



VALKYRIE: a suite of topology-aware clustering approaches for cloud-based virtual network services

Imane El Mansoum¹ · Laaziz Lahlou¹  · Fawaz A. Khasawneh¹ · Nadjia Kara¹ · Claes Edstrom²

Accepted: 21 August 2022 / Published online: 2 September 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

Complex networks are practical tools for modeling, studying, and analyzing complex interactions between objects. These tools are essential in understanding applications, end-users, interactions between compute nodes, and their behaviors in computer networks. Computer networks are undergoing significant expansion due to the proliferation of network devices and compute nodes. One of the main challenges in computer networks is categorizing these compute nodes into clusters of connected compute nodes within these large-scale structures sharing similar features (e.g., Central Processing Unit, memory, disk storage). This paper proposes a set of novel, dynamic, and proactive topology-aware clustering approaches, namely, an Integer Linear Program, chemical reaction optimization, and a game theory approach that leverages the Irving algorithm, originally proposed to solve the stable roommate problem, to form clusters based on the compute nodes' features and their topological structures. Our proposed techniques are suggested to reduce the search space concerning Network Function Virtualization, Cloud-based Networks deployment, and to build on-demand clusters to meet their requirements. In this regard, the solutions aim to help decision-makers facing issues related to scalability and computational complexities of their mechanisms to deploy their cloud-based services effectively. Experimental results demonstrate the proposed approaches' effectiveness and suitability, given their polynomial-time complexities, making them easy to integrate into cloud providers' orchestration systems compared to *K*-Means and Density-based spatial clustering of applications with noise.

Keywords Network function virtualization · Attributed network infrastructures · Clustering · Multi-objective optimization · Topological structure · Attribute similarity

Extended author information available on the last page of the article

1 Introduction

Complex networks, such as computer communication networks, brain networks, and social networks, are formal and practical methods for modeling, exploring, and evaluating complex interactions between objects having non-trivial properties in various fields [1]. These technologies are crucial for comprehending applications, end users, and the interactions and behavior of compute nodes in computer networks. These technologies fundamentally operate within the framework of graph theory, where nodes stand in for objects and edges for node interactions. Commodity servers, computing nodes, and network devices (legacy/virtualized) are represented as nodes in computer networks. In contrast, edges represent their interactions, which, depending on the environment in which they are created, might take many different forms (such as dependencies, bandwidth capacity, or delay).

To cluster attributed graphs, several methods have been put onwards [1]. There are two types of these techniques: parameter-free methods and parameter-dependent techniques. In contrast to the parameter-free strategy, which requires no such input, the user in a parameter-dependent approach supplies the clustering algorithm with the desired number of clusters. According to [2–5], clustering is a valuable unsupervised learning strategy. It seeks to maintain distinct entities in separate clusters and combine comparable ones into one cluster. Clustering has several uses, including financial and time series analysis, community discovery in social networks, and the study of geographical and astronomical data. Many currently used clustering techniques merely consider the nodes' characteristics or topological structure in addition to the categorization. The type of the current problem and the intended outcome is the key factors determining the technique to take. The nodes or the connections between them determine this decision (i.e., focused on the structural part).

A similarity function is computed by taking only node attributes or structural attributes into account when forming clusters. Clusters are built based on this similarity function because members of clusters are only grouped when they are similar. We consider clustering quality as part of our formulation and modeling of clustering and other evaluation cost functions. Clustering quality is evaluated once clusters are built in most existing methods for clustering attributed graphs [1–3], and [4]. Instead of evaluating quality during clustering, the traditional approach involves evaluating it afterward. In our approaches, clusters are formed by continuously improving them when considering the node properties and their structural aspects since the infrastructure is heterogeneous and complex. These aspects can impact the clustering task, and these clusters could negatively affect the resources available to the network or service provider when deploying and provisioning applications and services.

Attributed graphs model real networks by augmenting their nodes with attributes. For example, these attributes pertain to CPU, Memory, and Disk Storage in networking. Thus, our clustering approach of an attributed graph is devoted to the service function chaining problem and broadly to cloud-based virtual network services where the goals are:

- (1) Narrowing the search space from an algorithmic viewpoint to assist in finding quicker solutions for the algorithms created to deploy them.
- (2) Create on-demand clusters, such as CPU-intensive and bandwidth-efficient clusters, according to their demands.

As illustrated in Fig. 1, the proposed clustering approaches align with the Virtual Network Function Manager (VNFM) following the NFV MANO architecture [6], where three different approaches are proposed and compared together, namely, ILP, Game Theory (GT), and Chemical Reaction Optimization (CRO) [7]. Another flavor for each game theory and ILP approach is introduced. Our proposed approaches interact with the infrastructure to get all the required information, such as the network topology, CPU, and memory utilization, to perform the clustering dynamically. Proposing three different approaches will allow the decision-makers to choose one based on their cluster requirements. They may invoke a CRO-based clustering procedure, a GT-based procedure, or even the ILP solution to build the desired clusters according to the underlying network topology (small-scale, medium-scale, and large-scale networks).

This paper fills existing gaps in the research literature with the following key contributions : A formulation is provided for the pooled server and network attributes for the dynamic and proactive clustering problem tailored for service function chains and, broadly, for virtual network services in the context of NFV. The problem is formulated as a Quadratic Constrained Integer Linear Program, implemented and solved in line with Gurobi [8], to find optimal solutions for small-scale networks. Then, a fast and scalable Chemical Reaction Optimization approach is designed to handle medium and large-scale instances of the problem, leveraging the same ILP structure (cost functions and constraints) for reliable benchmarking. Similarly, the

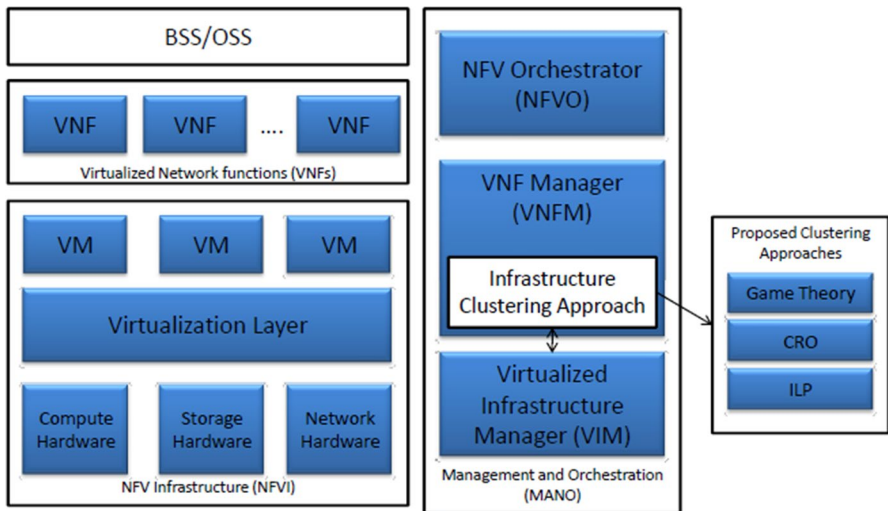


Fig. 1 VALKYRIE’s alignment with respect to MANO architecture

clustering problem is formulated as a matching game and solved using an adapted version of Irving's algorithm [9].

Unless otherwise specified, the word *valkyrie* derives from Old Norse *Valkyrja* (plural *valkyrjur*), which is composed of two words: the noun *valr* (referring to the slain on the battlefield) and the verb *kjósa* (meaning “to choose”). Informally, the word *slain* means successful. Then, *valkyrie* means choosing successfully while clustering.

The rest of this paper is organized as follows. Section 2 presents the related works, while Sect. 3 presents the system model and states the clustering problem. Our clustering techniques based on Chemical Reaction Optimization and Game Theory are presented in Sects. 5 and 6, respectively. Section 6 discusses their asymptotic analysis. The experimental evaluation is presented in Sect. 7, followed by a discussion of the results in Sect. 8. Finally, Sect. 9 concludes this paper.

2 Related works

This section reviews the most relevant techniques proposed in the context of the networks for service function chaining.

Although very few articles have proposed the formation of virtual clusters from physical servers, the literature on cloud environments, primarily virtual machine clustering, is presented in detail to optimize resource consumption and physical infrastructure management. In addition, we present the techniques for community detection in social networks, an application of clustering with attributed graphs that are similar to our considered context.

2.1 Clustering in cloud environments

A topology-aware approach to hosting virtual machine clusters on a compute pool consisting of a set of physical machines is proposed in [10]. The virtual machine cluster is a set of virtual machines that host a distributed application or service. However, to avoid high bandwidth consumption and low performance of the offered service, the challenge is to locate these VMs on physical machines so that two VMs belonging to the same cluster are not too distant apart. Authors represent the VM cluster as a directed graph where the nodes are the VMs, and the links are the interconnections. To address this placement problem, the authors suggest a greedy algorithm that first attempts to host the entire VM cluster on the same physical server if it has enough capacity. If not, it begins with the node with the minimum connectivity weight and then places that VM on the physical machine with the least available capacity to host the VM. At the end of the placement, the algorithm omits this VM from the original graph and proceeds the same way to host the remaining VMs. They compare their approach to their earlier work in [11], in which connectivity between VMs was not accounted for. Further, a comparison is made between this approach and two basic approaches: the first-come-first-served and round-robin approaches [12, 13], where both are applied to VM scheduling. Findings indicate

that this topology-aware technique enhances bandwidth consumption compared to the performance of the other three methods.

