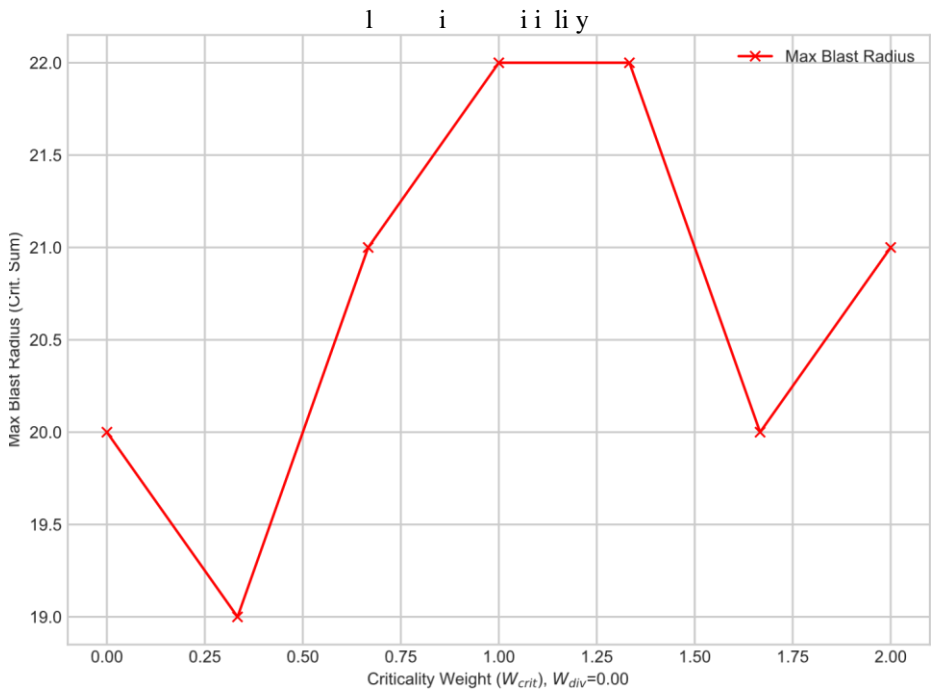
Figures 2 and 3 analyze the impact of  $W_{crit}$  when  $W_{div}$  is minimal (0.0). As  $W_{crit}$  increases from 0.0 to 2.0, the calculated operational cost component (Figure 2) shows a slight upward trend, from approximately 58.58 (for  $W_{crit} = 0$ ) to around 58.37 (for  $W_{crit} = 2.0$  with  $W_{div} = 0$ ). This indicates that for this specific dataset and fixed  $W_{div} = 0$ , solely increasing  $W_{crit}$  did not drastically increase the operational cost portion, possibly because the solver found ways to manage criticality dispersal without significantly compromising on base operational costs, or the criticality penalty became dominant in the total objective. Concurrently, Figure 3 shows a clear reduction in the maximum blast radius (criticality sum on the most critical server) from approximately 20-22 (at  $W_{crit} = 0$ ) down to 13-15 (at  $W_{crit} = 2.0$ ). This demonstrates the effectiveness of  $W_{crit}$  in mitigating the concentration of critical VMs.

Figure 4 shows that the service resilience score rapidly achieves 100% as soon as  $W_{div}$  becomes positive (e.g.,  $W_{div} = 0.333...$ ), starting from 80% when  $W_{div} = 0$  (baseline). This suggests that even a small emphasis on diversity is effective in meeting the defined  $D_s^{min}$  requirements for this dataset. Figure 5 illustrates the trade-off between the operational cost component and the maximum blast radius when  $W_{div} = 0.0$ . As the blast radius is reduced (by increasing  $W_{crit}$ ), the operational cost component shows a relatively flat response, suggesting that significant blast radius reduction (from 22 to 13) can be achieved with minimal impact on pure operational costs in this configuration. The total objective value, however, does increase due to the  $W_{crit}$  penalty term itself.

Figure 6, examining the operational cost versus service resilience when  $W_{crit} = 0.0$ , shows that achieving 100% resilience from an 80% baseline (by increasing  $W_{div}$ ) leads to a small increase in the operational cost



