

Metadata of the article that will be visualized in OnlineFirst

ArticleTitle	Placement Optimization of Virtual Network Functions in a Cloud Computing Environment	
Article Sub-Title		
Article CopyRight	Springer Science+Business Media, LLC, part of Springer Nature (This will be the copyright line in the final PDF)	
Journal Name	Journal of Network and Systems Management	
Corresponding Author	FamilyName	Said
	Particle	
	Given Name	Imad Eddine
	Suffix	
	Division	Research Unit LaMOS, Faculty of Exact Sciences
	Organization	University of Bejaia
	Address	06000, Béjaïa, Algeria
	Phone	
	Fax	
	Email	imadeddine.said@univ-bejaia.dz
	URL	
	ORCID	
Author	FamilyName	Sayad
	Particle	
	Given Name	Lamri
	Suffix	
	Division	Laboratory of Informatics and its Applications of M'sila (LIAM), Faculty of Mathematics and Computer Science
	Organization	University of M'sila
	Address	28000, M'sila, M'sila, Algeria
	Phone	
	Fax	
	Email	lamri.sayad@univ-msila.dz
	URL	
	ORCID	
Author	FamilyName	Aissani
	Particle	
	Given Name	Djamil
	Suffix	
	Division	Research Unit LaMOS, Faculty of Exact Sciences
	Organization	University of Bejaia
	Address	06000, Béjaïa, Algeria
	Phone	
	Fax	
	Email	djamil.aissani@univ-bejaia.dz
	URL	
	ORCID	
Schedule	Received	30 May 2023
	Revised	6 Nov 2023
	Accepted	19 Feb 2024
Abstract	<p>The use of Network Function Virtualization is constantly increasing in Cloud environments, especially for next-generation networks such as 5G. In this context, the definition of a deployment scheme defining for each Virtual Network Function (VNF) the appropriate server in order to meet the quality of service requirements. This problem is known in the literature as virtual network function placement. However, proper deployment of VNFs on servers can minimize the number of servers used, but may increase service latency. In this article, we propose a multi-objective integer linear programming model to solve the problem of network function placement. The objective is to find the best compromise between minimizing end-to-end total latency for users and reducing the number of servers used, while ensuring that the maximum number of VNFs is connected in the network. Our proposal to solve the NP-hard problem involves developing an algorithm based on the Particle Swarm Optimization metaheuristic to obtain a polynomial time resolution. By performing tests on a simple VNF deployment problem, we validated the relevance of our optimization model and demonstrated the effectiveness of our algorithm. The results obtained showed that our method provides feasible solutions very close to the exact optimal solutions.</p>	
Keywords (separated by '-')	Cloud - Virtualization - Virtual Network function placement - Multi-objective optimization - Metaheuristics	



Placement Optimization of Virtual Network Functions in a Cloud Computing Environment

Imad Eddine Said¹ · Lamri Sayad² · Djamil Aissani¹

Received: 30 May 2023 / Revised: 6 November 2023 / Accepted: 19 February 2024
 © Springer Science+Business Media, LLC, part of Springer Nature 2024

Abstract

The use of Network Function Virtualization is constantly increasing in Cloud environments, especially for next-generation networks such as 5G. In this context, the definition of a deployment scheme defining for each Virtual Network Function (VNF) the appropriate server in order to meet the quality of service requirements. This problem is known in the literature as virtual network function placement. However, proper deployment of VNFs on servers can minimize the number of servers used, but may increase service latency. In this article, we propose a multi-objective integer linear programming model to solve the problem of network function placement. The objective is to find the best compromise between minimizing end-to-end total latency for users and reducing the number of servers used, while ensuring that the maximum number of VNFs is connected in the network. Our proposal to solve the NP-hard problem involves developing an algorithm based on the Particle Swarm Optimization metaheuristic to obtain a polynomial time resolution. By performing tests on a simple VNF deployment problem, we validated the relevance of our optimization model and demonstrated the effectiveness of our algorithm. The results obtained showed that our method provides feasible solutions very close to the exact optimal solutions.

Keywords Cloud · Virtualization · Virtual Network function placement · Multi-objective optimization · Metaheuristics

1 Introduction

Since its inception, the Cloud Computing model has proven to be of crucial and major importance in academic and industrial settings, as well as in the field of computing in general [1–3]. In particular, it offers three primary service categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a

Lamri Sayad and Djamil Aissani have contributed equally to this work.

Extended author information available on the last page of the article

Service (SaaS). IaaS involves allocating computing resources like storage and processing power. PaaS provides web application development environments, such as Google App Engine and Microsoft Azure. SaaS refers to software services provided over the Internet by network service providers, accessible to end users upon payment. Cloud Computing offers significant benefits such as cost reduction, increased flexibility for organizations, and more efficient collaboration between academic researchers and industry professionals. For these reasons, it has become an essential element in academic and industrial settings [4]. Cloud Computing provides virtualized services. Virtualization is one of the key concepts that characterize Cloud environments, where virtual machines are created to run applications on the Cloud infrastructure. The evolution of this technology, as well as the variety of domains where it applies, has led to the virtualization of network functions and the emergence of Virtual Network Functions (VNF). These developments are driven by the increasing needs of next-generation networks, such as those of 5G [5–7].

The Network Function Virtualization (NFV) aims to dematerialize traditional network functions by implementing them as programs running on machines, called virtual network functions, such as firewalls, NATs, switches, etc.. It offers great flexibility related to the dynamic aspect of their deployment and independence from the service location, allowing for the reduction of investment (CAPEX) and operation (OPEX) expenses [8, 9]. With virtualized network functions, customers no longer need to manage hardware environments, purchase physical equipment, and bear the cost of its maintenance. They can move or instantiate these functions to different locations in the network with unparalleled flexibility, unlike traditional network functions. Although dedicated equipment, often more expensive, is often faster and more efficient than virtualized instances [10], VNFs offer many advantages in terms of flexibility and cost reduction. Furthermore, to deploy a Service Function Chain (SFC) forming a specific service chain, traditional network functions must be wired together in sequence, making them difficult to configure, modify, uninstall, and upgrade. With NFV, a chain of service functions is formed by an ordered sequence of VNFs. These NFV-based SFCs facilitate network service delivery and accelerate service innovation [8].