Authors in [14] put forward an approach to cluster virtual machines to enable their scheduling over a shared pool of physical machines and decrease the complexity of their placement, reconfiguration, or migration. Their proposed solution strives to guarantee fair resource sharing among VMs, which results in increased resource availability provided to end-users. The clustering technique utilizes the similarity between VMs in terms of attributes such as the memory, operating system (OS), or hardware configuration. In this work, *K*-Means was used to perform the clustering. The authors also devise a mathematical model based on linear programming to improve the utilization of the clustered VMs and boost the performance.

A method to build virtual clusters made up of virtual machines devoted to high-performance computing tasks is suggested by [15]. Since virtual clusters are often hosted on the same site to prevent performance deterioration and maintain the Quality of Service (QoS) delivered at a high level, the authors present multi-site clustering. To enable better administration, resource availability, and resource sharing among several end users, the suggested strategy focuses more on the communication between the VMs over remote locations. Assuring that VMs may communicate with one another even if they are not housed on the same physical pool of servers, an overlay network is used to divide the networks of each virtual cluster.

Although this method may provide clients access to more resources and computational and processing capabilities, it substantially impacts connection. The same application would incur longer delays and more bandwidth usage if split between two VMs hosted in different locations.

In IaaS platforms (Infrastructure as a Service), where it is crucial to secure the isolation between the virtual machines serving various customers, authors in [16] address the security issue. The authors use a modified sequential *K*-Means algorithm-based technique to identify unusual behavior and anomalies in resource consumption patterns across specific virtual machines. If the algorithm notices unusual resource usage peaks, it might indicate that a malevolent outsider is trying to assault the system. A clustering operation is carried out on sets of virtual machines depending on specific different parameters (such as CPU, Memory, and Disk Storage). The architecture and features of the application are taken into account by the *K*-Means technique. The resulting clusters (such as web, database, and applications) are organized into groups based on the type of traffic received. When a virtual machine's resource usage deviates significantly from the cluster centroid, it is classified as malicious. Based on a threshold, the cloud provider has established, this labeling choice. Using an OpenStack system, the method was tested. It was shown to be effective in spotting rogue virtual machines and improving the security and monitoring of dispersed cloud environments and IaaS systems.

The authors in [17] suggest a cluster-based technique for placing virtual network functions. The clustering phase, regarded as a pre-processing step, divides the substrate network into coherent groups regarding particular metrics like CPU, Memory, and Disk Storage. The results showed that the authors' adoption of a *K*-Medoids-based strategy helped shorten the overall placement period and any necessary migration. It is important to note that the suggested approach considers the servers'

characteristics. However, in real-world conditions, it is also required to consider the delays across the connections in the substrate network and their available bandwidth. They also assume that the properties of the nodes only determine the clustering. When this happens, it is probable for two servers in the same cluster to have delays that are too significant, which could raise the ratio of SLA/SLO violations in terms of the desired delay and significantly lower service performance.

2.2 Community detection in social networks

Finding groups of individuals with connections with one another and traits that indicate they are friends on a social network is known as community detection. In our research, these entities are servers with shared attributes.

The issue of community detection is investigated in data analysis and processing by [18]. Their study aims to enhance clustering quality beyond conventional approaches, hampered by high complexity and extensive execution time. To create the communities, the authors suggest using the genetic algorithm heuristic. The objective function of this study solely takes modularity into account. Although this metric does not take the node's properties into account, it is demonstrated through simulation that the suggested technique can produce acceptable results compared to the ground truth found in the data set employed. The algorithm's run-time is not specified, and the suggested method was not compared to the traditional clustering algorithms outlined in the literature.

In the context of social networks, [19] examined several evolutionary algorithms for community formation. Particle swarm optimization, cat swarm optimization, and the genetic algorithm coupled with simulated annealing are the three basic heuristics. The authors considered modularity as an objective function in this context. A weighted graph with nodes representing individuals and edges for connections between them serves as the representation of the social network. The suggested heuristics comparison is examined on traditional data sets, such as Zachary's karate club data set [20]. The findings demonstrate that each approach's modularity and the number of communities discovered vary. However, the strategy using a genetic algorithm combined with simulated annealing offers the best trade-off between the accuracy of the returned solutions and the overall execution time.

A *K*-Means strategy is suggested by [21] to identify communities in social networks. In contrast to traditional clustering algorithms, where the emphasis is mainly on profile similarities, the authors emphasize how people connect and consider their mutual interests and activities (e.g., age, gender, education). The study provides a "common social activity" factor, which includes both similarities in terms of features and the type of interactions between people. Adopting *K*-Means also has substantial drawbacks, primarily due to the random initial phase, which entails selecting random first seeds or centroids, which leads to defective solutions. Additionally, *K*-Means does not scale well when there are many attributes to take into account, which is the issue in the context of social networks.

A group of researchers [22] have developed a technique for extracting non-overlapping clusters from large attributed networks that use semantic and topological

distances, two distance measurements. The suggested heuristics consider category and quantitative variables when calculating the semantic distance. They employ the l -neighborhood search method for the topological distance, which creates a collection of nodes that paths of length l may reach. The clustering operation is completed by calculating the maximum of the two distances between nodes. It is, nonetheless, ideally suited for social media.

In the context of social networks, [23] suggested a heuristic to cluster attributed graphs. The authors approached the clustering problem by considering vertex connection and structural similarity, which measures how many vertices two vertices have in common. To form non-overlapping clusters, their heuristic uses the K -Means algorithm's idea of centroids. To create homogeneous clusters that consider vertex connection and structural similarity or the shared neighbors of two vertices according to a user-defined weight, two distance measurements are used similarly to [22].

In order to find communities in complex networks, this study [24] enhanced the Cuckoo Search Optimization (CSO) technique with a Genetic Algorithm (GA). Premature convergence, delayed convergence, and local trap are issues mainly with the CSO algorithm. In order to enhance exploration and exploitation in complex networks, GA has proved highly successful in detecting communities. In order to boost the CSO's speed and accuracy, GA operators have been deployed dynamically. Depending on the level of exploration and exploitation, the population size is dynamically changed. Normalized Mutual Information (NMI) is employed as an optimization function with the modularity [25] as an objective Function.

Discovering communities from disjoint complex networks using multi-layer ant colony optimization is proposed by [26]. The proposed multi-layer ACO (MLACO) solves multi-objective community detection problems and improves convergence by applying a parallel search method.

Numerous research studies use clustering algorithms [27] to create identical sets of objects in various application domains, such as social networks or cloud computing environments. For instance, most of the works discussed above use traditional techniques based on K -Means or K -Medoids. Additionally, several research papers use evolutionary methods like genetic algorithms or simulated annealing. The majority of community detection methods based on evolutionary algorithms [28] suffer from slower convergence rates and a propensity to converge to local optimum solutions, despite the fact that many of them have been available in the literature thus far. The clustering methods used in the current research are applied to the data center to construct homogeneous groups of servers in terms of chosen attributes. Through efficient resources management of the physical infrastructure, which is essential for service providers, the complexity of the service function chain (SFC) placement is reduced thanks to these solutions. To our knowledge, no current research has studied topology-aware clustering in the context of a dynamic and proactive approach tailored to NFV to support resource allocation techniques.

3 Problem statement, assumptions and system model

The physical network infrastructure is defined formally in this part, along with a list of keywords used throughout the study. Before delving into the mathematical model employed by the ILP solution, the game theory-based method, and the meta-heuristic-based approach, the problem statement is first introduced.

3.1 Definitions and notations

The formal description of the proposed mathematical model is presented, along with the notations used. Formally, attributed graphs extend the concept of graphs by enriching nodes with attributes. An attributed graph $G = (V, E, A)$ consists of a set of V nodes, a set of links interconnecting them E , and the set of node attributes A [1]. Thus, our physical network infrastructure is represented as an attributed graph.

Table 1 summarizes the different parameters of our clustering model with the different inputs.

3.2 Problem statement

In Network Function Virtualization, the problem of clustering attributed graphs is defined as a Quadratic Assignment Problem, which is recognized in the literature as one of the most challenging optimization problems. The purpose is to allocate n facilities to n locations typically. The price is determined by multiplying the flow between the facilities by the distances between the locations and the future cost of constructing facilities at each site. Accordingly, the goal is to assign each facility to

Table 1 Notations utilized subsequently in the paper

Notations	Descriptions
C	Set of clusters or a partition
C_i	Cluster C_i
$\text{Sim}(i, j)$	Similarity ratio between servers i and j
$\delta(C)$	Density function of a partition C
V	The set of servers
A	The set of servers attributes
E	The set of links
$E(C_i)$	The set of links within cluster C_i
$\delta(i, j) \in \{0, 1\}$	Equals 1 if servers i and j belong to the same cluster and 0 otherwise
A	The adjacency matrix where $A_{i,j} = 1$ if servers i and j are directly connected and 0 otherwise.
k_i	The degree of server i
x_i^d	The tuple of attributes associated with the server i
$\gamma_i^c \in \{0, 1\}$	Equals 1 if server i belongs to cluster c and 0 otherwise

a place where the overall cost is either diminished or increased, based on the fundamental characteristics of the problem under consideration.

In our context, the locations stand for the clusters that will be formed, and the facilities are the servers. In conclusion, identical facilities are grouped in one specific place, while distinct facilities are partitioned into several locations. F_1 and F_2 , which are the cost functions, are presented in the following subsections. Since the Quadratic Assignment Problem is NP-Hard and our model reduces to it, it demonstrates that Network Function Virtualization's clustering of attributed graphs is equally NP-Hard [29].