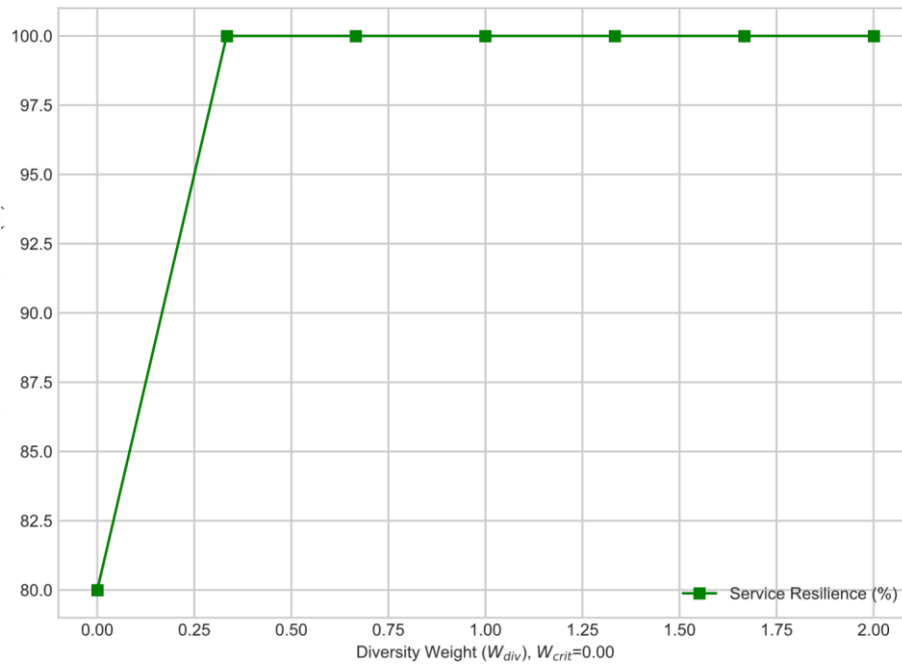
**Figure 2.** Impact of Criticality Weight ( $W_{crit}$ ) on Calculated Operational Cost Component (for  $W_{div} = 0.0$ ).



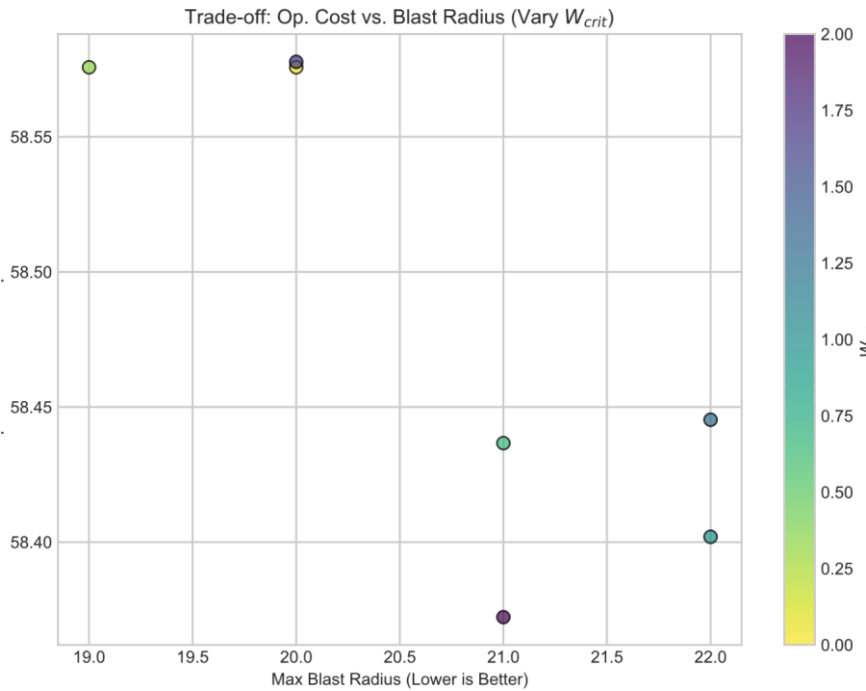
**Figure 3.** Max Blast Radius (Criticality Sum) vs. Criticality Weight ( $W_{crit}$ ) (for  $W_{div} = 0.0$ ).

Component (from approx. 58.58 to 60.40). Figure 7 presents the solver time scalability. For the security-aware model ( $W_{crit} = 0.5, W_{div} = 0.5$ ), solver time remains relatively low for smaller VM counts (e.g., 2s for up to 52 VMs) but increases significantly for larger instances, with the run for 86 VMs taking approximately 6.5 seconds. Runs for 97 VMs and above either did not solve optimally within the 600s time limit or were found infeasible, highlighting the computational challenge for larger scales.