VNFs can be hosted in operator network nodes or in end-user premises. Network operators can also host their VNFs within their own data centers. A data center is a collection of computing elements such as servers, network and telecommunications equipment, data storage devices, as well as control and security systems, all dedicated to performing large-scale and data-intensive operations [11, 12]. Large online service providers such as Google, Microsoft, and Amazon use data centers to facilitate the management, maintenance, and security of their information systems, and to offer cloud computing services. However, to ensure reliability and meet user requirements, computing equipment such as servers, cooling systems, and network equipment consume a significant amount of energy in a data center. The distribution of energy consumption within different types of data center equipment varies depending on the models and categories of equipment. In general, it is observed that 40% of energy consumption is due to the cooling system, while computer servers represent about 44% of total consumption. Servers are particularly energy-intensive, with an average power of 250W per server [13]. These figures emphasize the importance

of energy management in data centers, particularly to improve the sustainability of these key facilities for online service providers and businesses. Furthermore, these significant resource consumption levels are always accompanied by carbon dioxide emissions, which pose a potential risk for climate change in the near future. This situation is increasingly concerning service providers and governments. It's worth noting that at the recent United Nations Climate Change Conference in Glasgow (COP26), it was clearly highlighted that global efforts to reduce carbon emissions are still not meeting the vital targets to ensure a sustainable climate and prevent a planetary warming catastrophe. Furthermore, comprehensive analyses of 5000 servers' data show a utilization rate of only 10–15% while machines needlessly consume up to 70% of the energy in data centers [14]. Therefore, inadequate resource management, such as deploying virtualized network functions on servers, can result in excessive use and significant waste of these resources. For all the aforementioned reasons, one of our research objectives is to focus on the efficient allocation of VNFs to hosting servers, in order to minimize the number of servers used in Data-Centers. It should be noted that optimizing server utilization also reduces energy consumption, cooling costs, and required surface area [15].

Telecommunications service providers are facing increasingly demanding innovative network service standards. These standards include growing demands for advanced cloud applications for new mobile customers, which are data-intensive and highly sensitive to latency. In order to fulfill these demands, a high-speed, low-latency network is essential. Various applications, such as high-definition video streaming and remote surgery, depend on such networks to deliver seamless and responsive user experiences. To meet these needs, Cloud Computing and NFV technologies can be interconnected with Multi-Access Edge Computing technology to support these new use cases of the Internet of the future. The MEC standard reduces end-to-end latency between vNFs and end-users, as well as response time and unnecessary use of the central network. This is done by placing VNFs close to end-users, at the MEC infrastructure level rather than at the provider's internal NFV infrastructure level. In addition, careful management of the latency of virtualized network functions in servers can allow for optimal allocation of VNFs to achieve an optimal total system latency [16, 17]. Thus, our second objective in this study is to determine the optimal distribution of virtualized network functions on hosting servers based on the expected total network latency.

In this article, we address the problem of placing VNFs in a Cloud environment from the perspective of a multicriteria decision problem. The goal is to simultaneously minimize the number of used servers and the total network latency, while maximizing the number of VNFs that can be connected to the hosting servers. To solve this problem, we propose a multi-objective integer linear programming model. We chose to consider a scenario similar to the one described by P. Pezaros and R. Cziva in their article [17]. The mathematical formulation of this problem is NP-hard, which requires the use of heuristic algorithms to obtain satisfactory solutions in polynomial time. To achieve this objective, we introduce an algorithm based on Particle Swarm Optimization (PSO) to obtain solutions adhering to the concept of lexicographic optimality. After validating our optimization model, we demonstrated the effectiveness of our optimization algorithm by applying it to a simple VNF

deployment. The results obtained are significantly close to the exact optimal solutions provided by the proposed model.

The rest of this paper is structured as follows: Sect. 2 discusses related work. The problem formulation and definition are presented in Sect. 3. The proposed optimization algorithm is described in Sect. 4. Experimental results obtained for a simple VNF deployment are discussed in Sect. 5. Finally, Sect. 6 concludes the article and presents future work directions.

2 Related Work

The placement of virtualized network functions and their benefits has generated great interest among researchers and service providers worldwide, both in scientific research literature and in the business world. This problem has been studied from different angles to best exploit the emerging virtual efficiency of network functions in different areas. Studies differ from each other in terms of system modeling, targeted improvement criteria, and the efficiency of proposed solutions.

In this context, some research has proposed to move the processing and analysis of data from virtualized network functions to the network edge, closer to where data is generated and consumed by network users, in order to improve network traffic management and meet the requirements of next-generation network standards that require low-latency services. Studies have been conducted on this subject by researchers such as [16–20]. In this regard, Cziva and Pezaros were the first to study the impact of latency between users and virtualized network functions at the network edge [16, 17]. In their first study, published in [17], they examined the optimal placement of virtualized network functions based on end-to-end latency between all users and their associated network functions, using an integer linear programming method. In order to improve the placement of virtualized network functions based on network-wide latency fluctuations, the authors introduced a dynamic planner based on the theory of optimal stopping [16]. They presented an ILP model to determine the minimum latency of VNFs in next-generation edge networks, taking into account latency variations. However, their initial model did not guarantee the optimal connection of all virtualized network functions to hosting devices, due to a constraint that caused modeling failures in some deployment scenarios. This limitation was discussed by Ghai et al. in [21, 22], where they proposed a modification of the problem formulation defined by Cziva and Pezaros. This modification relaxes the allocation constraint that requires each virtualized network function to be connected to a hosting device exactly once. The new modified model aims to minimize the total network latency and was solved using another ILP model that calculates the maximum number of virtualized network functions that can be connected in the system. To solve the problem in polynomial time, the authors developed a heuristic using a stable matching-based algorithm for the placement of VNFs on hosting devices based on their priorities established from latency. In [22], the same authors presented a local search algorithmic technique aimed at improving the efficiency of their optimization algorithm.

