Optimizing Resource Allocation in the Edge: A Minimum Weighted Vertex Cover Approach

Antonios Makris*
Department of Informatics and Telematics,
Harokopio University of Athens &
School of Electrical and Computer Engineering,
National Technical University of Athens
Greece
amakris@hua.gr

Emmanouil Maragkoudakis
Department of Informatics and Telematics,
Harokopio University of Athens
Greece
csi22304@hua.gr

Ioannis Kontopoulos
Department of Informatics and Telematics,
Harokopio University of Athens &
School of Electrical and Computer Engineering,
National Technical University of Athens
Greece
kontopoulos@hua.gr

Theodoros Theodoropoulos
Department of Informatics and Telematics,
Harokopio University of Athens &
School of Electrical and Computer Engineering,
National Technical University of Athens
Greece
ttheod@hua.gr

Ioannis Korontanis
Department of Informatics and Telematics,
Harokopio University of Athens &
School of Electrical and Computer Engineering,
National Technical University of Athens
Greece
gkorod@hua.gr

Emanuele Carlini
Institute of Information Science and Technologies, National Research Council (CNR), Pisa
Italy
emanuele.carlini@isti.cnr.it

Matteo Mordacchini
Institute of Information Science and Technologies, National Research Council (CNR), Pisa
Italy
matteo.mordacchini@isti.cnr.it

Patrizio Dazzi
Department of Computer Science, University of Pisa
Italy
patrizio.dazzi@unipi.it

Theodora Varvarigou
School of Electrical and Computer Engineering,
National Technical University of Athens
Greece
dora@telecom.ntua.gr

ABSTRACT

The transition from Cloud Computing to a Cloud-Edge continuum introduces novel opportunities and challenges for data-intensive and interactive Next Generation applications. Strategies that were effective in the Cloud environment now necessitate reconsideration to adapt to the Edge's distributed, dynamic, and heterogeneous ecosystem. Proactively planning the placement of application images becomes crucial to minimize image transfer time, align with the dynamic nature of the system, and meet the strict demands of applications. In this regard, this paper proposes an empirical experimental analysis, by comparing the results of different placement strategies and various network topologies. In particular, we model the problem of proactive placement of application images as a Minimum Weighted Vertex Cover problem. Our results demonstrate that the Greedy approach seems to offer the optimal tradeoff with respect to the number of allocated images and execution time.

*Corresponding author.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

CCS CONCEPTS

• **Computing methodologies** → *Simulation evaluation*; • **Computer systems organization** → *Availability*; • **Theory of computation** → **Graph algorithms analysis**; *Approximation algorithms analysis.*

KEYWORDS

Edge Computing, Cloud Computing, Proactive Image Placement, Application Placement, Optimization Problem

ACM Reference Format:
Antonios Makris, Emmanouil Maragkoudakis, Ioannis Kontopoulos, Theodoros Theodoropoulos, Ioannis Korontanis, Emanuele Carlini, Matteo Mordacchini, Patrizio Dazzi, and Theodora Varvarigou. 2018. Optimizing Resource Allocation in the Edge: A Minimum Weighted Vertex Cover Approach. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The proliferation of data-intensive applications and latency-sensitive services, also referred to as NextGen applications, is driving the adoption of edge computing architectures [17, 20, 35, 36]. NextGen applications, encompassing data analytics, artificial intelligence, augmented/virtual reality, and Internet of Things (IoT) platforms, demand real-time processing and low latency for responsive and immersive user experiences. However, centralized cloud computing models face challenges in meeting these demands due to constraints related to bandwidth, network latency, location-awareness, mobility support, security, and privacy.

In response to these challenges, edge computing emerges as a solution by facilitating computation and data storage at the network edge, closer to end users and data sources [28]. Rather than sending all data to distant cloud servers, edge computing allows localized analysis and decision-making on resource-constrained edge devices [21, 23, 37]. This allows latency-critical tasks to be executed within milliseconds, addressing issues associated with high bandwidth consumption, intermittent connectivity, and single points of failure in cloud-only architectures [22].

A prevalent design pattern for NextGen edge applications involves microservice-based architectures, leveraging lightweight and modular, independently deployable services [3, 24]. Decomposing applications into granular, loosely-coupled microservices offers flexibility for rapid updates without disrupting other components. This also enables services to be dynamically instantiated on demand based on real-time user presence, workload shifts and resource availability.

However, edge infrastructures pose unique challenges for deploying and managing microservices at scale, given the limited resources of edge nodes compared to data centers. Additionally, the dynamic nature of edge workloads demands fast and adaptive service provisioning. Naively downloading application images for each service instance activation from centralized repositories fails to meet these requirements, due to possible bandwidth costs and transfer time.

This work focuses on addressing these challenges through a distributed edge image management approach. In particular, we model the problem of proactive image placement of application images as a Minimum Weighted Vertex Cover (MWVC) which enables a concise and precise representation, effectively reducing the complexity of the problem formulation. The goal is to balance the storage usage with the latency induced by image transfer. MWVC has a well-established theoretical foundation in graph theory, with a plethora of research and efficient algorithms tailored to discover the minimum weighted vertex cover set in diverse graph topologies. By leveraging this rich body of knowledge, we can benefit from proven techniques and algorithms to solve the placement problem optimally or approximately. Additionally, MWVC-based solutions offer resource optimization benefits. Selecting the minimum weighted vertex cover guarantees the involvement of the fewest number of nodes required to cover all edges, resulting in efficient resource allocation. This choice minimizes the computational and memory overhead associated with the placement process, ultimately enhancing the overall performance of the system.