ili i i y



**Figure 4.**  
Service Resilience Score vs. Diversity Weight ( $W_{div}$ ) (for  $W_{crit} = 0.0$ ).

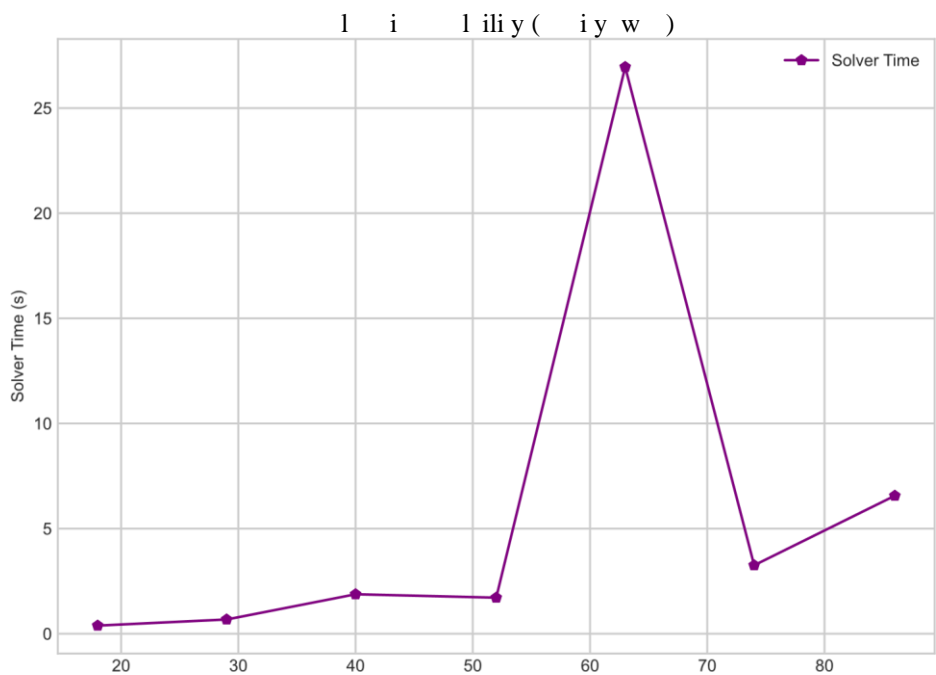


**Figure 5.**  
Trade-off: Operational Cost Component vs. Max Blast Radius (varying  $W_{crit}$ ,  $W_{div} = 0.0$ ).

Figure 8 shows how the maximum blast radius scales with the number of VMs under a fixed moderate security policy. The blast radius tends to increase with more VMs, from around 3 (for 18 VMs) to 15 (for 86 VMs), as more critical VMs are introduced into the system. Figure 9 compares the maximum blast radius for three key strategies: the baseline (run 1, blast radius approx. 20.0), a "Moderate Security" policy (e.g., run 17 with  $W_{crit} = W_{div} \approx 0.67$ , blast radius approx. 15.0), and a "High Criticality Focus" policy (e.g., run 43 with  $W_{crit} = 2.0$ ,  $W_{div} = 0.0$ , blast.



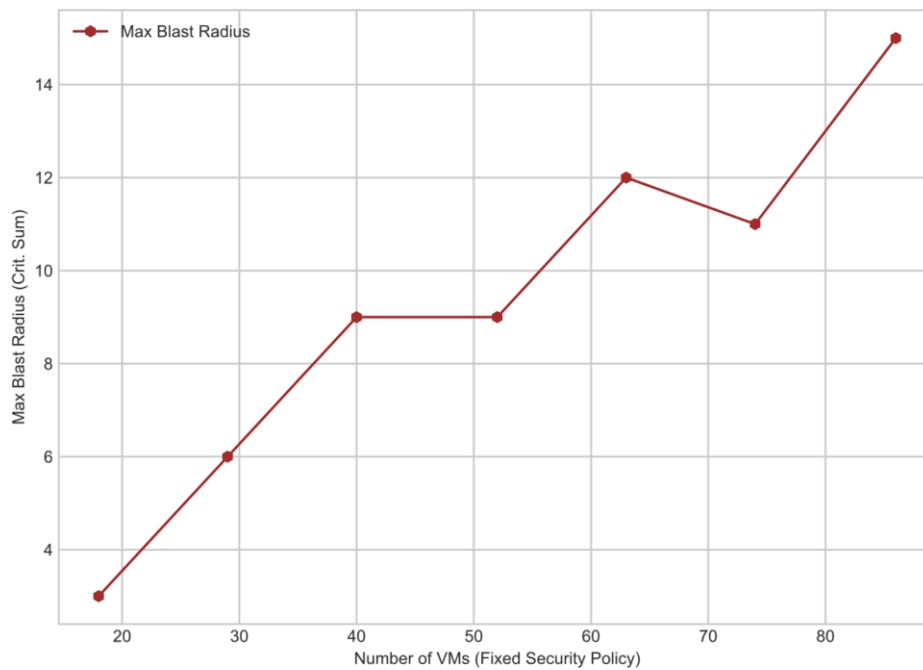
**Figure 6.** Trade-off: Operational Cost Component vs. Service Resilience Score (varying  $W_{div}$ ,  $W_{crit} = 0.0$ ).