VNFs can be interconnected with each other and with associated hosting servers and ordered in sequence to provide a specific service. This is known as Service Function Chaining. Furthermore, correct chaining of VNFs can optimize resource consumption such as bandwidth [23]. In this article, the authors use integer linear programming to study the placement of the VNF service chain and traffic flow routing. The developed mathematical model minimizes resource consumption using their proposed objective function that calculates the total bandwidth consumed by all requested flows. Unlike previous works in this article, the authors take into account the number of network nodes capable of hosting VNFs and analyze different service chaining deployment strategies to update the service provider infrastructure with required computing resources. While other research works, such as [10], focus on NFV resource allocation, which concentrates on allocating virtual machines (VMs) and services in requested service chains. Their model, called VNF-P (Virtual network function placement), aims to minimize the number of used servers. A number of works deal with VNF placement to improve several critical criteria. For example, in [24], the authors presented important work for optimal placement of SFCs in a network cloud infrastructure that can be extended using MEC infrastructure. They proposed a mixed-integer programming formulation whose objective is to jointly minimize overall deployment costs and system latency, which takes into account several types of communication delays. Thus, they used two algorithms for its resolution: the Branch-and-Cut search algorithm of the CPLEX solver and a proposed heuristic based on taboo search algorithm. Another very recent work dealing with the problem in the context of multicast-oriented virtual network function placement (MVNFP) [25] for minimizing latency and processing resource consumption, while respecting bandwidth requirements. In [26], the authors of this article studied the problem in the form of a multi-objective mixed-integer programming problem. The first objective of the proposed model is to minimize the overall deployment cost of SFCs, which takes into account the costs of basic resources, physical node costs, and communication costs required to execute a service chain. The second objective of the proposed model is to minimize the total network latency, which takes into account queuing delay, processing and virtualization delay, and propagation delay. For its resolution, the authors proposed two metaheuristic-based algorithms such as the genetic algorithm and the bee colony algorithm. Whose solution optimality is in the Pareto sense. In [27], the authors propose a modeling of the VNF placement problem to minimize deployment cost. Their study shows that effective VNF placement can involve an increase in network reliability and performance and a decrease in operating cost. And many other studies that have paid considerable attention to the problem of virtualized network function placement in a cloud environment for different optimization aspects.

In this article, we consider two often conflicting objective functions: minimizing the number of servers used and minimizing the total network latency. Generally, it is not possible to find a feasible solution that optimizes both of these criteria simultaneously. Therefore, we formulate the problem of placing virtualized network functions as a multicriteria decision problem, where the objective is to determine the best compromise solution between these two objective functions according to the

preferences of service providers, while maximizing the number of VNFs that can be connected in the system.

We improve and generalize the models presented in previous work [17, 22] by combining them into a single model that covers all the important critical cases of the problem. This model is based on multi-objective integer linear programming and ensures the availability of VNFs, satisfies quality of service (QoS) by minimizing total end-to-end latency, and optimizes resources by minimizing the number of servers used. Therefore, it is a tri-objective model, offering an improved approach to solving this complex problem.

To solve our model, we have developed an optimization algorithm based on the PSO metaheuristic, enhanced by an initialization heuristic. Additionally, we have implemented a strategy specifically tailored to our study's problem, enabling us to efficiently explore the solution space. Our algorithm uses the principle of the lexicographic method to take into account the objective functions of the model. In this way, the problem of placing virtualized network functions is correctly formulated with respect to the reality of the decision problem.

3 System Model and Problem Definition

As shown in Sect. 1, the high demands for VNF services and infrastructure in cloud computing environments, despite their many benefits, can have undesirable consequences such as increased energy consumption in data centers, the number of servers used and their maintenance costs, as well as CO₂ emissions caused by the increased number of servers and high energy consumption for cooling [15, 28]. On the other hand, effective management of these services, such as reducing the number of servers used, plays a major role in reducing the aforementioned undesirable consequences [15, 28]. Appropriate and efficient assignment of VNFs to servers can significantly reduce the number of hosting servers used. However, reducing the number of servers used can lead to overutilization of server resources, which can have a negative effect on total network latency [29]. It should be noted that one of the main causes of network latency is the distance between the user and the servers that respond to it [17]. To provide good quality of service, reducing end-to-end latency and reasonable response times for end users are very important factors for network service providers (NSP) [16]. Therefore, it is important to find the best compromise solution between reducing network latency and the number of servers, based on the priorities of network service providers and service requirements.

In the following, we focus on proposing a problem modeling to provide the best compromise solution between the above two conflicting criteria while ensuring a maximum number of VNFs can be connected in the system using multi-objective integer linear programming. To our knowledge, none of the previous work has studied this problem as a multi-objective programming problem, to simultaneously optimize the number of servers used, the total network latency, and maximize the number of VNFs that can be connected in the system.

We considered a similar problem scenario described in [17, 22] to make their models more efficient and more general. So, a system of n VNFs must be

connected to h hosting servers where n is the number of VNFs and h is the number of servers. The set of VNFs is represented by $N = \{n_1, n_2, n_3, \dots, n_i\}$ and $H = \{h_1, h_2, h_3, \dots, h_j\}$ as the set of servers. Each server h_j has its own capacity (C_j), and similarly, a VNF n_i has requirements (R_i) such as CPU, memory, IO, etc. (ML_i) is the maximum latency value that a VNF can handle. A latency matrix I is calculated by the location of users, servers, and network topology, such that each element I_{ij} of the matrix gives the latency between the user of VNF n_i in the case where the VNF is located at h_j . Additionally, let U be the set of all users. Each u_i has a set of assigned VNFs $\{u_i; u_1; \dots\}$.

Taking into account the above data, our objective is to find the optimal assignment of all VNFs in set N to servers in set H , taking into account the total network latency expected by all users to access their VNFs, while minimizing the number of servers used and maximizing the number of VNFs connected to servers. To do this, we consider binary decision variables x_{ij} and y_j which denote, respectively, the allocation of VNFs to servers and whether the server is used. Such that:

$$x_{ij} = \begin{cases} 1 & \text{if } n_i \text{ is allocated to } h_j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$y_j = \begin{cases} 1 & \text{if the server } h_j \text{ is used} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

So the expected latency of all users to their VNFs is formulated by:

$$Z_1 = \sum_{n_i \in N} \sum_{h_j \in H} I_{ij} x_{ij} \quad (3)$$

And the number of used servers is formulated by:

$$Z_2 = \sum_{h_j \in H} y_j \quad (4)$$

Finally, the last function representing the number of VNFs that can be connected to servers and that we aim to maximize is as follows:

$$Z_3 = \sum_{n_i \in N} \sum_{h_j \in H} x_{ij} \quad (5)$$

The above objectives must be subject to certain constraints on resource capacities and the degree of service satisfaction provided to customers. Such as:

$$\sum_{n_i \in N} R_i x_{ij} \leq C_j \quad \forall h_j \in H \quad (6)$$

This constraint ensures that VNFs are placed on servers with sufficient capacity. By multiplying the second term of this constraint by the variable y_j , we get:

$$\sum_{n_i \in N} R_i \cdot x_{ij} \leq C_j \cdot y_j \quad \forall h_j \in H \quad (7)$$

In this way, if the server h_j has not yet been selected, we cannot allocate VNFs to this server. We still have two types of constraints, which are as follows:

$$\sum_{h_i \in H} I_{ij} \cdot x_{ij} \leq ML_i \quad \forall n_i \in N \quad (8)$$

$$\sum_{h_i \in H} x_{ij} \leq 1 \quad \forall n_i \in N \quad (9)$$

In which constraint 8 ensures that latency-sensitive VNFs are placed while not violating their users' maximum latency requirement. The last constraint 9 ensures that each VNF cannot be allocated to two different servers at the same time. The formulation proposed in [17] has a drawback that fails the model in certain problem scenarios, which we will not discuss here as it has been clearly indicated in [22], and the authors have demonstrated this drawback through an example.