With the aid of the notion of attributed graphs, we aim to offer a collection of clustering methods specifically suited to the service function chaining issue and cloud-based virtual network service orchestration. The technique of clustering makes an effort to extract non-overlapping clusters using a combined distance that considers node attributes and takes advantage of the properties of the underlying networking topologies. It requires optimizing at least two objective functions to get a good clustering of an attributed graph (see [1, 3] for more information). Compositional and structural dimensions will always be in trade-off with one another. The nodes and connections, respectively, fall under these dimensions.

For node attributed graphs, the objectives are:

- (1) *The structural quality of the clusters* the modularity function [4] is considered, where higher modularity corresponds to better clustering. Thus, this measure is maximized.
- (2) *The intra-cluster homogeneity of the node attributes* The similarity-based measure [1], which is the key to building clusters since cluster members are grouped only when being similar is considered. Thus, this measure is maximized.

Both of these objective functions have a strictly technical rationale. The attributes of the servers and the network architecture, consisting of several servers and links, are sufficient to calculate these measurements, which are simple to execute and do not require any extra information. Due to their linear growth, they do not require a lot of processing time. A minimum of two optimization objectives must be considered for node-attributed graphs, as stated in the survey [1]. Before implementing the clustering techniques suggested, a trade-off between compositional and structural dimensions will be required.

3.3 Assumptions

The network topology's pre-processing task is a significant factor that should be mentioned. The data must be processed before being used with clustering or any unsupervised learning approach. According to the on-demand clusters' requirements, the servers' links are filtered in terms of bandwidth and latency. For instance, if a link linking two servers is overloaded, the clustering procedure will not include it. All three approaches can rely on the information (e.g., CPU, Memory, Disk Storage) that can be provided by the data plane using monitoring tools (e.g., Nagios) as well as the current

network infrastructure topology (the available servers with their inter-connectivity). Without loss of generality, it is assumed that the network or service provider leverages such mechanisms and tools.

3.4 VALKYRIE clustering model

3.4.1 Objective functions

The aim is to build similar clusters while considering not only their topological distance but also the capacity of the links interconnecting them to communicate with each other while within the same clusters.

1. **F1 (1)** aims to form clusters of servers with similar characteristics. The similarity will therefore be maximized by **F1**. Nevertheless, clusters that share one or more attributes but have diverse characteristics may be built. For instance, one could intend to create CPU-intensive, energy-efficient clusters or even multi-attribute clusters that combine numerous features.

$$F1 = \frac{1}{2} \times \sum_{\forall c \in C, \forall i \in V, \forall j (j \neq i) \in V} \text{Sim}(i, j) \quad (1)$$

where :

$$\text{Sim}(i, j) = \gamma_i^c \gamma_j^c \left[\frac{1}{1 + \sqrt{\sum_{\forall d} (x_i^d - x_j^d)^2}} \right]$$

2. **F2 (2)** maximizes the modularity for better clustering. Modularity is a measurement of the quality of partitioning the nodes of a graph or network into communities or classes. High modularity (see [1] for a working example) features sparse connections between nodes in different modules but dense connections between nodes within modules. It is primarily used to analyze social networks [1].

$$F2 = \frac{1}{2|E|} \sum_{\forall i \in V, \forall j (j \neq i) \in V, \forall c \in C} [A_{ij} - \frac{k_i k_j}{2|E|}] \delta(i, j) \quad (2)$$

$\delta(i, j)$ is the Kronecker delta which returns one if i and j belong to the same cluster and zero otherwise. In our problem formulation, it is replaced with the product of γ_j^c and γ_i^c which is equivalent to the Kronecker delta. Thus, **F2** becomes:

$$F2 = \frac{1}{2|E|} \sum_{\forall i \in V, \forall j (j \neq i) \in V, \forall c \in C} [A_{ij} - \frac{k_i k_j}{2|E|}] \gamma_j^c \gamma_i^c$$

3.4.2 Constraints

$$\forall c \in C : \sum_{i \in V} \gamma_i^c = 1 \quad (3)$$

$$\forall i \in V : \sum_{c \in C} \gamma_i^c \geq 2 \quad (4)$$

$$\forall c \in C, \forall i \in V : \gamma_i^c \in \{0, 1\} \quad (5)$$

Each server can only be a part of one cluster, thanks to constraint (3). The model, therefore, is free of outliers. The generated clusters must contain at least two servers, thanks to constraint (4). Depending on the requirements of the decision maker running the model or techniques, this option might be supplied with a different value. The decision variable is expressed by constraint (5) as binary (its value is either one or zero).

4 CRO-based approach

4.1 CRO approach description

A population-based meta-heuristic termed chemical reaction optimization utilizes chemical reactions as a framework to address challenging optimization problems. The molecule representing a potential solution for the issue under consideration is an essential part of CRO. The potential energy of every molecule in the population is what distinguishes it. CRO aims to improve population quality over iterations by implementing a sequence of reactions. Uni-molecular and multi-molecular reactions make up the two main groups into which these reactions are classified.

- *Uni-molecular reactions* These reactions start with a single molecule and might produce just one new molecule, called On-wall ineffective collisions. Likewise, it could lead to the formation of two distinct solutions, which would be a decomposition as a single molecule would break into two separate molecules with distinct potential energies.
- *Multi-molecular reactions* This set consists of two entering molecules that, following the reaction, might produce a single molecule via synthesis or two new solutions out of an inter-molecular reaction. Here, synthesis is the process of two molecules coming into contact to form a new one. In contrast, two molecules hit in an ineffectual inter-molecular collision bounce back, and each is slightly altered though not in a way that affects the other.

While decomposition and synthesis function primarily as diversification techniques, both forms of ineffective collisions enable intensifying the population by carrying out a local search in the solution space. This yields a balanced set of

Table 2 First molecule encoding

Cluster ID	1	2	3
Included nodes	1,3,7,10	2,4,6,8	5,8,9

Table 3 Adopted molecule encoding

Server ID	1	2	3	4	5	6	7	8	9	10
Cluster ID	1	2	1	2	3	2	1	3	2	1

operations that incorporates all these reactions, facilitating a successful search in the solution space and enabling the identification of nearly optimum solutions for the under consideration optimization problem [7]. The following parameters drive the overall CRO approach:

- *Potential energy* This is a quality measure that measures the objective function of a given solution/molecule.
- *Kinetic energy KE* This measure quantifies the tolerance of the whole system to accept solutions that are worse than the initial ones.
- *Decomposition rate A* At each iteration, this value determines the uni-molecular reactions to be applied: decomposition or on-wall ineffective collision.
- *Synthesis rate B* At each iteration, this value determines the multi-molecular reaction to be applied: synthesis or inter-molecular ineffective collision.
- *Maximum number of iterations* This is a counter that is used as a stopping criterion. Once the maximum number of iterations is reached, the CRO heuristic converges to the optimal solution available in the current population.

Depending on the optimization problem under consideration, these parameters, which the network administrators provide as inputs, should be tweaked to produce the best optimum solution. The application of the chemical reaction optimization meta-heuristic to our clustering problem is discussed in the following section. Additionally, it specifies the operators employed in each of the fundamental processes as well as the molecular encoding.

4.2 Molecule encoding

One possible way to encode the solution is to use the grouping technique defined in [30] where each cluster will represent the group of servers it contains as described in Table 2.

Table 2 shows a topology of 10 nodes distributed in three different clusters. Using this encoding, if the usual operator used for the chemical reaction optimization or genetic algorithm is applied, then the algorithm would produce overlapping clusters that violate the constraint (3), requiring each node to belong to only one cluster. To overcome this problem, it is necessary to go with the encoding shown in Table 3.

This encoding consists of an N size array where N is the number of nodes in the substrate graph; each server is assigned to a selected cluster with a specific label.

4.3 Initial population

Our CRO approach consists of a centroid-based clustering where a set of points are specified in the data-set and then assigned to the rest of our servers to one of the defined centroids based on the similarity measure defined in Sect. 3.2. Prior to executing the different CRO operators, the choice of the initial k centroids is a critical step. The challenge in this phase is ensuring that these centroids are spread out enough to reflect the distribution of our servers in the data set to allow us to distinguish the different clusters/groups within the studied network topology. Inspired by K -Means++ [31], the variance of our considered servers in terms of their attributes and defining the initial centroids accordingly is employed. This ensures that the initialization phase would not be blind or random. The chosen centroids would represent our servers' variety, allowing us to enhance the quality of our initial population.

4.4 Elementary reactions

This section will describe how each CRO operator is used to enhance the quality of the formed solutions present in the initial population. As mentioned previously explained in Sect. 4.1, the operators of CRO consist of two major sets:

- *Uni-molecular reactions* The first reaction in this set is called the *On-wall ineffective collision*. This operator consists of a single input molecule representing a potential solution called $M1$ in Algorithm 1 and then results in a new solution that is called Mo . Next, a set of nodes are selected such that they have low similarities within their assigned clusters in $M1$, re-compute their similarities to the updated centroids, and then re-assign them to the cluster to which they are the most similar in terms of the selected attributes (steps 1–3 in Algorithm 1). Hence, it will allow us to correct the misplaced nodes and re-enforce the homogeneity of the formed clusters. Once the new solution is formed, its modularity is computed. If it is higher than the initial input solution $M1$, it is destroyed, and Mo is added to the population. Otherwise, $M1$ is kept, and resulting Mo is destroyed.