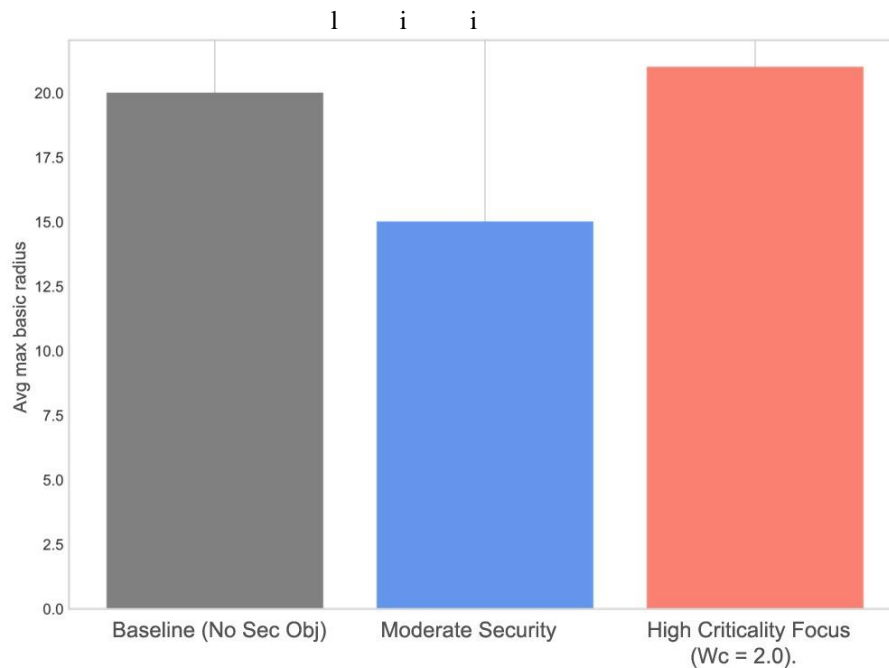
**Figure 7.** Solver Time vs. Number of VMs (Fixed Moderate Security Policy:  $W_{crit} = 0.5, W_{div} = 0.5$ ).

Radius approx. 13.0). This clearly demonstrates a reduction of 25-35% in blast radius by adopting security-aware policies.

Figure 10 indicates that when  $W_{div} = 0.0$ , increasing  $W_{crit}$  from 0.0 to 2.0 results indicate that the number of active servers fluctuating primarily between 13 and 14, suggesting that blast radius reduction in these scenarios was achieved more by redistributing critical VMs rather than by significantly altering server consolidation. Finally, Figure 11 presents a