The following example will help better understand the improvement that our multi-objective linear model can bring to our placement problem. We consider a system consisting of five functions that can connect to three servers, as specified in the dataset instance.

$$\mathbf{I} = \begin{pmatrix} 20 & 91 & 103 \\ 89 & 36 & 94 \\ 56 & 79 & 90 \\ 105 & 16 & 16 \\ 118 & 22 & 22 \end{pmatrix}$$

Server	Capacity
1	6
2	6
3	6

VNF	Requirement
1	4
2	3
3	2
4	2
5	1

The optimal assignment in terms of total system latency in this scenario is represented in the following Fig. 1

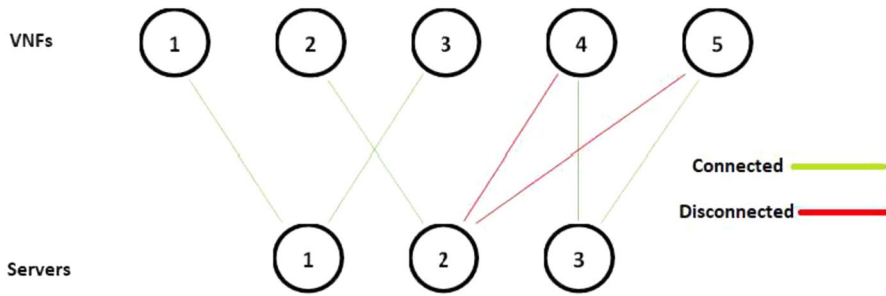


Fig. 1 Example of an optimal assignment

The solution proposed by our model for this problem scenario, which consists of releasing the third server and assigning the fourth and fifth VNFs to the second server, is indeed a better decision, as it maintains the same overall latency value while reducing the number of used servers to 2 instead of 3. This solution is ideal, especially for larger-scale scenario, involving more than 5000 VNFs and 2000 servers. It should be noted that the priorities of the criteria may vary depending on the needs and preferences of network service providers, with some flexibility in choosing between latency reduction and server reduction.

AQ3

4 Algorithm Design for Resolutions

In this section, we present the main concepts of the optimization algorithm that we used to solve our proposed model. Due to the complexity of our model, we chose to build our algorithm based on metaheuristics, specifically the particle swarm optimization. PSO is one of the major paradigms of swarm intelligence (SI). Its guiding idea is to simulate the dynamics of a swarm of animals evaluating their environment without the help of a designated leader, like a swarm of bees, a flock of birds, or a school of fish. Each particle of the swarm has a memory of the best sites it has visited, as well as all the particles of the swarm, communicating with each other during their search for food. The choice of the PSO algorithm is motivated by its ability to effectively handle complex optimization problem solving. Thus, the approach based on collective swarm intelligence and solutions in population makes PSO an attractive choice for a complex problem such as optimizing the placement of VNFs in cloud environments. For more details on the PSO algorithm, we invite the reader to consult the following references: [30, 31].

In the context of our optimization model, the feasible solutions do not take the form of real vectors, but rather of binary matrices with n rows and h columns. Here, n corresponds to the total number of VNFs in the network, while h represents the number of available servers. Thus, to integrate this type of data into our PSO algorithm, it was necessary to adjust the equations of the model so that the swarm particles could adequately represent these binary matrices. To do this, each particle p of the swarm is modeled by its position vector $\vec{x}_p = (x_{p1}, x_{p2}, \dots, x_{pn})$ of n components,

where each component x_{pi} represents an assignment of a VNF i to server x_{pi} , with $x_{pi} \in \{0, 1, 2, \dots, h\}$ and $i \in 1, 2, \dots, n$. $x_{pi} = 0$ means that VNF i is not assigned to any server. Thus, the positions of the particles \vec{x}_i are appropriate assignments of VNFs to servers in the system. For each iteration t of the optimization, particle p moves from one position to another by applying a velocity $\vec{v}_p = (v_{p1}, v_{p2}, \dots, v_{pn})$ to the current position using the following two equations:

$$v_{p,i}^{(t+1)} = w \cdot v_{p,i}^t + c_1 \cdot r_{1(i,j)}^t \cdot (P_{\text{best}_{p,i}}^t - x_{p,i}^t) + c_2 \cdot r_{2(p,d)}^t \cdot (G_{\text{best}_i}^t - x_{p,i}^t) \quad (10)$$

$$x_{p,i}^{(t+1)} = x_{p,i}^t + v_{p,i}^t \quad (11)$$

Where w is the constant inertia factor. c_1 and c_2 are two constant parameters of the acceleration affecting the cognitive and social behavior of the particles. r_1 and r_2 are two random numbers generated in $[0, 1]$ at each iteration t for each dimension j .

The velocity of the particle is influenced by its current position as well as the best positions it has recorded in its personal and collective memory. Therefore, the direction of the particle's movement is attracted by three factors:

1. An inertia component: the particle is attracted to its personal direction.
2. A cognitive component: the particle is attracted to its own personal experience, tending to return to the best position it has already passed.
3. A social component: the particle is attracted to the collective experience of the swarm, tending to move towards the best position found by its neighbors.

It is through this movement strategy that each particle can move from one position to another while evaluating the quality of each site using the objective functions to be optimized. Thus, each particle can build its personal experience as well as the social experience of the swarm using the following equations:

$$P_{\text{best}_p}^{t+1} = \begin{cases} x_p^{t+1}, & \text{if } Z(x_p^{t+1}) \leq P_{\text{best}_p}^t \\ P_{\text{best}_p}^t, & \text{otherwise} \end{cases} \quad (12)$$

$$G_{\text{best}} = \arg \min \{Z(P_{\text{best}_p}); p = 1, \dots, P\} \quad (13)$$

where P is the population size.

