Metaheuristics as Enablers for VNF Scheduling in the Network Slice Set Up Process

Anastasios-Stavros Charismiadis, Dimitris Tsolkas, Nikos Passas, Dionysis Xenakis, and Lazaros Merakos

***Abstract*—Network slicing refers to the capability of composing mobile networks by chaining a set of virtualised functions on top of shared infrastructures. In the research domain, special attention is paid on the problem of scheduling network slices, i.e., the challenge of managing efficiently computation resources when multiple network slices share the same infrastructure. So far, the rich toolset that has emerged from the studies on the spectrum resource management, as well as the rapid development of cloud computing, have provided the means for scheduling decisions in 5G networks. Capitalizing on the existing studies, we examine the potential of using metaheuristic algorithms for providing scheduling outputs that minimize the slice set up time. Performance evaluation results show that evolution-based approaches (e.g., a genetic algorithm) provide better overall performance than swarm-based ones (e.g., an ant colony optimiser). However, since the slice set-up process is a real-time process, the processing time that is consumed by the scheduler itself is an important evaluation factor, for which, the swarm-based approaches have an advantage.**

***Index Terms*—5G, ant colony optimisation, genetic algorithm, job-shop scheduling problem, metaheuristics, network slicing, service creation time, VNF.**

I. INTRODUCTION

THE fifth generation (5G) of mobile networks has brought a new era in telecommunications where new research challenges and business potentials have emerged. The key advances of 5G refer to i) the convergence of telecom sector with the cloud infrastructures, through the virtualisation of the network functions, and ii) the unprecedented performance gains, due to the evolution of the radio access network (RAN) part of the network. Already, a plethora of telecom operators around the globe [1], have integrated 5G compliant equipment in their infrastructures and they have set the scene for providing 5G-enabled services. The research community is also intensively active on 5G-related topics. For instance, 5G PPP projects [2] have provided a plethora of 5G experimentation infrastructures around Europe. Academia has also produced a great number of results in emerging 5G research fields, such as network slicing [3], [4], mobile network openness [5], resource sharing [6], [7], and (radio) access technologies engagement [8], [9].

The goals of 5G networks were set from ITU IMT 2020 [10], and they can be described briefly by the following higher level key performance indicators (KPIs) compared to the previous generation; namely: Increase by 1000 times the wireless area capacity; reduce the average service creation time cycle from 90 hours to 90 minutes; create secure, reliable and dependable internet with a “zero perceived” downtime for services provision; save up to 90% of energy per service provided; support ultra-dense device deployments; and enable advanced user controlled privacy. From the service perspective, one of the ultimate goals of 5G, is to be able to handle heterogenous services, with different, and sometimes competitive requirements over a unified network infrastructure. The three extremes of such services are [11]: i) Enhanced mobile broadband (eMBB) – also called extreme mobile broadband, ii) ultra-reliable and low latency communications (URLLC), and iii) massive machine type communications (mMTC).

Towards the above-mentioned 5G developments and performance gains, a set of technologies have been emerged, including a game changer one; the “network slicing” [12]. The main target of the network slicing concept is to enable the creation of multiple virtual isolated networks on a shared physical infrastructure, so that heterogeneous and conflicting service requirements are fulfilled. A prerequisite for network slicing implementation is the concept of decoupling network functions from dedicated hardware, to make them software functions that can be hosted in/migrated to virtual environments. The network functions softwarization, requires the transformation of physical network functions (PNFs) to virtualised/containerized network functions (VNFs/CNFs). Thus, the network slices are considered sets of VNFs, hosted on several virtual servers of a provider’s network.

In this context, resources of whatever kind (compute, network, storage, etc.) should be allocated efficiently to VNFs, fulfilling functional requirements, such as the network slice set up time. The slice set up process includes the instantiation and the configuration of the VNFs/CNFs that compose a slice; hence it affects a key 5G KPI, the service creation time [13] (referring to the time needed for a network service at the “zero” status to become fully functional).

In this article, the problem of minimising the time needed for instantiation and configuration of VNFs is mapped to the job-shop scheduling problem (JSSP) [14]. To resolve the problem the toolbox provided by the metaheuristic processes is examined. More precisely, we consider a system that receives requests for setting up network slices and provides scheduling

This research is co-financed by Greece and the European Union (European Social Fund- ESF) through the Operational Programme «Human Resources Development, Education and Lifelong Learning 2014-2020» in the context of the project CHAMELEON: service CHAins integrating Mobile Edge computing and network function virtuaLization towards optimized vidEO conteNt delivery (MIS 5070956).

Manuscript received November 4, 2021; revised Jaunary 21, 2022; approved for publication by Ruilong Deng, Division 3 Editor, September 4, 2022.

The authors are with the department of Informatics & Telecommunications, NKUA, Athens, Greece, email: {anachar, dtsolkas, passas, nio, merakos}@di.uoa.gr.