We have considered two algorithmic implementations of the problem, the Greedy and Genetic algorithm, aiming to determine which approach exhibits superior performance. We run these implementations against different network topologies, simulating various possible Edge Computing environments. The experimental results demonstrate that the Greedy implementation offers the best trade-off for the given problem with respect to the number of allocated images and execution time.

The remaining of this paper is organized as follows. Section 2 serves as a literature review in the field of optimal image placement in Edge computing environments. Section 3 defines the problem formulation and the proposed approach for the set cover problem. Section 4 presents the experimental evaluation, the simulation methodology, and discusses the experimental result. Finally, Section 5 summarizes the merits of our work and highlights some perspectives that require further attention in the future.

2 RELATED WORK

The effective placement of application images in the dynamic and heterogeneous Cloud-Edge Continuum is a crucial concern [9, 16]. Addressing this challenge requires the application of diverse solution methodologies based on optimization techniques, including Integer Programming [15, 25], Markov Decision Processes [29], and stochastic optimization [26]. Various versions of the Integer Programming paradigm are discussed in [15, 25, 32]. In [42], authors delve into the use of Integer Nonlinear Programming in Fog Computing, while [6] recommends Mixed Integer Linear Programming. Another approach to formulating the image placement problem involves the implementation of Constrained Optimization, as outlined in [2]. Constrained Optimization involves methods for determining optimal values for specific variables while adhering to specified constraints. Additionally, [1] suggests that Markov Decision Processes [29], stochastic optimization [26], and general convex optimization are potential solutions to effectively address the challenges of formalizing the image placement problem.

Moreover, another crucial aspect observed in various research studies within the scientific literature is the framing of the optimization problem within the utilized metrics. In [19, 34, 40], the emphasis was on minimizing service latency in the image placement process. Low latency is considered essential for delay-sensitive applications, directing the majority of scientific efforts towards utilizing latency as the primary metric. Additionally, [31] explores Reinforcement Learning techniques to extend Kubernetes, enabling the deployment and replication of delay-sensitive containerized services in a geographically distributed system. Another proposed solution, ICON, is introduced in [44], where autonomous containers collaborate to find the optimal allocation of their services in terms of latency. Given the nature of Edge and Cloud infrastructures, minimizing operational expenses emerges as a pivotal concern.

Therefore, it becomes evident that the cost associated with image placement is another measure that should be considered, as highlighted in [5]. Furthermore, resource utilization is identified as another crucial metric explored within optimization strategies, as evidenced in [33] and [27]. Lastly, the congestion rate is an additional metric deserving exploration. Yu et al. [43] investigated the congestion ratio, exploring its potential as the minimum ratio between flow and link capacity to accommodate service placement.

Furthermore, prior efforts have emphasized reactive image placement strategies activated in response to service requests within Edge computing environments. However, given the intricate functionalities and stringent Quality of Service (QoS) requirements associated with contemporary services, this reactive paradigm appears inadequate. Our focus is on implementing a proactive approach, a relatively unexplored area within the scientific community. Only a few exceptions, such as the work in [30] and [12], have explored proactive image placement. For instance, in [30], a Reinforcement Learning-based mechanism proactively deploys microservices on edge servers, taking into consideration the structure of the microservice graph application. Similarly, in [12], efforts aimed at establishing a service placement and migration model leveraging mobility prediction in Mobile Edge Computing (MEC). However, these studies primarily focus on specific solutions rather than conducting a comprehensive analysis of the potential effects of various algorithmic approaches or network topologies on proactive image placement. In our recent work [38], we modeled the problem of proactive application image placement as a Minimum Vertex Cover. Recognizing the importance of considering weighted edges, in this paper, we further extended our model to Minimum Weighted Vertex Cover (MWVC).

3 PROBLEM FORMULATION

The Minimum Weighted Vertex Cover (MWVC) problem is a classical NP-hard optimization problem in graph theory. Given an undirected weighted graph, the minimum weighted vertex cover problem is to find the vertex cover with the minimum sum of weights of the constituent vertices. A slightly different formulation of the problem and the introduction of certain constraints can transform the proactive image placement problem into a set cover problem. More specifically, the MWVC algorithm assigns weights to each vertex in the graph based on two key factors:

- Average Latency ($L(i)$): The average latency of edges connected to a vertex. Lower latency indicates a better choice for inclusion in the vertex cover.
- Available Storage ($S(i)$): The available storage capacity of the device associated with each vertex. A device with more available storage is more likely to be included in the vertex cover.

The weight $w(i)$ assigned to each vertex i is calculated using the following formula:

$$w(i) = c_1 \cdot L(i) + c_2 \cdot \left(1 - \frac{S(i)}{\text{maxStorage}}\right)$$

where:

- $w(i)$ represents the weight of vertex i
- $L(i)$ represents the average latency of edges connected to vertex i
- $S(i)$ denotes the available storage capacity of the device associated with vertex i
- maxStorage is the maximum available storage among all devices in the graph
- c_1 and c_2 are constant values representing the weights assigned to the latency and storage terms respectively. They are used to adjust the contribution of each term to the overall weight calculation.

The term $1 - \frac{S(i)}{\text{maxStorage}}$ normalizes the available storage by the maximum storage capacity (maxStorage) across all nodes.

The objective of the optimization problem is to minimize the total weighted sum of selected vertices in the vertex cover:

$$\text{Minimize} \sum_{i \in V} w(i) x_i$$

subject to the constraints that ensure at least one vertex incident to each edge is included in the vertex cover:

$$\forall \{u, v\} \in E : \quad x_u + x_v \geq 1$$

Additionally, x_i is a binary variable representing whether vertex i is part of the vertex cover (1) or not (0).