We have set a maximum velocity for particle movements in the algorithm process. This decision was made after conducting tests on our problem and considering in-depth studies carried out by researchers such as [32]. Their work has highlighted that particle velocity can undergo significant oscillations, which can lead to their departure from the search space and thus cause algorithm divergence. To avoid this problem, we chose to limit the velocity components of each particle using a new parameter V_{max} . This approach improves the convergence of the algorithm and controls the explosion of the system.

On the other hand, restricting the velocity on each dimension i does not confine the values of x_{pi} to the interval $[V_{min}, V_{max}]$. It only limits the maximum distance a particle can travel in one iteration. We introduce a more suitable strategy for our placement problem, allowing the transition from one solution to another while remaining in the search space during the solution generation process. If a move causes a particle p to leave the search space on dimension i , we then search for a position with minimized end-to-end latency I_{ij} .

Therefore, an appropriate initialization of the starting positions of particles, rather than selecting them randomly, could improve the efficiency of the PSO algorithm and reduce the convergence time to the optimal solution. To do this, we have developed a heuristic for our resolution algorithm that initializes the starting positions of particles as follows:

- We arrange the servers we have in descending order of capacity. We do the same for the VNFs, arranging them according to their capacity requirements in descending order.
- Next, we fill the servers one by one, assigning the VNFs in their previous order to a randomly chosen server (with the server with the largest capacity as a special case for the first particle).
- If the selected VNF does not satisfy the model's constraints, we move on to the next VNF with less requirement.
- We finish when we can no longer assign VNFs to servers.

Note that this heuristic maximizes the number of connected VNFs to servers while minimizing the number of simultaneously used servers.

From the intermediate solutions obtained using this heuristic, the particles explore the search space and tend to converge towards the optimal solution of the optimization problem. Indeed, our criteria in the model are contradictory to each other, making the search for a global optimum difficult or even impossible. Unlike mono-objective problems where the goal is to find the global optimum for a single criterion, multi-objective problems aim to simultaneously optimize several criteria of the model. However, the global optimal solution of such a problem is generally non-existent, thus requiring the use of a decision-making framework to determine a solution. In this article, we used the definition of lexicographic optimality to solve our multicriteria problem.

4.1 Lexicographic Optimality

Lexicographic optimality definition implies that the optimal solution depends on the priority (or preference) order established by the decision-maker for the criteria. Consequently, the comparison of solution efficiency is carried out while respecting the order sequence assigned to the objectives. Denoting S as the set of feasible solutions for such an r -criteria optimization problem, lexicographic optimality is defined as follows [33]: A feasible solution $\hat{x} \in S$ is lexicographically optimal if

$Z(\hat{x}) <_{lex} Z(x), \forall x \in S$. Where $Z(\hat{x}) <_{lex} Z(x)$ means that: $\exists q \in \{1, 2, \dots, r\}$ such that for all $k \leq (q - 1)$, we have $Z(\hat{x}) = Z(x_k)$, and $Z(\hat{x}) < Z(x_q)$.

The lexicographic method is one of the main techniques for solving multi-criteria problems. Its principle consists of solving the optimization problem in a sequential way by optimizing the criteria one after the other according to a known order of preferences of the decision-maker. At each successive optimization, the already optimized criteria are maintained at their optimal values as constraints, thus ensuring the quality of the previously treated criteria when optimizing the next criterion. The procedure is repeated until all criteria have been treated. For further information, we invite the reader to consult the references [33, 34].

In the context of our research, this method has proven to be particularly effective. Its adaptation requires prior knowledge of the decision-maker's preference structure among network service providers. This is why we have carefully ranked the criteria based on their relevance. Our optimization algorithm is based on this multi-criteria problem-solving technique, which has been adapted to meet the specific needs of our study.

The structural adaptations can be summarized by the following Algorithm 1:

At line 24, when optimizing the criteria Z_{k^2} and Z_{k^3} , particle performances are updated based on constraint 12 and 13, as well as optimality constraints determined during the optimization of the previous criteria. These optimality constraints are based on the optimal values obtained previously. Therefore, when updating particle performances, improving higher-level criteria is considered a priority over lower-level criteria.

The Eqs. 12 and 13 of our algorithm are designed to minimize the objective function. However, our goal Z_3 requires maximization. To address this inconsistency, we have adopted a common approach of transforming the maximization of Z_3 into the minimization of $-Z_3$ while maintaining consistency among the objectives for multi-criteria resolution.

5 Evaluation of Results

To evaluate the performance of our optimization algorithm and demonstrate the effectiveness of our proposed solution, we conducted simulations of various scenarios of the VNF deployment problem. We implemented the algorithm in C++, where the experimental platform is a PC equipped with Windows 10, 64 bits with a processor of i3 1.9 GHz and a RAM of 12.00 GB.

Using the IBM CPLEX solver, we calculated the exact optimal solution for each criterion individually, according to the priority order set by the lexicographic optimality method. The comparison measure used to evaluate the proposed solutions was the average value of the solutions obtained during 100 runs (Avg), and we recorded the best values obtained among them. The model parameters for the simulations were randomly selected from specific real value intervals for each simulation, namely: server capacities and VNF capacity requirements were between 35 – 80 and 1 – 15, respectively. The maximum latency tolerated by a VNF and the latencies between VNFs and hosted servers were between 35 – 60 and 15 – 25, respectively.

Algorithm 1 System optimization for virtual network function placement (OSVNFP)

```

1: procedure OPTIMIZE THE CRITERIA ONE BY ONE IN THE ORDER DEFINED
2:    $T$  : Number of iterations;  $Z_k (1 \leq k \leq 3)$  : Objective functions ;  $v_{max} = h$  ;  $P$ :
   the population size
3:   initialize the population according to the previously mentioned heuristic.
4:   define the lexicographic order for the criteria  $Z_{k1}, Z_{k2}, Z_{k3}$ 
5:   evaluate the positions. For each particle  $p$ ,  $P_{best_p} = x_p$  and determine  $G_{best}$ 
   according to 13.
6:   Optimize the criteria one by one in the order defined above, leveraging the
   following structure:
7:   while the number of iterations is less than  $T$  do
8:     for all particle  $p$  on each dimension  $i$  do
9:       Update the velocity  $v_{pi}$  according to equation 10
10:      if  $v_{pi} \notin [-v_{max}, v_{max}]$  then
11:         $v_{pi}$  takes a random value in  $[-v_{max}, v_{max}]$ 
12:      end if
13:      Generate a new position  $x_{pi}$  according to equation 11
14:
15:      if  $x_{pi} \in [0, h]$  and satisfies constraints 7 and 8 then
16:         $x_{pi}$  takes the higher or lower integer part with low end-to-end
        latency, takes 0 if  $-0.5 < x_{pi} < 0.5$ 
17:      else ( $x_{pi} = j$ ) where  $j$  is the server with the lowest possible end-to-end
        latency and satisfies constraints 7 and 8 (In other words, the smallest element  $I_{ij}$ 
        in row  $i$  of matrix  $I$ ).
18:      end if
19:      if the displacement has not been made then
20:         $x_{pi} = 0$ .
21:      end if
22:    end for
23:    Evaluate the positions of the particles.
24:    Update  $P_{best}$  and  $G_{best}$  according to 12 and 13 using the objective function
    under current optimization
25:    Add the optimal value  $\hat{Z}_k$  found as constraints for the optimization of the
    lower-level criteria.
26:  end while
27:  Go to the optimization of the next criterion (Go to 6)
28:  Output: Results
29: end procedure