Algorithm 1 On-wall ineffective collision

```

Input: One initial molecule  $M_1$ , Servers attributes  $A$ 
1:  $Nodes \leftarrow \text{Argmin}(F_1, V)$ 
2:  $SimM_1 \leftarrow \text{Compute } F_1 \text{ with respect to } M_1$ 
3:  $Mo \leftarrow \text{Re-assign each node to its closest centroid based } F_1$ 
4: If  $Modularity(Mo) > Modularity(M_1)$ 
5:   Destroy  $M_1$ 
6:   Add  $Mo$  to the population
7: else
8:   Destroy  $Mo$ 
9: end If

```

The second operator in this category is the *Decomposition*. This reaction also starts with a single molecule, M_1 but results in two distinct solutions D_1 and D_2 . A random node is selected in the initial solution M_1 , which ranges between 0 and N , where N is the number of nodes in our data set. The first part of this molecule, M_1 , is kept in the first resulting solution, D_1 , meaning that the clustering from Server[0] to Server[R] is copied (step 2 Algorithm 2) and then assigned the rest of the nodes to their closest centroids based on similarity (step 3 Algorithm 2). The second solution is constructed from the second part of M_1 . However, this time nodes from Server[$R + 1$] to Server [N] are kept (step 4 algorithm 2) and assigned servers 0 to R to their closest cluster again based on similarity (step 5 Algorithm 2). Once the new offspring is formed, the modularity of both D_1 and D_2 is computed, and if one of them is higher than that of the initial solution M_1 , the latter is destroyed while adding D_1 and D_2 to the population. Otherwise, the new solutions are destroyed, and the population remains unchanged.

Algorithm 2 Decomposition

```

Input: One initial molecule  $M_1$ , Servers attributes, Number of servers  $N$ 
1:  $R \leftarrow \text{rand}(0, N)$ 
2:  $D1[0:R] \leftarrow M_1[0:R]$ 
3: Assign nodes from  $R+1$  to  $N$  to their closest centroids based on similarity to form  $D1[R+1:N]$ 
4:  $D2[R+1:N] \leftarrow M_1[R+1:N]$ 
5: Assign nodes from 0 to  $R$  to their closest centroids based on similarity to form  $D2[0:R]$ 
6: If  $Modularity(D1) > Modularity(M_1)$  Or  $Modularity(D2) > Modularity(M_1)$ 
7:   Destroy  $M_1$ 
8:   Add  $D_1$  and  $D_2$  to the population
9: else
10:  Destroy  $D_1$  and  $D_2$ 
11: end If

```

- *Multi-molecular reactions* The first operator in this category is the Synthesis, and its entire process is described in Algorithm 3. As an input, two molecules represent two potential solutions chosen randomly from the current population, M_1 and M_2 . The collision between these two elements results in a single output molecule M_s which is also a probable solution to our optimization problem. This reaction consists of choosing a random point on both parents M_1 and M_2 and populating the offspring solution with elements from the first parent ranging from server[0] to server [R]. Then, the rest of M_s is formed by the second parent ranging from server [$R + 1$] to server [N], where N is the number of servers considered in our data-set (step 2 in Algorithm 3). Suppose the resulting modularity of the newly formed solution

M_s is higher than that of the initial M_1 and M_2 . In that case, these two are later discarded while M_s is added to the population. Otherwise, the population remains the same, and both M_1 and M_2 are kept in the population, while M_s is destroyed (steps 4–8 in Algorithm 3).

Algorithm 3 Synthesis

Input: Two initial molecule M_1 and M_2 , Number of servers
 1: $R \leftarrow \text{rand}([0, N])$
 2: $M_s \leftarrow M_1[1:R] \cup M_2[R+1:N]$
 3: **If** $\text{Modularity}(M_s) > \text{Modularity}(M_1)$ **Or** $\text{Modularity}(M_s) > \text{Modularity}(M_2)$
 4: Destroy M_1 and M_2
 5: Add M_s to the population
 6: **else**
 7: Destroy M_s
 8: **end if**

The second operator in this category is the Inter-molecular ineffective collision. This reaction also starts with two molecules, M_1 and M_2 , resulting in two potential solutions, C_1 and C_2 . In this reaction, each new molecule is generated from its parent independently of the second one. Therefore, the on-wall ineffective collision defined in Algorithm 1 is used to generate two distinct solutions.

4.5 Overall CRO-based clustering algorithm

Algorithm 4 describes the step-wise methodology to execute the overall steps of the CRO-based clustering approach process. The first step consists of generating a set of feasible solutions equal to the population size. It starts first by generating a random number \mathbf{B} between $[0,1]$ (step 4 Algorithm 4); if \mathbf{B} is higher than \mathbf{Mol} , then it randomly selects one Molecule from the initial population that was generated. The next step is to verify the number of hits attained if it is higher than the value of \mathbf{A} (specified as an input to the Algorithm 4), then it applies the decomposition process 2. Otherwise, the on-wall ineffective collision process (Algorithm 1) will be executed. The other scenario is for the case where the generated number \mathbf{B} is lower than the input \mathbf{Mol} . Here, it applies the multi-molecular operators. It is first checked if the set kinetic energy is lower than the value of \mathbf{B} (specified as an input to the Algorithm 4). The synthesis operator (Algorithm 3) is executed. Otherwise, it applies the inter-molecular ineffective collision process and decrease kinetic energy. This process of CRO operators will be repeated as long as the total number of iterations has not been attained, which is the stopping criterion for this algorithm. Once the population reaches the maximum number of iterations set as an input, the optimal solution is chosen for modularity as defined in Sect. 3.4. The solution that returns the highest modularity is selected as the optimal solution.

Algorithm 4 Overall CRO Algorithm

Input: Number of servers, Servers attributes, Molecules, Population size, MaxIteration, $A=0.6 \cdot \text{MaxIteration}$, $B=0.3 \cdot \text{MaxIteration}$, KineticEnergy=MaxIteration, Num-Hits=0
Output: Final Solution O

```

1: Mol = rand([0,1])
2: while Num-Hits < MaxIteration do
3:   B = rand([0,1])
4:   If B > Mol do
5:     Molecule = rand(Molecules)
6:     If Num-hits > A do
7:       Execute Decomposition Algorithm 2 on Molecule
8:     else
9:       Execute On-wall ineffective collision Algorithm 1 on Molecule
10:    end If
11:  else
12:    Molecule_M1, Molecule_M2 = rand(Molecules)
13:    If KE < B do
14:      Execute Synthesis Algorithm 3 on Molecule_M1 and Molecule_M2
15:    else
16:      Execute Inter-molecular ineffective collision on Molecule_M1 and Molecule_M2
17:    end If
18:    KineticEnergy = KineticEnergy - 1
19:  end If
20: Num-Hits = Num-Hits + 1
21: end while
22: {BestMolecule, Modularity} = Argmax(Molecules, F2)
23: return Final solution BestMolecule and its Modularity

```

5 A game theory-based approach

In this approach, the Irving's algorithm was used to find what are known as stable matching pairs between servers, such that no two non-matched servers prefer each other more than their actual matching. This algorithm comprises the preference list computation phase (see algorithm 5) and the preference cycle elimination phase (see Fig. 2). In the first phase, the attributes of each server were given as inputs, along with the list of servers in the topology. The preference table was then built by calculating the similarity between each server i and every other server j in the substrate network. The similarity values calculated were ranked (sorted) in descending order and formed the preference table as shown in Algorithm 5.

Algorithm 5 Preference List Computation

Input: Set of attributes for all servers A , list of servers V
Output: *PreferenceList*

```

1: SimArray = {}
2: for each server  $i$  in  $|V|$  do
3:   for each server  $j$  in  $|V|$  do
4:     Compute  $\text{Sim}(i, j)$  from equation (1)
5:     SimArray.append( $\text{Sim}(i, j)$ )
6:   Sort(SimArray)
7:   end for
8: end for
9: return PreferenceList

```

After the preference table was built, each server should be uniquely paired with only one other server in the substrate, which is accomplished by applying the Irving Algorithm [9]. The inputs of this algorithm are the preference table, along

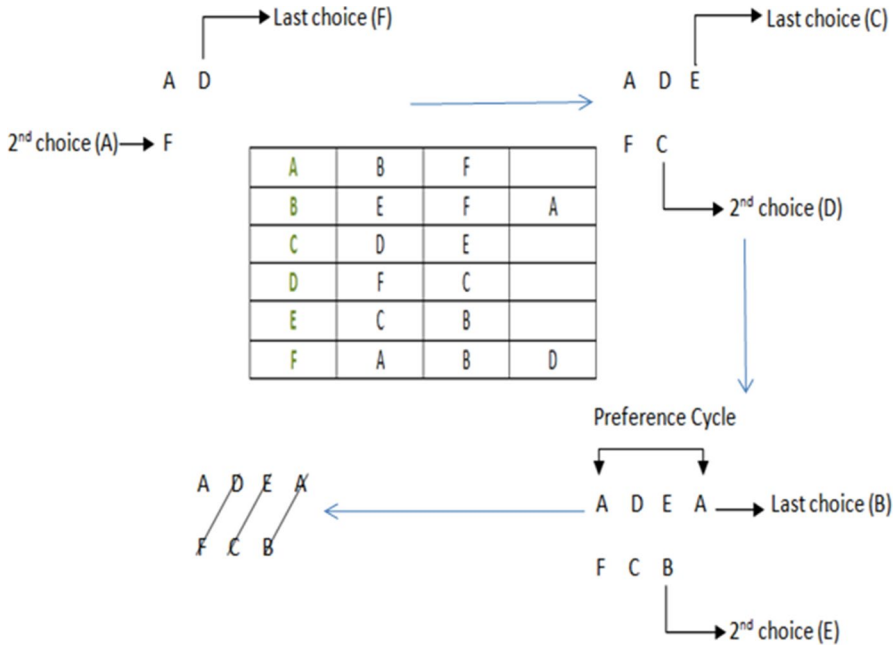


Fig. 2 Preference cycle elimination process.

with the list of servers in the substrate topology, with their respective attributes, as briefly described in Algorithm 7. The Irving algorithm matches each server to only one other server. Each server i sends its proposal to the preferable server j computed in the previous algorithm and specified in the preference table. If any server has more than one proposal at a time, it will keep the best one and discard the others. This process is repeated until each server has only one proposal. Based on the position of the current proposal for each server in the preference table, all the less preferable servers will then be discarded from the preference table. This generates a shortlisted preference table, such as the one in Fig. 2. The last step in the Irving algorithm is eliminating all the preference cycles in the shortlisted preference table.