This formula combines the factors of latency and available storage to assign a weight to each vertex, considering both aspects in the MWVC algorithm. Higher weights indicate a less favorable choice for inclusion in the MWVC set.

4 EXPERIMENTAL EVALUATION

4.1 Simulation Methodology

To simulate different network topologies, as well as image placement and image transfers among network nodes, Python scripts were developed and utilized. More specifically, the NetworkX Python package [13] was leveraged to generate diverse network topologies with varying parameters. This package allows for the generation of various network topologies and provides objects that streamline the storage and manipulation of node and edge attributes.

The assessed algorithms' performance, i.e., Greedy and Genetic (Section 4.2) is evaluated across four different network graph topologies. These topologies are simulated at different scales, spanning from 64 to 1024 vertices, including $V = [64, 128, 256, 512, 1024]$. Nevertheless, the number of edges in each graph exhibits significant variation due to the unique characteristics, input parameters, and connectivity properties inherent in each topology. This variability stems from the different number of vertices. The network topologies employed in this research work are:

Full r-ary tree. A full r-ary tree is a hierarchical tree data structure where each internal node has exactly r children, except possibly for the last internal node, which may have fewer children if the total number of nodes is not a multiple of r. This structure ensures a uniform distribution of nodes across levels, optimizing space utilization and facilitating efficient traversal and search operations. The total number of nodes in a full r-ary tree of height h is given by the formula $\frac{r^{(h+1)}-1}{r-1}$, where r is the branching factor. These trees generalize the concept of binary trees, offering a broader perspective on

hierarchical data structures and their algorithmic implications in diverse computing domains [39]. This work considers two variants of the full r-ary tree: $r = 2$ and $r = 4$.

Erdö–Rényi random network. The concept of random graphs introduced by Paul Erdös and Alfréd Rényi, who demonstrated the effectiveness of probabilistic methods in solving graph theory problems [10, 11]. Initially introduced in the 1950s, when computing power was limited, much of the modeling focused on relatively small "ordered" or "regular" networks, which are infrequent in real-world scenarios [4]. An alternative definition of a random graph is the binomial model, where the $G(N, p)$ model begins with N nodes and connects each distinct node pair with a probability p. This work considers two variants of the Erdös-Rényi model based on edge probability p: $p = 0.2$ and $p = 0.5$. As expected, the structure of the graph generated by each probability varies significantly; as p increases, the number of edges also increases.

The uncorrelated Erdö–Rényi random graph model assumes equal and independent probabilities for every pair of vertices, treating the network as a collection of equivalent units. However, real networks are inherently correlated systems, and their topology often deviates from the uncorrelated random graph model. The focus has shifted to developing more sophisticated graph models, with an emphasis on "real-world" networks such as the Internet and the World-Wide Web. To understand the general properties of such networks, two prominent classes of models have emerged: "small-world" and "scale-free". Small-world networks aim to capture the clustering observed in real graphs and are inhomogeneous, with relatively localized patterns of connectivity between nodes. Scale-free networks exhibit inhomogeneity in the "degree" of nodes (i.e., the number of connections a node has to other nodes) and replicate the power-law degree distribution present in many real networks.

Barabási-Albert scale-free network. Barabási and Albert [7] introduced the concept of scale-free networks, capable of reproducing networks with "hubs", where a few nodes have considerably more connections than the average, a property known as scale-free. Since numerous real-world networks exhibit degree distributions similar to the Barabási-Albert model, it continues to be one of the most renowned and frequently employed network generation methods. The algorithm that produces a Barabási-Albert scale-free network of size N, starts with a small number of nodes m_o and then $N - m_o$ nodes are introduced sequentially into the network, where each node connects to/from $m \leq m_o$ existing nodes. Therefore, the initial size of the network m_o determines the maximum mean degree of the network. This work considers two variants of the Barabási-Albert model: $m = 1$ and $m = 3$. As expected, the graph structure generated by each m value varies significantly; as m increases, the number of edges also increases.

Watts–Strogatz small-world network. Watts and Strogatz [41] proposed what has become known as the archetypical small-world network. The algorithm begins by constructing an undirected ring lattice network, consisting of a ring of nodes with edges evenly distributed between its k_L nearest left and right neighbors. The value k_L denotes the degree of each node in the initial lattice. Subsequently, a random rewiring process is applied, where each edge has a probability p of being rewired. The algorithm only rewires one end of each edge and traverses edges in a way that ensures that each node loses at most half of its edges. It is important to note that edges are only replaced, not added or removed, thus the total number of edges and the mean degree is unchanged. By varying the rewiring probability p, it can be demonstrated that only a small number of rewires is required to produce a low average path length while preserving a high clustering coefficient. Specifically, for $p = 0$, the small-world model retains a regular graph, while for $p = 1$, a random graph is generated, differing only slightly from the uncorrelated random graph. For intermediate values of p, the Watts-Strogatz model produces a small-world network, which captures the high clustering properties of regular graphs and the small characteristic path length of random graph models. Consequently, our focus was exclusively on a single rewiring probability of $p = 0.5$, exploring two distinct degree values: $k_L = 2$ and $k_L = 4$. As the degree of a node represents the number of edges connecting it to other nodes in the graph, the resulting graph structure exhibits significant variations for each degree value; as k_L increases, the number of edges also increases.