```

Thus, in order to better adapt the PSO algorithm parameters to our VNF placement problem, we adjusted the values of these parameters for each scenario. However, in this article, we present the results obtained using the standard configuration of the PSO algorithm ($w = 1, c_1 = c_2 = 2.0, r_1$ and r_2 are randomly drawn from $[0; 1]$) [35]. On the other hand, we have given special attention to the establishment of the priority order of criteria. Our approach is built on a deep understanding of the needs and priorities of network service providers, who are at the core of our approach. Our top priority, which is to maximize the number of connected VNFs in the system (represented by the objective function Z_3), is based on the imperative of ensuring

VNF availability. Any malfunction or service interruption in this area can have significant repercussions on user experience and network service continuity. Therefore, we have placed this priority at the top of our list, as it embodies the commitment to provide reliable network services. Next, we prioritized service quality by aiming to minimize end-to-end total latency in the network (represented by the objective function Z_1). High latency can lead to undesirable delays in network communications, which can impact the quality of the user experience. Meeting the needs for service quality thus became our second priority to meet customer expectations. Finally, we chose to optimize resource utilization by minimizing the number of servers used in the system (represented by the objective function Z_2). This priority, while crucial, was placed last because our analysis revealed that, in our specific context, ensuring VNF availability and service quality were major imperatives. Our choice of priority order, therefore, reflects a deep understanding of the needs and priorities of network service providers and aims to ensure a robust solution tailored to their operational environment. To evaluate the performance of our algorithm, we used the lexicographic optimality definition to obtain optimal solutions following this priority order. Furthermore, we varied the size of input data instances to create various scenarios and compared our results with exact optimal solutions calculated by the IBM CPLEX solver. A summary of executions for latency comparison is presented in Table 1 below for different VNF and server scenarios:

The results presented in Table 1 demonstrate that our algorithm provides solutions that are practically identical to the exact optimal solutions produced by the CPLEX solver in terms of latency. This is achieved by ensuring a maximum number of VNFs that can be connected in the system, as shown in the fourth column of Table 1 for the different deployment scenarios examined. Moreover, our optimization algorithm is not limited to optimizing the number of VNFs that can be connected and the total network latency. The system is also optimized by reducing the number of servers used for the network while respecting the priority order we defined in the previous paragraph. We achieve this by setting the optimal values of the priority criteria presented in Table 1 to optimize (minimize) the number of servers for the same data sets. Table 2 below presents the optimal solutions obtained using the lexicographic optimality method.

For example, our mathematical model proposes the optimal decision for the deployment scenario of 500 VNFs and 300 hosting servers, represented by the cost vector (7582; 56; 500). This decision consists of assigning 500 VNFs with a total latency of 7582 and using only 56 servers. We consider this an ideal solution for this scenario, while our optimization algorithm was able to achieve the exact optimal solution with 97 servers, maximizing the number of connected VNFs and minimizing latency.

We have reached the maximum performance limits of the CPLEX software in terms of providing accurate optimal solutions, considering the hardware constraints mentioned at the beginning of this section. As a result, our proposed solutions are significant as they take into account the reduction in the number of servers. We have further optimized the system to maximize resource availability, specifically servers, which is a crucial factor in reducing power consumption, cooling costs, equipment footprint, and CO2 emissions, as demonstrated in

Table 1 Comparison of results in terms of latency

VNFs	Servers	Population size	Number of iterations	Max number of VNFs	Optimal latency	Avg	Best solution
20	5	7	40	20	380	380.45	380
	10	7	40	20	322	322	322
	15	7	40	20	326	326	326
30	10	10	50	30	498	498	498
	15	10	50	30	480	480	480
	20	10	50	30	475	475	475
50	10	20	100	50	827	827.12	827
	15	20	100	50	798	798	798
	20	20	100	50	793	793	793
100	20	40	250	100	1570	1571.93	1571
	30	40	250	100	1544	1544	1544
	40	40	250	100	1520	1520	1520
200	30	50	300	200	3076	3105.23	3087
	40	50	300	200	3041	3047.37	3043
	50	50	300	200	3036	3037.13	3036
500	100	50	300	500	7507	7518.46	7515
	200	50	300	500	7501	7515.10	7501
	300	50	300	500	7482	7484.007	7482
1000	200	50	300	1000	15002	15011.13	15002
	300	50	300	1000	15000	15009.23	15000
	500	50	300	1000	21000	21004.6	21000
2000	300	50	300	2000	30003	30026.78	30005
	500	50	300	2000	30000	30013.24	30000

Sect. 3 of this article. It is worth noting that our solutions are obtained within a reasonable timeframe using our optimization algorithm, as shown in Table 3.

Table 4 presented below showcases the promising results obtained for over 3000 VNFs.

We have enriched our analysis by providing specific temporal details for each scenario evaluated. Table 5 shows the final solutions obtained by our OSVNF algorithm and their comparison with the optimal solution provided by the CPLEX model. It is shown that our OSVNF algorithm provides a significant reduction in time, ranging from 12% to 99% compared with the optimal solution. In terms of latency, our OSVNF approach shows a slight increase ranging from only 0.04% to 1% compared to the optimal solutions. Furthermore, the table provides a comparative analysis of the number of servers used in the system, revealing that the OSVNF solution leads to an increase of approximately 8% to 90% in the number of servers used in comparison to the optimal solutions.