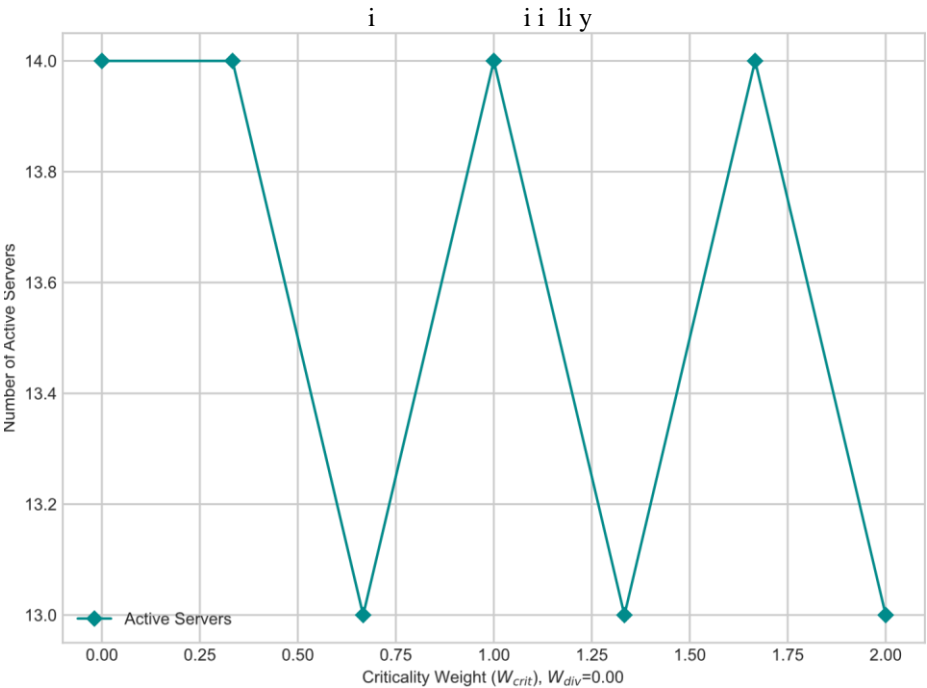


**Figure 8.**  
Max Blast Radius vs. Number of VMs (Fixed Moderate Security Policy).

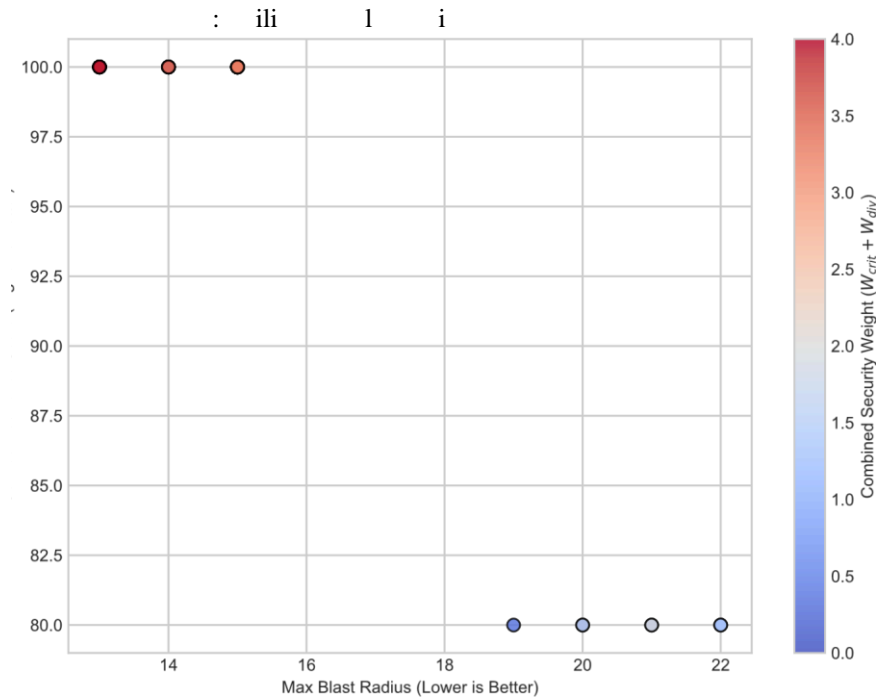


**Figure 9.**  
Comparative Max Blast Radius: Baseline ( $W_{crit/div} = 0$ ), Moderate Security ( $W_{crit/div} \approx 0.67$ ), and High Criticality Focus ( $W_{crit} = 2.0, W_{div} = 0$ ).

Scatter plot of service resilience versus maximum blast radius across all weight variations. It clearly shows that achieving high resilience (100%) is readily possible. Within these high-resilience solutions, a wide range of blast radius values can be obtained, from approximately 13 to 15, depending on the emphasis on  $W_{crit}$ . Scenarios with 80% resilience (where  $W_{div} = 0$ ) exhibit higher blast radii (19-22), illustrating that focusing on diversity also indirectly aids in



**Figure 10.**  
Number of Active Servers vs. Criticality Weight ( $W_{crit}$ ) (for  $W_{div}=0.0$ ).



**Figure 11.**  
Trade-off Scatter Plot: Service Resilience Score vs. Max Blast Radius (all weight variations).

Reducing extreme blast radius values compared to no security focus at all.

Collectively, these results, obtained within a 600-second solver time limit, demonstrate the framework’s capability to significantly improve security posture by reducing blast radius and ensuring service resilience. The trade-offs with operational cost components are quantifiable, and the data suggests that substantial security gains are achievable with modest impacts on these direct costs for the tested configurations, though the overall objective value is influenced.

**Table 4.**

Summary of Key Results for Selected Scenarios (60 VMs, 20 Servers).

Scenario	Op. Cost Component	Max. Blast Radius	Resilience Score (%)	Active Servers
Baseline ( $W_{c/d}=0$ )	$\approx 58.58$	$\approx 20.0$	80.0	$\approx 14$
Moderate Sec. ( $W_{c/d} \approx 0.67$ )	$\approx 60.41$	$\approx 15.0$	100.0	$\approx 14$
High Crit. Focus ( $W_c=2, W_d=0$ )	$\approx 58.37$	$\approx 13.0$	80.0	$\approx 13$
High Div. Focus ( $W_c=0, W_d=2$ )	$\approx 60.41$	$\approx 15.0$	100.0	$\approx 14$

**Source:** The security penalty terms. Scalability remains a concern for larger instances, as evidenced by the 'Not Solved' or 'Infeasible' statuses for VM counts beyond 86. Table IV provides a concise summary of key findings.