4.2 Algorithmic Approaches

4.2.1 Greedy. The Greedy algorithm is a powerful method for effectively addressing various optimization problems. It works by always selecting the option that appears best at the moment, attempting to maximize the return based on local conditions, assuming that this will lead to a globally optimal solution [8]. When applied to the minimum weighted vertex cover problem, the Greedy algorithm begins with an empty set of vertices, denoted as S. The algorithm dynamically updates the weights during each iteration, with the weight of a vertex being the sum of the weights of the edges it covers. This weight update mechanism ensures that the algorithm considers the cumulative impact of edge weights when selecting vertices for inclusion in the vertex cover. It then proceeds by selecting an arbitrary edge, e, from the graph and adds both endpoints of e to set S. Subsequently, the algorithm removes all edges covered by the vertices in S from the graph. If the graph becomes empty, set S is returned as the solution to the minimum weighted vertex cover. However, if the graph is not empty, the process iterates by selecting another arbitrary edge, e. At each iteration, the algorithm prioritizes the edge with the fewest number of uncovered vertices, adding both endpoints to the vertex cover set. This strategy ensures the set of vertices covers as many edges as possible while considering the weights assigned to each vertex. While the Greedy algorithm may not always yield an optimal solution, encountering local optima in certain scenarios, it demonstrates favorable performance for the minimum weighted vertex cover problem. Specifically, the Greedy algorithm is known to produce a solution size at most twice that of the optimal solution, establishing a 2-approximation guarantee.

4.2.2 Genetic. Genetic Algorithms (GAs) leverage evolutionary principles such as natural selection, genetic variation, and the survival of the fittest observed in biological organisms to create heuristic search algorithms [14]. Inspired by Darwinian evolution, GAs employ concepts like crossover, mutation, and natural selection to intelligently explore a defined search space and tackle complex problems [18]. In cases characterized by vast exploration spaces and the availability of fitness evaluations for solutions, GAs prove highly effective. Hence, GAs can be applied to solve the MWVC problem.

The evolutionary process initiates with the creation of a population of chromosomes encoding potential solutions to the MWVC problem. These chromosomes are generated either randomly or using heuristic methods. Parent chromosomes are then chosen based on their fitness values, which reflect how effectively their corresponding sets of vertices cover edges in the graph. Through stochastic selection and modification employing crossover and mutation operators, new populations emerge, inheriting advantageous traits from the fittest individuals of the preceding generation. This iterative cycle persists until an optimal solution is attained or a predefined termination condition is met. The optimization mechanism of a genetic algorithm relies on pivotal elements such as the fitness function, encoding scheme, crossover, and mutation. Together, these components drive the generation of increasingly fit solutions. The output of the GA is the chromosome with the highest fitness value, corresponding to a set of vertices constituting a minimum weighted vertex cover.

The genetic algorithm was configured with the following parameters: a) Population Size, 100 individuals per generation, b) Generations, 150, c) Selection Method, Roulette wheel selection was utilized to choose individuals from the current population as parents for generating offspring, d) Crossover Method, Order Crossover (OX) was applied to combine genetic material from selected parents and e) Mutation Rate, A mutation rate of 0.1 was set, indicating a 10% chance of mutation for each gene in an individual solution during each generation of the genetic algorithm.

4.3 Simulation Results

This section presents the simulation results for the different network topologies obtained by the two algorithms examined. Two performance metrics are employed to evaluate the efficacy of each algorithm in handling the proactive application image placement problem. The metrics considered include: i) execution time (ExT): total duration each algorithm takes to produce a solution and ii) vertex cover set (VCS): the size of vertices in it.

As discussed in subsection 4.1, different variants are considered for each network topology, taking into account the input parameters associated with each model. A single representative variation is visually depicted for each network topology: $r = 2$ for full r-ary tree, $p = 0.2$ for Erdö–Rényi, $m = 1$ for Barabási-Albert and $k_L = 2$ for Watts–Strogatz. Detailed results for the remaining variants of each model are provided in Table 1. In the table, the abbreviations FR, ER, BA and WS correspond to Full r-ary tree, Erdö–Rényi, Barabási-Albert and Watts–Strogatz, respectively.