This process is performed by checking all the rows in the shortlisted preference table with more than one preference. For instance, as shown in Fig. 2, server A has two preferences, namely, B and F (B is better and preferable). In order to detect the current preference cycles, starting from the first row, the `second_choice(server A)` and the `last_choice(second_choice(server A))` are noted, and this step is repeated until the preference cycle is detected, as shown in Figure. Once this cycle is detected, the following preferences between (A and B), (E and C), and (D and F) are discarded. This process is repeated until each server has only one preference in the table, called the stable matching solution.

Algorithm 6 Irving Algorithm

Input: The servers set V and $PreferenceList$.
Output: $Matching_pairs$
1: $Matching_pairs \leftarrow Irving(V, PreferenceList)$
2: **return** $Matching_pairs$

The initial clusters are formed by pushing the generated pairs into different clusters based on similarity characteristics. This clustering is improved by checking if adding one of the servers to a specific cluster would improve the modularity value for the whole partition. If this is the case, the server is added to one of the clusters. This process is performed iteratively to maximize the modularity value for the entire partition while minimally affecting the similarity value for each formed cluster, as shown in Algorithm 7.

Algorithm 7 Outlier-free clusters

Input: $Matching_pairs$, $Clusters$
Output: Clusters of servers
1: **foreach** $Cluster \in Clusters$
2: **foreach** $pair \in Matching_pairs$
3: Compute $Sim(i, j)$ from equation (1)
4: **if** ($pair$ maximizes $F2$) **then**
5: $Cluster \leftarrow pair$
6: $Matching_pairs \leftarrow Matching_pairs - \{pair\}$
7: **end if**
8: **end for**
9: **return** Clusters of servers

The centroid value of each cluster is updated dynamically when the server is added or removed from the cluster. Two flavors of the game theory approach are proposed in this paper, one of which focuses more on maximizing the modularity with the existence of outliers. At the same time, the other one has no outliers with a reduced modularity value. Algorithm 7 is invoked to generate clusters without outliers, which improves the Irving algorithm outcome. In that case, our solution will force each outlier server to join one of the clusters based on the similarity value between this server and the centroid.

6 Asymptotic complexity analysis

A description of the algorithmic complexity of our proposed suite of VALKYRIE approaches is provided. It is worth mentioning that this complexity analysis is performed for the worst-case scenario. The way our mathematical model is described is NP-Hard since it embodies the form of the well-known Quadratic Assignment Problem [29].

CRO-based approach Our CRO-based approach consists of four algorithms, as described in Sect. 4.1. The complexity of each algorithm is reported as follows:

- The initial population task runs in $O(\text{Popsiz}e \times |V|)$. This is because we loop over the entire set of molecules (Popsiz)e, i.e., candidate solutions and perform a set of basic operations (e.g., array creation, adding the formed molecules to the population and so on) on the entire set of nodes V .
- Algorithm 1 runs in $O(|V|^2 + |V|)$. As we maximize the modularity function, we loop over the entire set of nodes twice in the defined equation, thus the term $|V|^2$. Furthermore, the set of nodes with low similarity is calculated in the order of $O(|V|)$.
- Algorithm 2 also runs in $O(|V|^2 + |V|)$. The same analysis can be drawn from Algorithm 1.
- Algorithm 3 runs in $O(|V|^2)$. As we maximize the modularity function, we loop over the entire set of nodes twice in the defined equation, thus the order $O(|V|^2)$.

Based on this analysis, Algorithm 4 (CRO-based approach) complexity is $O(|\text{Popsiz}e| \times |V| + \text{MaxIteration} \times (|V|^2 + |V|))$. The first term comes from the initial population generation, and the second term comes from the complexity of Algorithms 1 and 2, respectively. Finally, the entire process goes through MaxIteration iterations.

Game theory-based approach Similarly, three algorithms were employed, and their complexities are computed as follows:

- Algorithm 5 runs in $O(|V|^3 \times \text{Log}(|V|))$ as it uses TimSort algorithm to sort the similarity array which runs in $O(|V| \times \text{Log}(|V|))$ [32]. In addition, as we maximize the modularity function, we loop over the entire set of nodes twice in the defined equation, thus the order $O(|V|^2)$. Therefore, this fact justifies the cubic power over the set of nodes $|V|$.
- Algorithm 6 runs in $O(|V|^2)$. The matching pairs are computed while looping through the entire set of nodes twice in a nested structure.
- Algorithm 7 runs in $O(|V|^3)$. In addition to the computation of the matching pairs performed in $O(|V|^2)$, the similarity is checked in $O(|V|)$ inside the two nested loops.

Based on this asymptotic complexity analysis, the GT-based approach complexity is $O(|V|^2 + |V|^3 \times \text{Log}(|V|))$. Furthermore, the CRO-based technique exhibits lower overhead compared to the GT-based approach. Finally, it is fair to correlate the results with this asymptotic complexity analysis to better appreciate one approach over the others in terms of their performances in each metric.

7 Evaluation

In this section, VALKYRIE is evaluated. The chemical reaction optimization algorithm is assessed both on small and large-scale networks. The quality and effectiveness of the solution with and without the presence of ground truth are evaluated. The ground truth refers to what is known to be accurate or authentic, supplied by direct observation and measurement (i.e., empirical evidence) instead of the information

given by inference. In the context of clustering, it refers to the clusters that need to be discovered by the clustering algorithm [1].

7.1 Setup

The mathematical model is implemented using Python 3.7 and solved thanks to Gurobi 7.5.1 [8] to get the optimal solution. Gurobi is shipped with a Python plug-in called “gurobipy” that offers handy object-oriented modeling constructions and an API for all Gurobi functionalities. Our chemical reaction optimization and game theory procedures were implemented using Python 3.7. The experiments and implementations were carried out on a physical machine composed of 8 CPU cores and 16 Gig of Memory.

Two types of network infrastructures were considered to evaluate VALKYRIE. The topologies were generated using the NetworkX library [33] which is a Python module for creating, processing, and studying a network’s complex structure, dynamics, and functions. Section 7.1.1 provides details about the scenarios, and Table 4 details the types of infrastructure considered in the experiments. Finally, we report the average values for all the experiments, which were repeated ten times as they were sufficient after a set of trials.

7.1.1 Scenarios

Our approaches are evaluated under a set of scenarios. It is worth mentioning two variants of the game theory-based approach. The GT-based approach uses algorithms 6, 7, and 8, while the Enhanced-Game Theory (E-GT)-based approach improves GT that suffers from outliers.

Scenario 1

In this scenario, it is assumed that the topology is modular that is considered the ground truth. In such a scenario, the distribution of CPU cores per module is defined as follows: 4–11 for the first module, 12–24 for the second one, and 25–36 for the third one. This can be viewed as if we had three different domains where the servers were physically present. However, this differs from the following scenario, where no assumption is made on the server’s domains.

Scenario 2

Table 4 Degrees of connectivity in our scenarios

Network size	Scenario 1	Scenario 2
20	2	3
50	7	3
100	14	3
200	29	3
300	44	3
500	74	3
1000	149	3

In this scenario, different topologies regarding the number of servers with fixed connectivity degrees for each server are defined. The servers in this topology are randomly connected, and they are not modular, as in the case of the previous scenario. The distribution of CPU cores is drawn between 4 and 64 units.

7.2 Performance metrics

VALKYRIE is evaluated according to the following metrics to assess its effectiveness:

- (1) *Runtime* the time taken by the different approaches to partitioning the network into the number of desired clusters is calculated.
- (2) *Similarity* the average similarity of all the clusters is evaluated.
- (3) *Modularity* evaluates how dense the connections between the nodes are within the clusters and how sparse they are while in different clusters.
- (4) *Density* computes the proportion of edges that lie within the clusters, and a higher density corresponds to a better clustering. It is defined as follows:

$$\delta(C) = \frac{1}{|E|} \sum_{\forall C_i \in C} |E(C_i)|$$

where $E(C_i)$ is the set of edges inside the i th cluster.

- (5) *Outliers* the number of servers that do not belong to any cluster is computed.

It is worth mentioning that Density is the validation technique used to assess our clustering approaches. Nevertheless, the values of all the metrics, shown on the y -axis of the Figures, are in the range $[0, 1]$, and the higher the value, the better it is.