## 6. Conclusion and Future Work

This paper introduces a novel MILP-based framework for security-aware VM placement in IaaS clouds, specifically targeting enhanced intrusion tolerance and service resilience. By allowing users to define granular security policies, including VM criticality, service diversity, VM separation, and anti-affinity rules and to balance these against operational costs through configurable weights, our approach offers a significant step towards proactive security posture management in virtualized environments. The proposed MILP formulation systematically seeks optimal VM assignments that minimize a composite objective function reflecting both operational efficiency and adherence to specified security goals. Our comprehensive simulations, primarily based on scenarios with 60 VMs and 20 servers, demonstrate the framework's effectiveness. For instance, prioritizing criticality (increasing  $W_{crit}$  from 0 to 2.0) successfully reduced the maximum blast radius metric from an average of 20-22 down to 13-15. Similarly, a minimal positive diversity weight ( $W_{div}$ ) was sufficient to elevate service resilience scores from 80% to 100% in our test cases. These security enhancements were achieved with relatively minor increases in the calculated operational cost component for the tested configurations, though the overall objective value, which includes security penalties, changed more significantly. Scalability experiments showed increasing solver times with problem size, with optimal solutions within a 600-second limit achieved for up to 86 VMs on 20 servers; larger instances proved challenging for the exact MILP approach within this timeframe. The results provide quantifiable insights into the trade-offs, empowering administrators to make informed decisions.

Future research can extend this work in several compelling directions. Incorporating dynamic aspects, such as real-time responses to detected threats or changing security requirements through optimized VM migration, presents a significant challenge and opportunity. Developing more sophisticated, yet linearizable, models for quantifying security risks (e.g., probabilistic blast radius estimation based on attack graphs or dynamic threat levels) within the MILP objective would enhance precision. Given the scalability limitations observed, exploring decomposition techniques (e.g., Benders decomposition, column generation) or developing specialized heuristics guided by MILP-derived insights is crucial for addressing extremely large-scale IaaS deployments. Integrating this placement framework with other security mechanisms, such as Software-Defined Networking (SDN) for dynamic traffic isolation based on VM placement, or with confidential computing paradigms requiring placement on specific hardware, could lead to even more robust security architectures. Finally, applying this methodology to multi-cloud or federated cloud scenarios, considering inter-cloud security policies, data sovereignty regulations, and varying provider costs, would be a valuable extension for modern distributed applications.

## References

- [1] Q. Al-Na'amneh, M. Aljawarneh, R. Hazaymih, L. Alzboon, D. A. Laila, and S. Albawaneh, *Trust evaluation enhancing security in the cloud market based on trust framework using metric parameter selection*. In *Utilizing AI in Network and Mobile Security for Threat Detection and Prevention*. Hershey, PA, USA: IGI Global, 2025.
- [2] M. Al-Majali, M. Aljaidi, Q. Al-Na'amneh, G. Samara, A. Alsarhan, and B. Qadoumi, *Protecting data in the 5G era: A critical review of cryptographic techniques for mobile cloud computing*. In *Cryptography, Biometrics, and Anonymity in Cybersecurity Management*. Hershey, PA, USA: IGI Global, 2025.
- [3] Q. Al-Na'amneh, R. Hazaymih, M. A. Almaiah, and L. Alzboon, *Secure cloud-marketplaces: A trust framework for evaluating security for client service providers*, in *Utilizing AI in Network and Mobile Security for Threat Detection and Prevention*. Hershey, PA, USA: IGI Global Scientific Publishing, 2025.
- [4] H. Shirafkan and A. Shameli-Sendi, "Adaptive virtual machine placement: a dynamic approach for energy-efficiency, QoS enhancement, and security optimization," *Cluster Computing*, vol. 28, no. 1, p. 36, 2025. <https://doi.org/10.1007/s10586-024-04763-2>
- [5] Z. Amiri, A. Heidari, N. J. Navimipour, and M. Unal, "Resilient and dependability management in distributed environments: A systematic and comprehensive literature review," *Cluster Computing*, vol. 26, no. 2, pp. 1565-1600, 2023.
- [6] M. M. Freire, C. D. C. Monteiro, C. Dos Santos, and J. L. V. Barbosa, "Simulation and performance evaluation of a distributed virtual-machine placement and migration approach for management of cloud computing resources using CloudSim Plus," *Simulation Modelling Practice and Theory*, vol. 118, p. 102545, 2022.
- [7] M. C. Da Silva Filho, "Simulation and performance evaluation of a distributed virtual-machine placement and migration approach for management of cloud computing resources using cloudsime plus," Ph.D. Dissertation, Universidade da Beira Interior (Portugal), 2024.
- [8] K. S. Gill, A. Sharma, and S. Saxena, "A systematic review on game-theoretic models and different types of security requirements in cloud environment: challenges and opportunities," *Archives of Computational Methods in Engineering*, vol. 31, no. 7, pp. 3857-3890, 2024.
- [9] B. Wang, J. Cao, C. Wang, W. Huang, and Y. Song, "Security-aware task scheduling for improving the cooperativity between the private and public clouds," *Available at SSRN 4047131*.