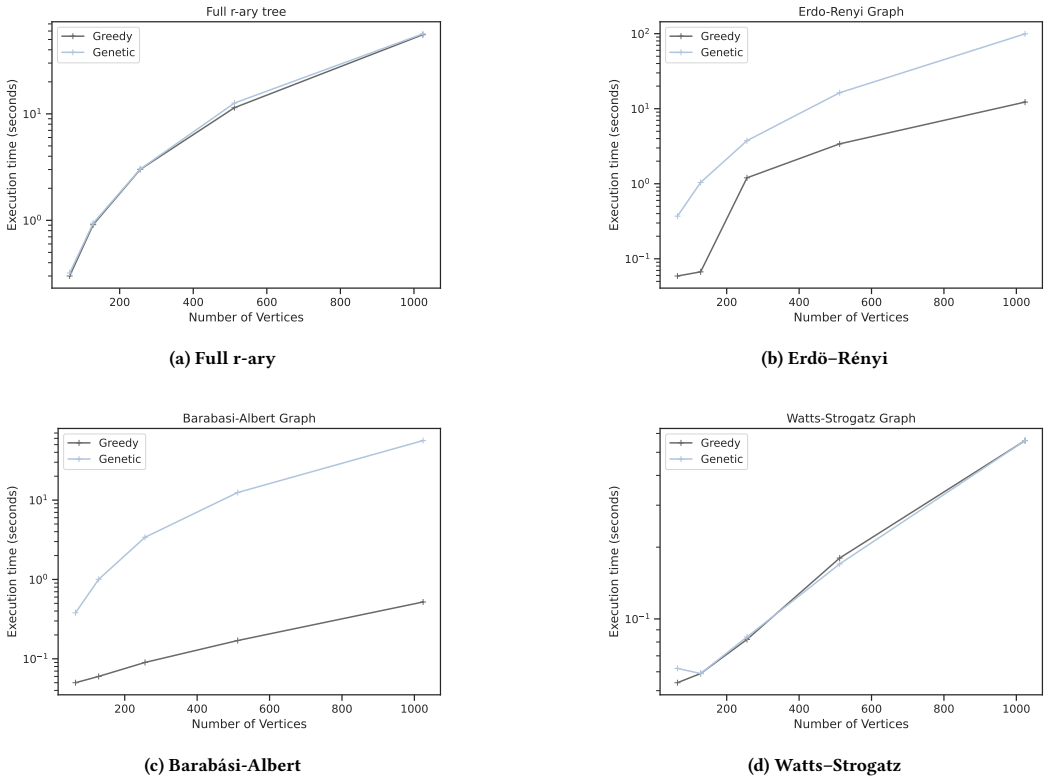
Execution time analysis. Figure 1 evaluates the execution time of each algorithm for the different network topologies. As the results indicate, the execution time of both examined algorithms is significantly higher for Erdö–Rényi graphs (Figure 1b) compared to other network topologies, especially as the number of vertices increases. As an illustration, when considering 1024 vertices, the Genetic algorithm demonstrates an execution time of 99 seconds for Erdö–Rényi graphs and 56 seconds (almost 2 times slower) for Watts-Strogatz graphs, whereas for the Greedy algorithm, the execution times are 2.2 seconds and 0.5 seconds (almost 5 times slower), respectively. Both algorithms exhibit the lowest execution times for Barabási-Albert graphs, however, the execution times are comparable with Watts-Strogatz and full r-ary tree. As the results suggest, the Genetic algorithm demonstrates higher execution times compared to the Greedy algorithm for all network topologies. Both algorithms demonstrate a consistent pattern across the various network topologies, with the execution time showing a linear growth in tandem with the increase in the number of vertices. Generally, as the number of vertices increases and additional nodes are incorporated into the vertex cover set, the execution time escalates, as the generated graph becomes larger.

Vertex Cover Set size analysis. Figure 2 evaluates the vertex cover set produced by each algorithm for the different network topologies. As expected, the vertex cover set's size increases in a linear fashion with an increase in the number of vertices. As the results suggest, compared to other network topologies, Greedy algorithm generates a significantly smaller vertex cover set for Barabási-Albert graphs. As an illustration, when considering Barabási-Albert graphs with 512 vertices, the Greedy algorithm generates a vertex cover set size of 176 while for Erdö–Rényi, Watts-Strogatz and full r-ary tree the sizes are 502, 328 and 256 respectively. As the results indicate, the Greedy algorithm produce the largest vertex cover sets for Erdö–Rényi graphs followed by Watts-Strogatz. The Genetic algorithm produces vertex cover sets of nearly equivalent sizes across the different network topologies, demonstrating minimal fluctuations. The results demonstrate that the Greedy algorithm yields smaller vertex cover sets compared to the Genetic algorithm across all network topologies.

Discussion. A brief overview of the experimental simulation results and key findings is provided to summarize the performance assessment of each algorithm. As the results suggested, the Genetic algorithm demonstrates higher execution times compared to the Greedy algorithm for all network topologies. This is due to the fact that applications of Genetic algorithms in the optimizations of complex problems can lead to a substantial computational effort, given the iterative evaluation of the objective function(s) and the population-based nature of the search process. As the size of the problem and the population increases, the time required to generate and evaluate each generation of candidate solutions also increases, resulting in longer execution time. Regarding different graph topologies, the Greedy algorithm produces the smallest vertex cover set sizes for Barabási-Albert and full r-ary tree, while the largest sizes are produced for Erdö–Rényi followed by Watts–Strogatz graphs. In general, in terms of the size of the vertex cover set, the Genetic algorithm generates larger results than the Greedy algorithm for all network topologies. To summarize, the Greedy approach presents superior results with respect to the number of allocated images (vertex cover set size) and execution time.

5 CONCLUSION

The effective placement of Edge resources plays a pivotal role in managing Edge Computing platforms, facilitating the deployment of applications near the end-users. This placement is usually driven by scheduling decisions aimed at optimizing various performance indicators, including network performance, and ensuring data proximity to its source. Implementing these scheduling decisions often involves transferring and deploying potentially large application