8 Evaluation results

In this section, an analysis of the results is presented for three different topologies to test our approaches. For all the evaluations regarding ILP, the results for network sizes more than 200 as the runtime is exponentially increasing and takes several hours are not reported. Instead, a sub-optimal version of ILP is devised, denoted by ILP_subopt by tuning the optimality gap parameter defined in [34] to obtain quick solutions.

8.1 Modular topology with variable connectivity degree

Based on Fig. 3, it can be seen that the clustering time for the ILP increases exponentially as the network size increases. For example, it takes hours once the number of servers exceeds 200. This behavior is expected as the ILP attempts to find the exact solution that justifies adopting one of the developed heuristics. Concerning the other heuristics approaches that were considered, it can be seen that CRO increases

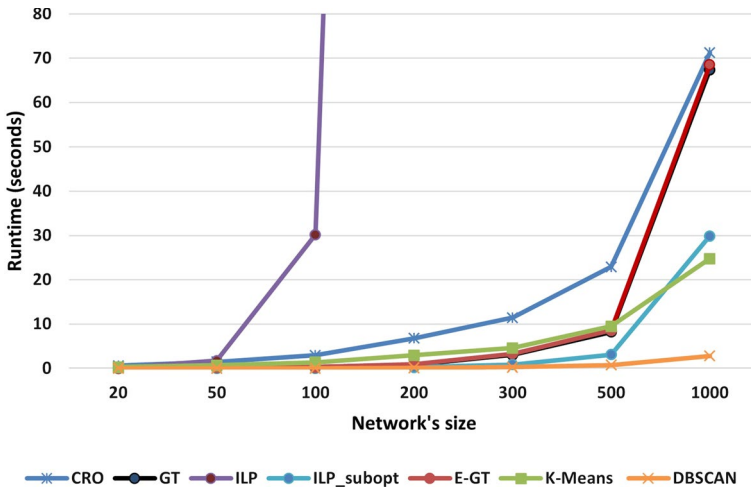


Fig. 3 Clustering time

from 0.6 s for 20 servers to 71 s for a topology of 1000 servers. For the two flavors of GT approaches, the runtime value ranges from 0.2 to 67 s for GT and from 0.03 to 68 s for E-GT, while ILP_subopt ranges from 0.11 to 29 s. Although ILP_subopt has the minimum runtime, all the heuristics approaches are still acceptable for a clustering solution by a cloud provider since clustering is done periodically and proactively. DBSCAN [35] and K-Means [36] clustering algorithms mostly give less runtime, but as mentioned in the asymptotic analysis, the running time for our proposed approaches is reasonably acceptable.

Another metric that requires evaluation is similarity, as shown in Fig. 4. The ILP identifies the best value in terms of similarity compared to the other approaches for a small-scale environment (up to 200 servers). When CRO and the two flavors of GT heuristics are compared, it can be seen that the values are close, with GT and E-GT demonstrating slightly better results in terms of similarity. This may be attributed to the fact that the GT approach has outliers. Although E-GT has slightly better similarity values, it compares unfavorably to CRO in modularity. About the Game Theory approach, not all servers are included in the formed clusters, which explains the difference in similarity. This difference is relatively small at an average of 0.05. When comparing our proposed algorithms with DBSCAN and K-Means, the results reveal that our proposed approaches have better similarity values regardless of the network's size.

Figure 5 shows the differences between the considered approaches in terms of modularity. The ILP and ILP_subopt solutions find the lowest values for all types of topologies compared to CRO and the two flavors of GT, mainly due to the simplex algorithm that finds the optimal tradeoff between both considered objective functions. As shown in Fig. 4, ILP finds the best similarity, but Fig. 5 illustrates that this impacts on the modularity. As for the heuristics of CRO and the two flavors of GT, it can be seen that GT finds better results for small topologies ranging from 20 to

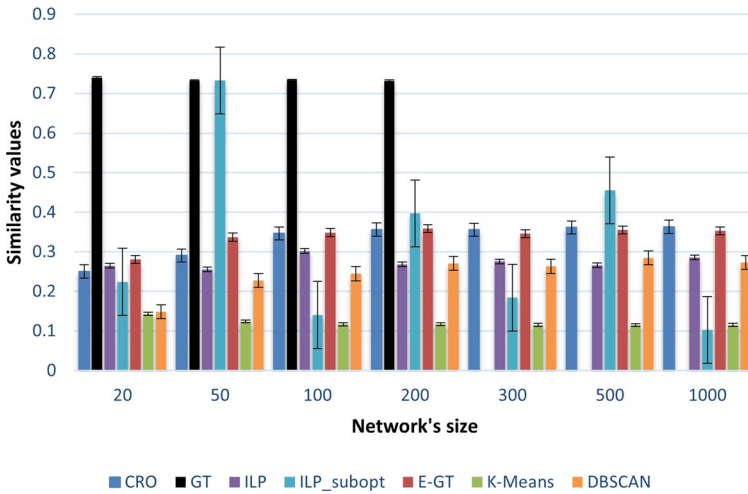


Fig. 4 Similarity

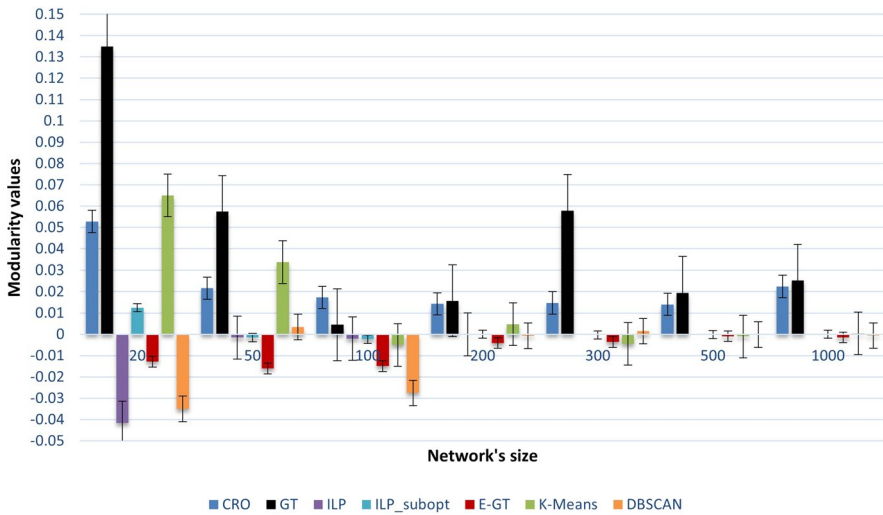


Fig. 5 Modularity

50 servers, and CRO takes the lead for medium topologies of 100 servers; the latter have pretty similar values for large topologies of more than 200 servers, with an advantage to the GT approach. The difference in modularity can be attributed to the GT approach's outliers. Another takeaway from this experiment is that our proposed approaches display better modularity values when compared with DBSCAN and K-Means regardless of the network's size.

The density is an internal validation measure used to evaluate the quality of the clustering when the ground truth is not known in advance, meaning that the clusters

are not known before the algorithms are applied. Based on Fig. 6, it can be seen that CRO and E-GT have almost the same density values, which are better than the other approaches. This is mainly because they do not return any outliers. The clusters formed by CRO and E-GT include more servers and are denser than those returned by GT, where the number of outliers jumps up to 25% of the substrate for large topologies composed of 1000 servers. Although ILP-SubOpt exhibits the highest density, it is not deployable since some clusters contain only two servers, which is not preferable for the network administrator. The density values obtained by our proposed approaches are mostly better regardless of the network's size than those obtained by the traditional DBSCAN and *K*-Means clustering techniques.

8.2 Random topology with fixed connectivity degree

Like the previously tested topologies, the ILP solution takes considerable time to converge to the optimal solution, as shown in Fig. 7. It increases sharply as the network size grows. The two Game Theory flavors approach takes less time to converge to the sub-optimal than the CRO and ILP_SubOpt approaches. All of them except the ILP_SubOpt solution have a lower runtime than the previous topology, where the degree of each node varies. This is expected because the substrate network is less complex when the degree is fixed than varying connectivity degrees. The latter means a higher number of edges, more computational time, and a higher cost for computing the modularity values. DBSCAN and *K*-Means techniques exhibit a slightly better running time than the other approaches, particularly for DBSCAN. However, our proposed approaches still have a reasonably good running time.