Table 2 Comparisons of results in terms of the number of operated servers

VNFs	Servers	optimal	Avg	Best solution
20	5	5	5	5
	10	8	8	8
	15	10	10	10
30	10	9	9	9
	15	12	12	12
	20	10	11.97	12
50	10	10	10	10
	15	11	12.41	12
	20	16	17.98	17
100	20	20	20	20
	30	24	28.78	28
	40	30	36.70	35
200	30	30	30	30
	40	37	39.99	39
	50	41	46.89	45
500	100	64	93.70	88
	200	57	103.56	101
	300	56	107.21	97
1000	200	118	163.12	159
2000	300	246	293.71	289
	500	213	295.53	288

In summary, all the experimental results demonstrate the effectiveness of our algorithm, capable of generating solutions very close to the optimal solutions in a very short amount of time, while considering the maximization of the number of connected VNFs in the system, the minimization of total latency, and the reduction of the number of servers used - in that order. This speed of computation highlights the practicality and efficiency of the algorithm we have proposed.

6 Conclusion

In this article, we addressed the problem of optimizing the placement of VNFs in servers as a multi-objective assignment problem aimed at minimizing both the total network latency and the number of used servers while guaranteeing connectivity for the maximum number of VNFs. We extended a similar problem scenario described in references [17, 22], by developing a more general and reinforced model. Since the problem is NP-hard, we have developed an optimization algorithm based on the PSO metaheuristic to solve it. The final solutions provided by our algorithm are based on the definition of lexicographic optimality. Experimental results for different VNFs and server scenarios in a small-scale network show that our algorithm provides nearly identical results to exact optimal solutions.

Table 3 Working Time for the various configurations

VNFs	Servers	Time(second)
20	5	0.828
	10	0.818
	15	0.891
30	10	0.937
	15	0.969
	20	0.953
50	10	1.281
	15	1.234
	20	1.386
100	20	1.625
	30	1.500
	40	1.453
200	30	2.484
	40	2.437
	50	2.390
500	100	6.812
	200	9.484
	300	12.077
1000	200	18.358
	300	28.326
	500	39.778
2000	300	41.999
	500	75.073

Table 4 Promising results revealed for over 3000 VNFs

VNFs	Servers	Population	Iterations	Max number of VNF	Total latency	Servers used	Time(second)
3000	300	50	300	2566	53995	284	110.870
	500	50	300	3000	54007	431	111.911
5000	500	50	300	4475	89364	419	193.667
	1000	50	300	5000	89456	545	274.621

In the future developments of our work, we intend to apply the definition of Pareto optimality, which is based on the concepts of efficiency and non-dominance, to solve our multi-objective problem. To further improve our optimization algorithm, we plan to adapt it to large-scale networks by adjusting the parameters appropriately. We also plan to compare the results obtained by our algorithm with those of existing methods in the literature.

Table 5 Comparison of OSVNFP Algorithm Results with the Optimal Solution Obtained by CPLEX Solver

VNFs	Servers	Optimal (Time)	OSVNFP	% Decrease (Time)	% Increase (Latency)	% Increase (server)
200	30	2.87 s	2.484 s	12.02	0.95	0
	40	3.21 s	2.437 s	24.08	0.2	8.08
	50	3.56 s	2.390 s	32.86	0.04	14.37
500	100	6 m: 26.54 s	6.812 s	98.23	0.15	46.41
	200	18 m:22.36 s	9.484 s	99.13	0.18	81.68
	300	21 m:28.82s	12.077 s	99.06	0.03	91.44
1000	200	46 m:17.26s	18.358 s	99.33	0.06	38.14
	300	45 m:09.56s	28.326 s	98.95	0.06	39.03
	500	51 m:52.68s	48.778 s	98.43	0.2	45.06
2000	300	48 m:42.71s	41.999 s	98.56	0.07	19.39
	500	1 h:14 m:17.26s	75.073 s	98.31	0.04	38.75

Acknowledgements The authors would like to express their sincere gratitude to the reviewers for their valuable contributions in shaping this article through their constructive suggestions.

Author contributions Imadeddine Said Conceptualization, Methodology, Writing original draft, Project administration. Lamri Sayad Validation, review & editing, Supervision. Djamil Aissani Validation, review & editing, Supervision.

Declarations

Conflict of interest The authors declare that they have no Conflict of interest related to this publication.

References

1. Sunyaev, A.: Cloud Computing. Springer, Cham (2020)
2. Wang, B., Qi, Z., Ma, R., Guan, H., Vasilakos, A.V.: A survey on data center networking for cloud computing. *Comput. Netw.* **91**, 528–547 (2015). <https://doi.org/10.1016/j.comnet.2015.08.040>
3. Jennings, B., Stadler, R.: Resource management in clouds: survey and research challenges. *J. Netw. Syst. Manag.* **23**, 567–619 (2015). <https://doi.org/10.1007/s10922-014-9307-7>
4. Sadiku, M.N.O., Musa, S.M., Momoh, O.D.: Cloud computing: opportunities and challenges. *IEEE Potentials* **33**(1), 34–36 (2014). <https://doi.org/10.1109/MPOT.2013.2279684>
5. ETSI: Network function virtualisation white paper 1. SDN and OpenFlow World Congress,2012,Darmstadt, Germany
6. ETSI: Network functions virtualisation white paper 3. SDN and OpenFlow World Congress,2014,Dusseldorf, Germany
7. Santos, G.L., Bezerra, D.d.F., Rocha, É.d.S., Ferreira, L., Moreira, A.L.C., Gonçalves, G.E., Marquezini, M.V., Recse, Á., Mehta, A., Kelner, J., et al.: Service function chain placement in distributed scenarios: a systematic review. *J. Netw. Syst. Manag.* (2022) <https://doi.org/10.1007/s10922-021-09626-4>