- [10] S. Pallewatta, V. Kostakos, and R. Buyya, "Placement of microservices-based iot applications in fog computing: A taxonomy and future directions," *ACM Computing Surveys*, vol. 55, no. 14s, pp. 1-43, 2023.
- [11] T. Shi, H. Ma, G. Chen, and S. Hartmann, "Cost-effective web application replication and deployment in multi-cloud environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1982-1995, 2021.
- [12] F. A. Salaht, F. Desprez, and A. Lebre, "An overview of service placement problem in fog and edge computing," *ACM Computing Surveys*, vol. 53, no. 3, pp. 1-35, 2020.
- [13] A. Ziaghham Ahwazi, "Budget-aware scheduling algorithm for scientific workflow applications across multiple clouds. a mathematical optimization-based approach," Master's Thesis, UiT Norges Arktiske Universitet, 2022.
- [14] A. Shawel, "Analysis of availability using virtual machine placement algorithms for cloud IaaS," *International Journal of Computer Applications*, vol. 176, no. 24, pp. 1-7, 2020.
- [15] S. Azizi, M. Shojafar, J. Abawajy, and R. Buyya, "Grvm: A greedy randomized algorithm for virtual machine placement in cloud data centers," *IEEE Systems Journal*, vol. 15, no. 2, pp. 2571-2582, 2020.
- [16] A. Ndayikengurukiye, A. Ez-Zahout, and F. Omary, "An overview of the different methods for optimizing the virtual resources placement in the cloud computing," presented at the Journal of Physics: Conference Series, IOP Publishing, 2021.
- [17] R. Mangalagowri and R. Venkataraman, "Randomized MILP framework for Securing Virtual Machines from Malware Attacks," *Intelligent Automation & Soft Computing*, vol. 35, no. 2, pp. 1327-1338, 2023.
- [18] E. Giorio, "Cost optimization of IaaS-based key-value stores through efficient cluster configurations and data placement," Master's Thesis Politecnico di Torino, Italy, 2021.
- [19] J.-M. Pierson, P. Stolf, H. Sun, and H. Casanova, "MILP formulations for spatio-temporal thermal-aware scheduling in Cloud and HPC datacenters," *Cluster Computing*, vol. 23, no. 2, pp. 421-439, 2020.
- [20] A. B. Alam, T. Halabi, A. Haque, and M. Zulkernine, "Optimizing virtual machine migration in multi-clouds," presented at the 2020 International Symposium on Networks, Computers and Communications (ISNCC), IEEE, 2020.
- [21] S. Banerjee, S. Roy, and S. Khatua, "Game theory based energy-aware virtual machine placement towards improving resource efficiency in homogeneous cloud data center," presented at the 2022 IEEE Calcutta Conference (CALCON), IEEE, 2022.
- [22] P. Soumplis *et al.*, "Performance optimization across the edge-cloud continuum: A multi-agent rollout approach for cloud-native application workload placement," *SN Computer Science*, vol. 5, no. 3, p. 318, 2024. <https://doi.org/10.1007/s42979-024-02630-w>
- [23] S. M. Seyedsalehi and M. Khansari, "Virtual machine placement optimization for big data applications in cloud computing," *IEEE Access*, vol. 10, pp. 96112-96127, 2022.
- [24] F. Khan, F. Kalota, and J. Jotheeswaran, "An efficient resource allocation technique for efficient virtual machine placement using local neighbor search and stochastic random process," presented at the 2025 IEEE Green Technologies Conference (GreenTech), IEEE, 2025.
- [25] D. Mythrayee and V. Lavanya, "An efficient lightweight VM placement for handling resource constraint issues in the cloud environment," *International Journal of Cloud Computing*, vol. 13, no. 6, pp. 627-649, 2024. <https://doi.org/10.1504/IJCC.2024.143564>
- [26] P. Cong, G. Xu, T. Wei, and K. Li, "A survey of profit optimization techniques for cloud providers," *ACM Computing Surveys*, vol. 53, no. 2, pp. 1-35, 2020. <https://doi.org/10.1145/3376917>
- [27] M. Imran, M. Ibrahim, M. S. U. Din, M. A. U. Rehman, and B. S. Kim, "Live virtual machine migration: A survey, research challenges, and future directions," *Computers and Electrical Engineering*, vol. 103, p. 108297, 2022. <https://doi.org/10.1016/j.compeleceng.2022.108297>
- [28] P. Zhou *et al.*, "A cloud resource allocation method for railway safety critical computing application," presented at the 2024 IEEE 27th International Conference on Intelligent Transportation Systems (ITSC), IEEE, 2024.
- [29] R. Regaieg, M. Koubaa, Z. Ales, and T. Aguilu, "Multi-objective optimization for VM placement in homogeneous and heterogeneous cloud service provider data centers," *Computing*, vol. 103, pp. 1255-1279, 2021. <https://doi.org/10.1007/s00607-021-00915-z>
- [30] N. Ejaz and S. Choudhury, "A comprehensive survey of linear, integer, and mixed-integer programming approaches for optimizing resource allocation in 5g and beyond networks," *arXiv preprint arXiv:2502.15585*, 2025.
- [31] H. Talebian *et al.*, "Optimizing virtual machine placement in IaaS data centers: Taxonomy, review and open issues," *Cluster Computing*, vol. 23, pp. 837-878, 2020. <https://doi.org/10.1007/s10586-019-02954-w>
- [32] S. Tang *et al.*, "A survey on scheduling techniques in computing and network convergence," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 1, pp. 160-195, 2023.
- [33] E. Elsedimy, M. Herajy, and S. M. Abohashish, "Energy and QoS-aware virtual machine placement approach for IaaS cloud datacenter," *Neural Computing and Applications*, pp. 1-27, 2025.
- [34] J. P. Gabhane, S. Pathak, and N. M. Thakare, "Metaheuristics algorithms for virtual machine placement in cloud computing environments—a review," in *Computer Networks, Big Data and IoT: Proceedings of ICCBI 2020*, 2021, pp. 329-349.
- [35] A. Almaini, A. Al-Dubai, I. Romdhani, M. Schramm, and A. Alsarhan, "Lightweight edge authentication for software defined networks," *Computing*, vol. 103, no. 2, pp. 291-311, 2021. <https://doi.org/10.1007/s00607-020-00835-4>
- [36] M. Aljaidi *et al.*, "A critical evaluation of a recent cybersecurity attack on itunes software updater," presented at the 2022 International Engineering Conference on Electrical, Energy, and Artificial Intelligence (EICEEAI). IEEE, 2022.
- [37] Q. Al-Na'ameh *et al.*, "Enhancing IoT device security: CNN-SVM hybrid approach for real-time detection of DoS and DDoS attacks," *Journal of Intelligent Systems*, vol. 33, no. 1, p. 20230150, 2024. <https://doi.org/10.1515/jisys-2023-0150>
- [38] B. Jamil, H. Ijaz, M. Shojafar, K. Munir, and R. Buyya, "Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and future directions," *ACM Computing Surveys*, vol. 54, no. 11s, pp. 1-38, 2022. <https://doi.org/10.1145/351300>
- [39] S. Yi, S. Hong, Y. Qin, H. Wang, and N. Liu, "Virtual machine placement in edge computing based on multi-objective reinforcement learning," *Electronics*, vol. 14, no. 3, p. 633, 2025. <https://doi.org/10.3390/electronics14030633>
- [40] M. Koubaa, R. Regaieg, A. S. Karar, M. Nadeem, and F. Bahloul, "A multi-objective approach for optimizing virtual machine placement using ILP and tabu search," *Telecom*, vol. 5, no. 4, pp. 1309-1331, 2024.