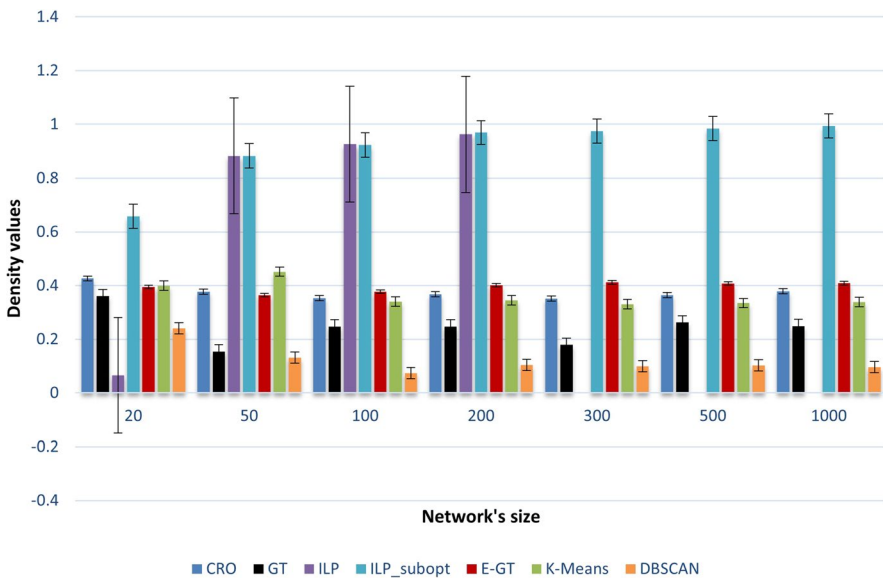


Fig. 6 Density

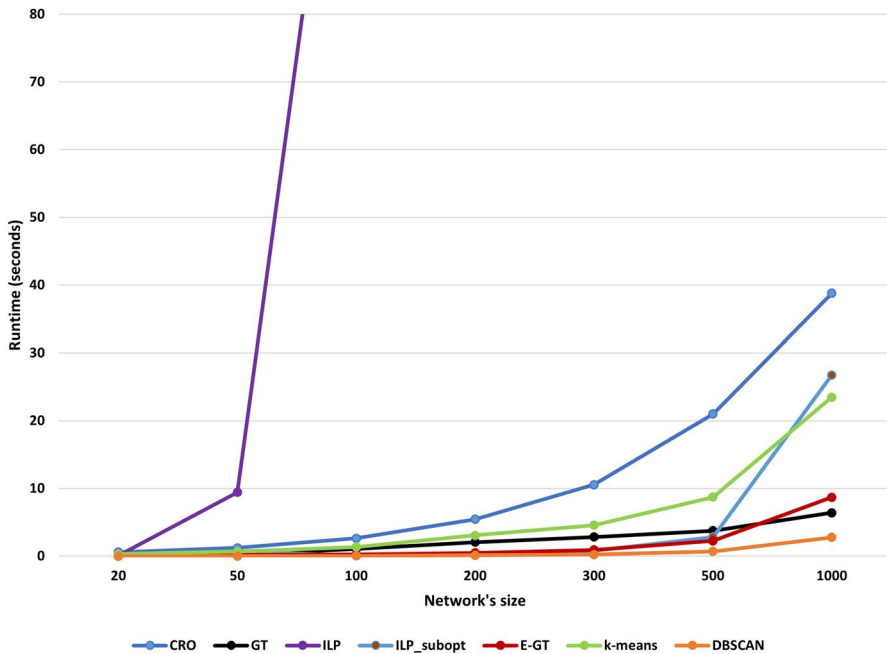


Fig. 7 Clustering time

When the degree is fixed, there is no significant difference between the CRO, GT, and E-GT approaches. This measure is more related to the servers' attributes than their connectivity. GT finds slightly better results compared to CRO when it comes to the similarity, as shown in Fig. 8, but again with an existence of outliers which explains this difference. ILP and ILP_subopt solutions exhibit the highest similarity values for the small-scale environment (up to 200 servers), and finally, GT and E-GT have relative values. The results have also shown that some of our proposed approaches outperform the results obtained by DBSCAN and K-Means.

The difference in modularity is higher in this topology between GT and all the other approaches, as shown in Fig. 9. It can be seen that GT finds better results for topologies composed of above 20 servers, but the number of outliers also jumps in this topology which explains this gap again. The ILP and ILP_subopt solutions have the same behavior compared to other topologies. However, our proposed approaches' modularity values are mostly better than the DBSCAN and K-Means in this setting.

CRO is the best in density for this type of topologies, as shown in Fig. 10, compared to GT and E-GT. However, E-GT has better values than GT since it has no outliers and increases the number of edges inside each formed cluster. ILP_subopt solution shows the best value in terms of density, although it puts the servers among the clusters unevenly. One cluster may be formed only with two servers which is not ideal and appropriate for a cloud provider. Again our proposed approaches have shown better density values than the traditional DBSCAN and K-Means algorithms.

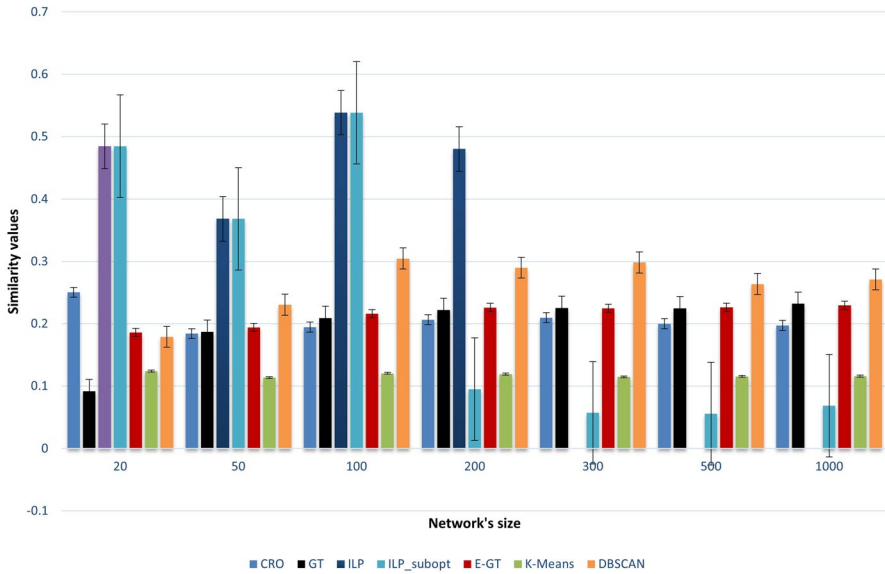


Fig. 8 Similarity

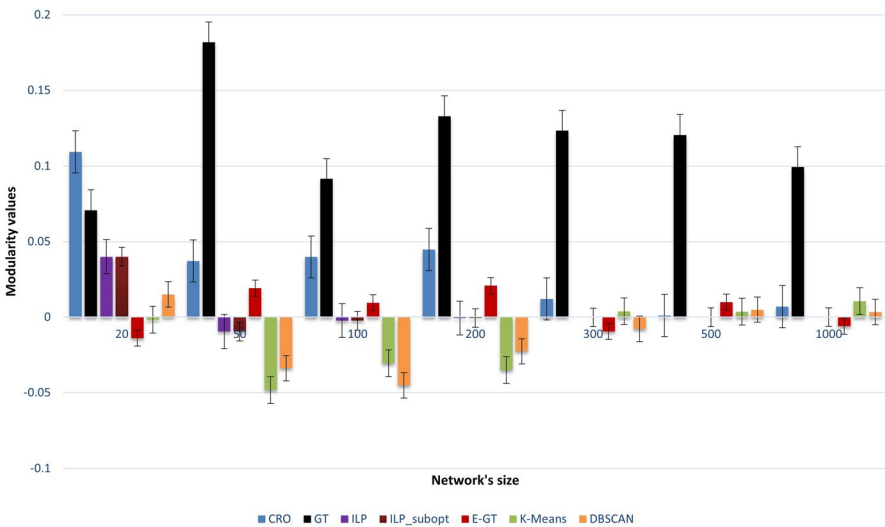


Fig. 9 Modularity

As shown in Table 5, the number of outliers increases linearly as the network size increases for both topologies. This only occurred in one of the proposed approaches, the GT approach, as all other approaches respect the constraints defined in the mathematical model. Having some outliers had an impact on getting better values for similarity (Fig. 8 and modularity (Fig. 9, which could benefit a cloud provider since

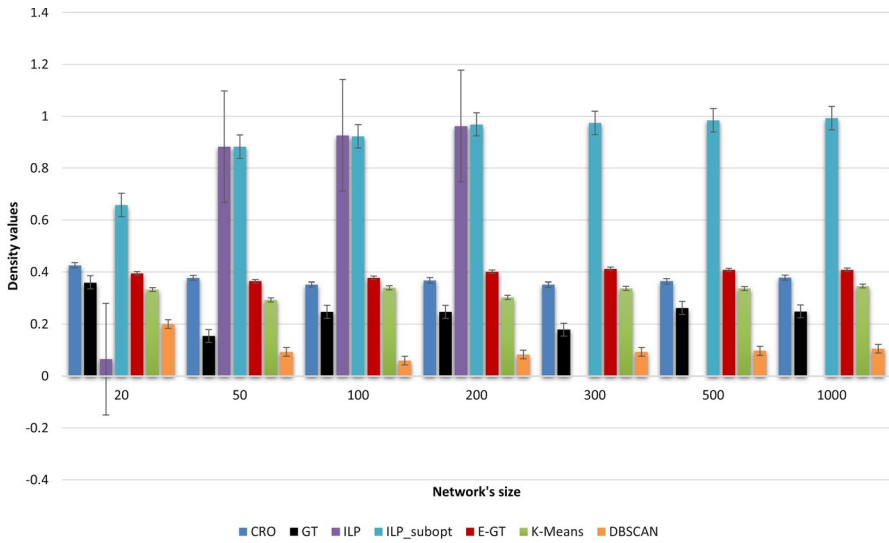


Fig. 10 Density

Table 5 Number of outliers using Game Theory

Network's size	GT (scenario 1)	GT (Scenario 2)
20	7	11
50	23	19
100	22	29
200	45	76
300	113	139
500	104	207
1000	242	454

there could be certain cases where the Service Function Chain (SFC) request has a specific requirement and preference for a cluster to be deployed with higher modularity and similarity values. On the other hand, both topologies detected some outliers when applying the DBSCAN algorithm, as shown in Table 6.