(a) Full r-ary **(b) Erdö–Rényi**

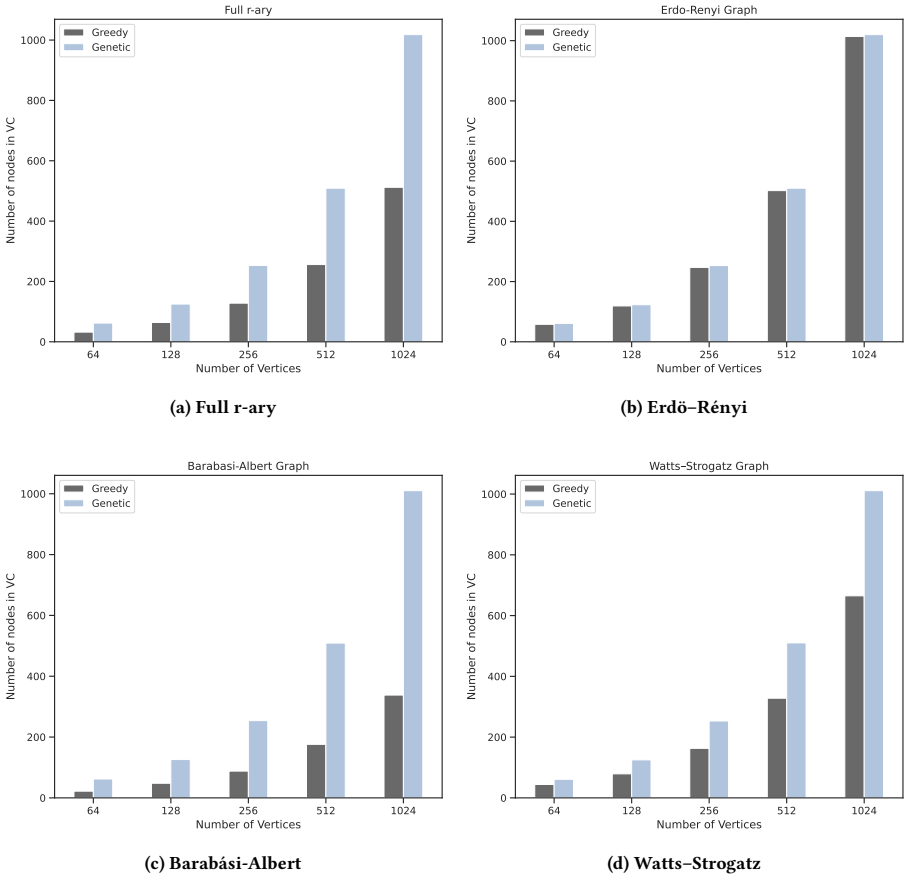
(c) Barabási-Albert **(d) Watts–Strogatz**

Figure 1: Execution time of each algorithm for the different network topologies

Table 1: Detailed results for the remaining variant of each model

V	Metrics		r=4			p=0.5			m=3			k=4	
			Greedy	Genetic		Greedy	Genetic		Greedy	Genetic		Greedy	Genetic
64	ExT		0.28	0.32		0.34	0.4		0.058	0.078		0.059	0.37
	VCS		16	62		58	62		10	62		33	61
128	ExT		0.92	0.98		0.98	1.17		0.069	1.01		0.062	0.19
	VCS		32	126		116	127		21	125		71	125
256	ExT		3.01	3.03		4.1	4.5		0.12	3.36		0.10	1.18
	VCS	FR	64	254	ER	504	510	BA	52	253	WS	138	252
512	ExT		11.32	12.65		22.34	26.08		0.31	12.57		0.26	1.89
	VCS		128	508		501	510		100	507		280	509
1024	ExT		46.43	56.43		138.21	165.32		1.16	56.23		0.93	2.34
	VCS		256	1011		1018	1022		206	1022		570	1021

images across numerous Edge resources. Hence, it is crucial to provide strategies that proactively transfer these images in a manner that minimizes the download time upon request while simultaneously reducing the overall number of transfers. In this paper, we



(a) Full r-ary **(b) Erdö–Rényi**

(c) Barabási-Albert **(d) Watts–Strogatz**

Figure 2: Vertex Cover set of each algorithm for the different network topologies

model the problem of proactive placement of application images as a Minimum Weighted Vertex Cover problem. Two placement strategies, Greedy and Genetic, are compared against different network topologies, simulating various possible Edge computing environments. The results indicate that the Greedy approach offers the optimal tradeoff with respect to the number of allocated images and execution time. As a future work, we intend to investigate other methodologies for solving the MWVC problem and broaden the scope of algorithmic approaches under consideration. Additionally, we will expand the formulation assessment beyond conventional factors such as latency and storage capacity, considering a multi-dimensional spectrum, taking into account variables such as the object's volume, available link bandwidth, transfer time constraints and placement cost.

ACKNOWLEDGMENTS

This project has received funding from the EU's Horizon 2020 program under Grant agreement No 101016509 (CHARITY). This paper reflects only the authors' view and the Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] Farah Aït-Salaht, Frédéric Desprez, and Adrien Lebre. 2021. An Overview of Service Placement Problem in Fog and Edge Computing. *ACM Comput. Surv.* 53, 3 (2021), 65:1–65:35. https://doi.org/10.1145/3391196

[2] Farah Aït-Salaht, Frédéric Desprez, Adrien Lebre, Charles Prud'homme, and Mohamed Abderrahim. 2019. Service Placement in Fog Computing Using Constraint Programming. In *2019 IEEE International Conference on Services Computing, SCC 2019, Milan, Italy, July 8-13, 2019*, Elisa Bertino, Carl K. Chang, Peter Chen, Ernesto Damiani, Michael Goul, and Katsunori Oyama (Eds.). IEEE, 19–27. https://doi.org/10.1109/SCC.2019.00017

[3] Muhammad Alam, Joao Rufino, Joaquim Ferreira, Syed Hassan Ahmed, Nadir Shah, and Yuanfang Chen. 2018. Orchestration of microservices for iot using docker and edge computing. *IEEE Communications Magazine* 56, 9 (2018), 118–123.

[4] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Reviews of modern physics* 74, 1 (2002), 47.