8. Tao, X., Han, Y., Xu, X., Zhang, P., Leung, V.C.M.: Recent advances and future challenges for mobile network virtualization. *Sci. Chin. Inform. Sci.* **60**(4), 1 (2017). <https://doi.org/10.1007/s11432-017-9045-1>
9. Yi, B., Wang, X., Li, K., Das, S., Huang, M.: A comprehensive survey of network function virtualization. *Comput. Netw.* **133**, 212–262 (2018). <https://doi.org/10.1016/j.comnet.2018.01.021>
10. Moens, H., De Turck, F.: Vnf-p: A model for efficient placement of virtualized network functions. In: 10th International Conference on Network and Service Management (CNSM) and Workshop, pp. 418–423 (2014). IEEE
11. Barroso, L.A., Clidaras, J., Hözlze, U.: The datacenter as a computer: an introduction to the design of warehouse-scale machines. *Synth. Comput. Archit.* **8**(3), 1–154 (2013)
12. Amin, R., Hussain, M., Bilal, M.: In: Aujla, G.S., Garg, S., Kaur, K., Sikdar, B. (eds.) *Network Policies in Software Defined Internet of Everything*, pp. 79–96. Springer, Cham (2022)
13. Johnson, P., Marker, T.: Data centre energy efficiency product profile. Pitt & Sherry, report to equipment energy efficiency committee (E3) of The Australian Government Department of the Environment, Water, Heritage and the Arts (DEWHA) (2009)
14. Sinha, R., Purohit, N., Diwanji, H.: Power aware live migration for data centers in cloud using dynamic threshold. *Int. J. Comput. Technol. Appl.* **2**(6) (2011)
15. Safieddine, I.: Optimisation d'infrastructures de cloud computing sur des green datacenters. PhD thesis, Université Grenoble Alpes (2015)
16. Cziva, R., Anagnostopoulos, C., Pezaros, D.P.: Dynamic, latency-optimal VNF placement at the network edge. In: IEEE Infocom 2018-IEEE Conference on Computer Communications, pp. 693–701 (2018). IEEE
17. Cziva, R., Pezaros, D.P.: On the latency benefits of edge NFV. In: 2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), pp. 105–106 (2017). <https://doi.org/10.1109/ancs.2017.23>. IEEE
18. Hu, Y.C., Patel, M., Sabella, D., Sprecher, N., Young, V.: Mobile edge computing—a key technology towards 5G. ETSI White Pap. **11**(11), 1–16 (2015)
19. Cziva, R., Jouet, S., Pezaros, D.P.: Roaming edge vnfs using glasgow network functions. In: Proceedings of the 2016 ACM SIGCOMM Conference, pp. 601–602. Association for Computing Machinery, New York, NY, USA (2016). <https://doi.org/10.1145/2934872.2959067>
20. Cziva, R., Pezaros, D.P.: Container network functions: bringing NFV to the network edge. *IEEE Commun. Mag.* **55**(6), 24–31 (2017)
21. Ghai, K.S., Choudhury, S., Yassine, A.: A stable matching based algorithm to minimize the end-to-end latency of edge NFV. *Procedia Comput. Sci.* **151**, 377–384 (2019). <https://doi.org/10.1016/j.procs.2019.04.052>
22. Ghai, K.S., Choudhury, S., Yassine, A.: Efficient algorithms to minimize the end-to-end latency of edge network function virtualization. *J. Ambient. Intell. Humaniz. Comput.* **11**(10), 3963–3974 (2020). <https://doi.org/10.1007/s12652-019-01630-6>
23. Gupta, A., Habib, M.F., Mandal, U., Chowdhury, P., Tornatore, M., Mukherjee, B.: On service-chaining strategies using virtual network functions in operator networks. *Comput. Netw.* **133**, 1–16 (2018). <https://doi.org/10.1016/j.comnet.2018.01.028>
24. Leivadeas, A., Kesidis, G., Ibnkahla, M., Lambadaris, I.: VNF placement optimization at the edge and cloud. *Futur. Internet* **11**(3), 69 (2019). <https://doi.org/10.3390/fi11030069>
25. Wang, X., Xing, H., Zhan, D., Luo, S., Dai, P., Iqbal, M.A.: A two-stage approach for multicast-oriented virtual network function placement. *Appl. Soft Comput.* **112**, 107798 (2021). <https://doi.org/10.1016/j.asoc.2021.107798>
26. Khoshkholghi, M.A., Gokan Khan, M., Alizadeh Noghani, K., Taheri, J., Bhamare, D., Kassler, A., Xiang, Z., Deng, S., Yang, X.: Service function chain placement for joint cost and latency optimization. *Mobile Netw. Appl.* **25**, 2191–2205 (2020)
27. Cohen, R., Lewin-Eytan, L., Naor, J.S., Raz, D.: Near optimal placement of virtual network functions. In: 2015 IEEE Conference on Computer Communications (INFOCOM). IEEE (2015). <https://doi.org/10.1109/infocom.2015.7218511>
28. Bayati, L.: Data Centers Energy Optimization. PhD thesis, Paris Est (2019)
29. Leivadeas, A., Kesidis, G., Ibnkahla, M., Lambadaris, I.: VNF placement optimization at the edge and cloud. *Futur. Internet* **11**(3), 69 (2019). <https://doi.org/10.3390/fi11030069>
30. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-international Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995). IEEE

31. Shi, Y., Eberhart, R.C.: Empirical study of particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary computation-CEC99 (Cat. No. 99TH8406), vol. 3, pp. 1945–1950 (1999). <https://doi.org/10.1109/CEC.1999.785511>. IEEE
32. Eberhart, R., Simpson, P., Dobbins, R.: Computational Intelligence PC Tools. Academic Press Professional Inc, USA (1996)
33. Ehrgott, M.: Multicriteria Optimization. Springer, Berlin, Heidelberg (2005)
34. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston (1999)
35. Abdelbar, A.M., Abdelshahid, S.: Instinct-based pso with local search applied to satisfiability. In: 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541), vol. 3, pp. 2291–2295 (2004). IEEE

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

Springer Nature or its licensor (e.g. a society or other partner) 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

Imad Eddine Said¹ · Lamri Sayad² · Djamil Aissani¹

✉ Imad Eddine Said
imadeddine.said@univ-bejaia.dz

Lamri Sayad
lamri.sayad@univ-msila.dz

Djamil Aissani
djamil.aissani@univ-bejaia.dz

¹ Research Unit LaMOS, Faculty of Exact Sciences, University of Bejaia, 06000 Béjaïa, Algeria

² Laboratory of Informatics and its Applications of M'sila (LIAM), Faculty of Mathematics and Computer Science, University of M'sila, 28000 M'sila, M'sila, Algeria




Journal:	10922
Article:	9812

Author Query Form

Please ensure you fill out your response to the queries raised below and return this form along with your corrections

Dear Author

During the process of typesetting your article, the following queries have arisen. Please check your typeset proof carefully against the queries listed below and mark the necessary changes either directly on the proof/online grid or in the ‘Author’s response’ area provided below

Query	Details Required	A u t h o r ’ s Response
AQ1	Please provide author biography.	
AQ2	Please check and confirm all the author names and initials are correct.	
AQ3	Figure 3 has been changed to figure 1 kindly check and confirm its correct.	
AQ4	Please provide a complete detatails for the references [12, 14 and 15].	