8.3 Discussion and overall observations

Based on the presented results, we draw the following conclusions:

- The clustering time is influenced by the number of servers, links, and each server's connectivity degree.
- The Theory-based game approach is better in modularity than other proposed approaches, but it suffers from outliers. It is calculated only with the clustered servers as the others are outliers.

Table 6 Number of outliers using DBSCAN.

Network's size	DBSCAN (scenario 1)	DBSCAN (Scenario 2)
20	5	3
50	1	3
100	1	0
200	0	1
300	0	0
500	0	0
1000	0	0

- ILP solution has the best similarity value for the small-scale environment (≤ 200 servers). In contrast, GT has a slightly better similarity value than other heuristics but suffers from outliers.
- If a cloud provider prefers no outliers when performing the substrate clustering, then CRO is the best heuristic solution since it has close similarity values compared to other heuristics and has the highest modularity values in all topologies. Although ILP_subopt might have a higher similarity value than CRO, the servers are not evenly distributed between clusters and could lead to two servers inside some clusters.
- Cloud providers could choose to perform clustering using ILP_subopt solution if clusters with high similarity values are needed, which could be the case in some scenarios.
- Density shows that CRO is better than the two flavors of the Game Theory approach only in random topologies but not in the modular one, where E-GT has a higher value. Although CRO has less density value than ILP_subopt solution in all kinds of topologies, in most cases, it can be preferable to be chosen by the cloud provider to have clusters that are almost even number of servers inside them.
- ILP is better for small-scale networking topologies. However, CRO and Game Theory are better choices for large-scale networks because they exhibit reasonable clustering runtime and a good tradeoff between similarity and modularity.
- Overall, our proposed approaches exhibit, most of the time, better performances compared to *K*-Means and DBSCAN in terms of similarity, modularity, and density but with an acceptable additional runtime.

9 Conclusion

This paper presents VALKYRIE as a suite of solutions for clustering and partitioning large attributed graphs for virtualized and non-virtualized environments. It is meant to help decision-makers eliminate scalability and computational time burdens when deploying their services in cases where each end user's requirements could be completely different from those of others. First, the system architecture is

defined, the problem is formalized, and the system model is presented. The problem was formulated using an Integer Linear Program to get the optimal solution for small-scale sizes. A chemical reaction-based meta-heuristic and a game theory-based approach were proposed to handle the scalability issues in the mathematical program. Our approaches jointly consider the nodes and the network attributes by optimizing the similarity and modularity as cost functions. Experiments with different infrastructures have shown that our solutions achieve reasonable clustering time concerning their size and can find a good trade-off between density, modularity, and similarity as a set of cost functions and being outliers-free. However, our proposed approaches were also compared with the two well-known clustering techniques. Namely, DBSCAN and *K*-Means, and results have revealed a better performance in both topologies. Finally, given their low computational complexities, our techniques are viable solutions to be integrated into orchestration systems following the NFV MANO framework.

Acknowledgements This work has been supported by Ericsson Canada and the Natural Sciences and Engineering Research Council of Canada (NSERC). The authors would like to thank Mohssine Arrouch, a Master's student at the School of Superior Technology (ETS), University of Quebec (Canada), for providing the experimental results using DBSCAN and *K*-Means.

Funding This work has been supported by Ericsson Canada and the Natural Sciences and Engineering Research Council of Canada (NSERC).

Data availability Data sharing not applicable to this article.

Declarations

Conflict of interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

References

1. Bothorel C, Cruz JD, Magnani M, Micenkova B (2015) Clustering attributed graphs: models, measures and methods. *Netw Sci* 3(3):408–444
2. Baroni A, Conte A, Patrignani M, Ruggieri S (2017) Efficiently clustering very large attributed graphs. In 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), IEEE, pp 369–376
3. Zhou Y, Cheng H, Yu JX (2009) Graph clustering based on structural/attribute similarities. *Proc VLDB Endow* 2(1):718–729
4. Cheng H, Zhou Y, Yu JX (2011) Clustering large attributed graphs: a balance between structural and attribute similarities. *ACM Trans Knowl Discov Data (TKDD)* 5(2):1–33
5. Clauset A, Newman ME, Moore C (2004) Finding community structure in very large networks. *Phys Rev E* 70(6):066111
6. https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf
7. Lam A, Li VO (2012) Chemical reaction optimization: a tutorial. *Memet Comput* 4(1):3–17
8. <https://www.gurobi.com>
9. Irving RW (1985) An efficient algorithm for the “stable roommates” problem. *J Algorithm* 6(4):577–595

10. Chen CC, Du YL, Chen SJ, Wang WJ (2018) Partitioning and placing virtual machine clusters on cloud environment. In: 2018 1st International Cognitive Cities Conference (IC3), IEEE, pp 268–270
11. Chen SJ, Chen CC, Lu HL, Wang WJ (2017) Efficient resource provisioning for virtual clusters on the cloud. *Int J Serv Technol Manag* 23(1–2):52–63
12. Sefraoui O, Aissaoui M, Eleuldj M (2012) OpenStack: toward an open-source solution for cloud computing. *Int J Comput Appl* 55(3):38–42
13. Jackson K, Bunch C, Sigler E (2015) OpenStack cloud computing cookbook. Packt Publishing Ltd
14. Chavan V, Kaveri PR (2014) Clustered virtual machines for higher availability of resources with improved scalability in cloud computing. In: 2014 First International Conference on Networks & Soft Computing (ICNSC2014), IEEE, pp 221–225
15. Pongsakorn U, Uthayopas P, Ichikawa K, Date S, Abe H (2013) An implementation of a multi-site virtual cluster cloud. In: The 2013 10th International Joint Conference on Computer Science and Software Engineering (JCSSE), IEEE, pp 155–159
16. Abdelsalam M, Krishnan R, Sandhu R (2017) Clustering-based IaaS cloud monitoring. In: 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), IEEE, pp 672–679
17. Wahab OA, Kara N, Edstrom C, Lemieux Y (2019) MAPLE: a machine learning approach for efficient placement and adjustment of virtual network functions. *J Netw Comput Appl* 142:37–50
18. Liu S, Li Z (2017) A modified genetic algorithm for community detection in complex networks. In: 2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), pp 1–3
19. Jami V, Reddy GRM (2016) A hybrid community detection based on evolutionary algorithms in social networks. In: 2016 IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS), IEEE, pp 1–6
20. Zachary WW (1977) An information flow model for conflict and fission in small groups. *J Anthropol Res* 33(4):452–473
21. Aylani A, Goyal N (2017) Community detection in social network based on users social activities. In: 2017 international Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), IEEE, pp 625–628
22. Baroni A, Conte A, Patrignani M, Ruggieri S (2017) Efficiently clustering very large attributed graphs. In: 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), IEEE, pp 369–376
23. Zhou Y, Cheng H, Yu JX (2009) Graph clustering based on structural/attribute similarities. *Proc VLDB Endow* 2(1):718–729
24. Shishavan ST, Gharehchopogh FS (2022) An improved cuckoo search optimization algorithm with genetic algorithm for community detection in complex networks. *Multimed Tools Appl*:1–27
25. Combe D et al (2020) Attributed networks partitioning based on modularity optimization. *Adv Data Sci: Symb Complex Netw Data* 4:169–185
26. Imtiaz ZB, Manzoor A, ul Islam S, Judge MA, Choo KKR, Rodrigues JJ (2021) Discovering communities from disjoint complex networks using multi-layer ant colony optimization. *Fut Gener Comput Syst* 115:659–670
27. Chunaev P (2020) Community detection in node-attributed social networks: a survey. *Comput Sci Rev* 37:100286
28. Chai Z, Liang S (2020) A node-priority based large-scale overlapping community detection using evolutionary multi-objective optimization. *Evolut Intell* 13(1):59–68
29. Sahni S, Gonzalez T (1976) P-complete approximation problems. *J ACM (JACM)* 23(3):555–565
30. El Mensoum I, Wahab OA, Kara N, Edstrom C (2020) MuSC: a multi-stage service chains embedding approach. *J Netw Comput Appl* 159:102593
31. Vassilvitskii S, Arthur D (2006) k-means++: the advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp 1027–1035
32. Cormen TH, Leiserson CE, Rivest RL, Stein C (2009) Introduction to Algorithms, MIT Press and McGraw-Hill
33. <https://networkx.github.io>
34. <https://www.gurobi.com/documentation/8.1/refman/parameters.html>
35. Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: *kdd*, Vol. 96, No. 34, pp 226–231

36. Bishop CM, Nasrabadi NM (2006) Pattern recognition and machine learning (Vol. 4, No. 4, p 738). New York: Springer

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Authors and Affiliations

Imane El Mansoum¹ · Laaziz Lahlou¹  · Fawaz A. Khasawneh¹ · Nadjia Kara¹ · Claes Edstrom²

✉ Imane El Mansoum
imane.el-mensoum.1@ens.etsmtl.ca

✉ Laaziz Lahlou
laaziz.lahlou.1@etsmtl.net

✉ Fawaz A. Khasawneh
fkhasawn@nuct.edu.jo

✉ Nadjia Kara
nadjia.kara@etsmtl.ca

¹ Department of Software Engineering and IT, École de Technologie Supérieure, University of Quebec, Montreal, Canada

² Ericsson Canada, Mississauga, Canada