[5] Hamid Reza Arkian, Abolfazl Diyanat, and Atefe Pourkhalili. 2017. MIST: Fog-based data analytics scheme with cost-efficient resource provisioning for IoT crowdsensing applications. *J. Netw. Comput. Appl.* 82 (2017), 152–165. https://doi.org/10.1016/J.JNCA.2017.01.012

[6] Onur Ascigil, Truong Khoa Phan, Argyrios G. Tasiopoulos, Vasilis Sourlas, Ioannis Psaras, and George Pavlou. 2017. On Uncoordinated Service Placement in Edge-Clouds. In *IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2017, Hong Kong, December 11-14, 2017*. IEEE Computer Society, 41–48. https://doi.org/10.1109/CLOUDCOM.2017.46

[7] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *science* 286, 5439 (1999), 509–512.

[8] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2009. *Introduction to algorithms*. MIT press.

[9] Jad Darrous, Thomas Lambert, and Shadi Ibrahim. 2019. On the Importance of Container Image Placement for Service Provisioning in the Edge. In *28th International Conference on Computer Communication and Networks, ICCCN 2019, Valencia, Spain, July 29 - August 1, 2019*. IEEE, 1–9. https://doi.org/10.1109/ICCCN.2019.8846920

[10] Paul Erdös and Alfréd Rényi. 1959. On random graphs Publ. *Math. debrecen* 6 (1959), 290–297.

[11] Edgar N Gilbert. 1959. Random graphs. *The Annals of Mathematical Statistics* 30, 4 (1959), 1141–1144.

[12] Diogo Gonçalves, Karima Velasquez, Marília Curado, Luiz Fernando Bittencourt, and Edmundo Roberto Mauro Madeira. 2018. Proactive Virtual Machine Migration in Fog Environments. In *2018 IEEE Symposium on Computers and Communications, ISCC 2018, Natal, Brazil, June 25-28, 2018*. IEEE, 742–745. https://doi.org/10.1109/ISCC.2018.8538655

[13] Aric Hagberg, Pieter Swart, and Daniel S Chult. 2008. *Exploring network structure, dynamics, and function using NetworkX*. Technical Report. Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

[14] John H Holland. 1992. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

[15] Hua-Jun Hong, Pei-Hsuan Tsai, and Cheng-Hsin Hsu. 2016. Dynamic module deployment in a fog computing platform. In *18th Asia-Pacific Network Operations and Management Symposium, APNOMS 2016, Kanazawa, Japan, October 5-7, 2016*. IEEE, 1–6. https://doi.org/10.1109/APNOMS.2016.7737202

[16] Kiranpreet Kaur, Fabrice Guillemin, and Françoise Sailhan. 2022. Container placement and migration strategies for cloud, fog, and edge data centers: A survey. *Int. J. Netw. Manag.* 32, 6 (2022). https://doi.org/10.1002/NEM.2212

[17] Ioannis Korontanis, Konstantinos Tserpes, Maria Pateraki, Lorenzo Blasi, John Violos, Ferran Diego, Eduard Marin, Nicolas Kourtellis, Massimo Coppola, Emanuele Carlini, et al. 2020. Inter-operability and orchestration in heterogeneous cloud/edge resources: The ACCORDION vision. In *Proceedings of the 1st Workshop on Flexible Resource and Application Management on the Edge*. 9–14.

[18] Ketan Kotecha and Nilesh Gambhava. 2003. A Hybrid Genetic Algorithm for Minimum Vertex Cover Problem.. In *IICAI*. 904–913.

[19] Gilsoo Lee, Walid Saad, and Mehdi Bennis. 2017. An online secretary framework for fog network formation with minimal latency. In *IEEE International Conference on Communications, ICC 2017, Paris, France, May 21-25, 2017*. IEEE, 1–6. https://doi.org/10.1109/ICC.2017.7996574

[20] Antonios Makris, Abderrahmane Boudi, Massimo Coppola, Luís Cordeiro, Massimiliano Corsini, Patrizio Dazzi, Ferran Diego Andilla, Yago González Rozas, Manos Kamarianakis, Maria Pateraki, et al. 2021. Cloud for holography and augmented reality. In *2021 IEEE 10th International Conference on Cloud Networking (CloudNet)*. IEEE, 118–126.

[21] Antonios Makris, Ioannis Kontopoulos, Evangelos Psomakelis, Stylianos Nektarios Xyalis, Theodoros Theodoropoulos, and Konstantinos Tserpes. 2022. Performance Analysis of Storage Systems in Edge Computing Infrastructures. *Applied Sciences* 12, 17 (2022), 8923.

[22] Antonios Makris, Evangelos Psomakelis, Ioannis Korontanis, Theodoros Theodoropoulos, Antonis Protopsaltis, Maria Pateraki, Zbyszek Ledwoń, Christos Diou, Dimosthenis Anagnostopoulos, and Konstantinos Tserpes. 2023. Streamlining XR Application Deployment with a Localized Docker Registry at the Edge. In *European Conference on Service-Oriented and Cloud Computing*. Springer, 188–202.

[23] Antonios Makris, Evangelos Psomakelis, Theodoros Theodoropoulos, and Konstantinos Tserpes. 2022. Towards a Distributed Storage Framework for Edge Computing Infrastructures. In *Proceedings of the 2nd Workshop on Flexible Resource and Application Management on the Edge*. 9–14.

[24] Antonios Makris, Konstantinos Tserpes, and Theodora Varvarigou. 2022. Transition from monolithic to microservice-based applications. Challenges from the developer perspective. *Open Research Europe* 2 (2022), 24.