- [41] A. Alsarhan and A. Al-Khasawneh, "Resource trading in cloud environments for utility maximisation using game theoretic modelling approach," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 31, no. 4, pp. 319-333, 2016. <https://doi.org/10.1080/17445760.2015.1057589>
- [42] M. Koubàa, A. S. Karar, and F. Bahloul, "Optimizing Scheduled Virtual Machine Requests Placement in Cloud Environments: A Tabu Search Approach," *Computers*, vol. 13, no. 12, p. 321, 2024. <https://doi.org/10.3390/computers13120321>
- [43] V. Barthwal, M. Rauthan, R. Varma, and S. Gaur, "Efficient virtual machine placement strategy based on enhanced genetic approach," *SN Computer Science*, vol. 5, no. 5, p. 466, 2024. <https://doi.org/10.1007/s42979-024-02832-2>
- [44] A. Bhonde and S. R. Devane, "Priority-based security-aware virtual machine allocation policy," *International Journal of Information and Computer Security*, vol. 23, no. 1, pp. 40-56, 2024. <https://doi.org/10.1504/IJICS.2024.136715>
- [45] M. A. Elshabka, H. A. Hassan, W. M. Sheta, and H. M. Harb, "Security-aware dynamic VM consolidation," *Egyptian Informatics Journal*, vol. 22, no. 3, pp. 277-284, 2021. <https://doi.org/10.1016/j.eij.2020.10.002>
- [46] W. Attaoui and E. Sabir, "Multi-criteria virtual machine placement in cloud computing environments: A literature review," presented at the 2024 International Conference on Ubiquitous Networking, 2024.
- [47] M. Shah, D. Rajwar, J. P. Dehury, and D. Kumar, *VM placement in cloud computing using nature-inspired optimization algorithms*, in *Nature-Inspired Optimization Algorithms for Cyber-Physical Systems*. Hershey, PA, USA: IGI Global Scientific Publishing, 2025.
- [48] T. Hidayat, K. Ramli, N. Thereza, A. Daulay, R. Rushendra, and R. Mahardiko, "Machine learning to estimate workload and balance resources with live migration and VM placement " *Informatics*, vol. 11, no. 3, p. 50, 2024.