[25] Mohammed Islam Naas, Philippe Raipin Parvédy, Jalil Boukhobza, and Laurent Lemarchand. 2017. iFogStor: An IoT Data Placement Strategy for Fog Infrastructure. In *1st IEEE International Conference on Fog and Edge Computing, ICFEC 2017, Madrid, Spain, May 14-15, 2017*. IEEE Computer Society, 97–104. https://doi.org/10.1109/ICFEC.2017.15

[26] Michael J. Neely. 2010. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool Publishers. https://doi.org/10.2200/S00271ED1V01Y201006CNT007

[27] Charith Perera, Yongrui Qin, Júlio Cezar Estrella, Stephan Reiff-Marganiec, and Athanasios V. Vasilakos. 2017. Fog Computing for Sustainable Smart Cities: A Survey. *ACM Comput. Surv.* 50, 3 (2017), 32:1–32:43. https://doi.org/10.1145/3057266

[28] Evangelos Psomakelis, Antonios Makris, Konstantinos Tserpes, and Maria Pateraki. 2023. A lightweight storage framework for edge computing infrastructures/EdgePersist. *Software Impacts* 17 (2023), 100549.

[29] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley. https://doi.org/10.1002/9780470316887

[30] Kaustabha Ray, Ansuman Banerjee, and Nanjangud C. Narendra. 2020. Proactive Microservice Placement and Migration for Mobile Edge Computing. In *5th IEEE/ACM Symposium on Edge Computing, SEC 2020, San Jose, CA, USA, November 12-14, 2020*. IEEE, 28–41. https://doi.org/10.1109/SEC50012.2020.00010

[31] Fabiana Rossi, Valeria Cardellini, Francesco Lo Presti, and Matteo Nardelli. 2020. Geo-distributed efficient deployment of containers with Kubernetes. *Comput. Commun.* 159 (2020), 161–174. https://doi.org/10.1016/J.COMCOM.2020.04.061

[32] Olena Skarlat, Matteo Nardelli, Stefan Schulte, Michael Borkowski, and Philipp Leitner. 2017. Optimized IoT service placement in the fog. *Serv. Oriented Comput. Appl.* 11, 4 (2017), 427–443. https://doi.org/10.1007/S11761-017-0219-8

[33] Olena Skarlat, Matteo Nardelli, Stefan Schulte, and Schahram Dustdar. 2017. Towards QoS-Aware Fog Service Placement. In *1st IEEE International Conference on Fog and Edge Computing, ICFEC 2017, Madrid, Spain, May 14-15, 2017*. IEEE Computer Society, 89–96. https://doi.org/10.1109/ICFEC.2017.12

[34] Mohit Taneja and Alan Davy. 2017. Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, May 8-12, 2017*. IEEE, 1222–1228. https://doi.org/10.23919/INM.2017.7987464

[35] Theodoros Theodoropoulos, Dimitrios Kafetzis, John Violos, Antonios Makris, and Konstantinos Tserpes. 2023. Multi-Agent Deep Reinforcement Learning for Weighted Multi-Path Routing. In *Proceedings of the 3rd Workshop on Flexible Resource and Application Management on the Edge*. 7–11.

[36] Theodoros Theodoropoulos, Antonios Makris, Abderrahmane Boudi, Tarik Taleb, Uwe Herzog, Luis Rosa, Luis Cordeiro, Konstantinos Tserpes, Elena Spatafora, Alessandro Romussi, et al. 2022. Cloud-based xr services: A survey on relevant challenges and enabling technologies. *Journal of Networking and Network Applications* 2, 1 (2022), 1–22.

[37] Theodoros Theodoropoulos, Antonios Makris, Ioannis Kontopoulos, Angelos-Christos Maroudis, and Konstantinos Tserpes. 2023. Multi-Service Demand Forecasting Using Graph Neural Networks. In *2023 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 218–226.

[38] Theodoros Theodoropoulos, Antonios Makris, Evangelos Psomakelis, Emanuele Carlini, Matteo Mordacchini, Patrizio Dazzi, and Konstantinos Tserpes. 2023. GNOSIS: Proactive Image Placement Using Graph Neural Networks & Deep Reinforcement Learning. In *2023 IEEE 16th International Conference on Cloud Computing (CLOUD)*. IEEE, 120–128.

[39] Wanqing Tu, Xing Jin, and Peter S Excell. 2009. Performance Analysis for Overlay Multimedia Multicast on r-ary Tree and m-D Mesh Topologies. *IEEE transactions on multimedia* 11, 4 (2009), 696–706.

[40] Karima Velasquez, David Perez Abreu, Marília Curado, and Edmundo Monteiro. 2017. Service placement for latency reduction in the internet of things. *Ann. des Télécommunications* 72, 1-2 (2017), 105–115. https://doi.org/10.1007/S12243-016-0524-9

[41] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world'networks. *nature* 393, 6684 (1998), 440–442.

[42] Ashkan Yousefpour, Ashish Patil, Genya Ishigaki, Inwoong Kim, Xi Wang, Hakki C. Cankaya, Qiong Zhang, Weisheng Xie, and Jason P. Jue. 2019. FOGPLAN: A Lightweight QoS-Aware Dynamic Fog Service Provisioning Framework. *IEEE Internet Things J.* 6, 3 (2019), 5080–5096. https://doi.org/10.1109/JIOT.2019.2896311

[43] Ruozhou Yu, Guoliang Xue, and Xiang Zhang. 2018. Application Provisioning in FOG Computing-enabled Internet-of-Things: A Network Perspective. In *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018*. IEEE, 783–791. https://doi.org/10.1109/INFOCOM.2018.8486269

[44] Aleksandr Zavodovski, Nitinder Mohan, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. 2018. ICON: Intelligent Container Overlays. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks, HotNets 2018, Redmond, WA, USA, November 15-16, 2018*. ACM, 15–21. https://doi.org/10.1145/3286062.3286065

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009