A.-S. Charismiadis is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2022.000039

1229-2370/22/$10.00 © 2022 KICS

decisions based on nature-inspired metaheuristics. Expanding our previous work on metaheuristics [15], the evolution-based and the swarm-based metaheuristic categories are targeted through two key representatives, namely: The genetic algorithm (GA) and the ant colony optimisation (ACO), respectively. For each one of the approaches, we study thoroughly the impact of configuration parameters on their performance in order to select the best-fitting solution for the VNF/CNF scheduling problem. In the following, for simplicity, we use only the term VNFs, and VNF scheduling; however the proposed approach is valid for CNFs, as well.

The remaining of this paper is organised as follows. Section II describes the network slicing concept, as well as related resource allocation techniques for placement and scheduling of VNFs that compose network slices. In Section III, the adopted system model is presented by explaining how the problem of minimizing the slice set up time is mapped to the well-known JSSP. Section IV defines a case study, where metaheuristic algorithms are utilized to solve the target problem. Section V presents performance evaluation results. Finally, in Section VI, conclusions and insights for further study are provided.

II. The Concept of Network Slicing

A. Network Slice and VNFs

The new era of telecommunications brings up plenty of high-performance standards along with heterogeneous requirements set by emerging vertical services (e.g., Internet-of-things, automotive communication), which cannot be satisfied by the conventional/monolithic network deployments. High data rates, low latency, seamless coverage, and dense connectivity are few of the most important requirements that new generation networks must fulfil. The network's infrastructure must be able to cope with resource demanding and competitive services simultaneously. The countermeasure to this challenge is the concept of network slicing, chosen to be the main pillar in the new generation of mobile networks that will enable innovative vertical services. Network slicing refers to the process of setting up on demand end-to-end logical networks, that: i) Run on a shared underlying (physical or virtual) network, ii) are mutually isolated, and iii) can have independent control and management planes [16]. The implementation of the network slicing concept is based on two key technologies, namely, the software defined networks (SDN) and the network function virtualisation (NFV).

SDN architecture refers to decoupling routing (control plane) and forwarding (data plane) functions in network nodes, thus enabling the capability of programmable network control and the import of an abstraction level in the underlying infrastructure, suitable for application and network services. SDN purpose is to address the fact that the static architecture of traditional networks was decentralized and complex, making the network unable to keep up to dynamic changes and hard for troubleshooting. By disaggregating control and data plane, SDN attempts to centralise network intelligence in a set of limited network components, thus transforming conventional networks to dynamic, manageable, cost-effective, and adaptable ones. The basic architectural components of SDN are: i) SDN controller, which is the logical centralized entity responsible for every decision in the network, ii) SDN application, which is the software component importing a desired functionality in the network, iii) SDN datapath, which is the logical network device providing the physical network device with SDN capabilities, iv) SDN interfaces, divided in two categories: Control to data-plane interface, between SDN controller and SDN datapath, and northbound interface, between SDN application and SDN controller.

NFV is the concept of virtualising network services, such as firewall, encryption, routing etc., that have traditionally been run on dedicated hardware. These services are packaged as isolated logical entities (virtual machines or containers) on commodity hardware, which allows service providers to run these on standard servers instead of proprietary ones. NFV implementation can improve scalability and agility by allowing service providers to deliver custom network services and applications on demand, minimising the need for additional hardware resources [17]. According to ETSI [18] the ***NFV framework*** consists of: i) Virtualized network function (VNF), i.e., the software implementation of a network function, ii) NFV infrastructure, including the resources to be virtualised and iii) NFV management and orchestration (MANO), which is responsible for the orchestration and lifecycle management of physical and logical resources that support infrastructure virtualisation.

Consequently, a "network slice" can be considered as a set of VNFs that are hosted in compute nodes of a network infrastructure to offer isolated and scalable network services to vertical service providers.

B. VNF Placement and Scheduling

The research interest on network slicing lies mainly in the so-called resource management problem (management of the computation power), where two key algorithmic processes are defined; the VNF placement, referring to the placement of VNFs from a set of network slice requests to the physical computing hosts, and the VNF scheduling, referring to the order that the VNF are installed/served by the hosts.

VNF placement. In the literature the algorithmic part of VNF placement is mainly addressed using the integer linear programming (ILP). For instance, in [19], ILP is used to deal with VNF placement problem, targeting at maximising continuous available time of service chain functions (SCFs). ILP is also used to formalise the problem in [20], where the authors study VNF placement and chaining in network slicing considering geographic location of network slice users. The authors in [21] are dealing with VNF placement problem, studying the degree that intra-slice isolation and end-to-end delay restrictions affect resource utilization (CPU and bandwidth). They adopt a MILP approach with AMPL to model the optimization algorithm and CPLEX as MILP solver [22]. The problem is modelled as a MILP in [23] as well, where the authors focus on how to place VNF instances in distributed data centers and schedule the appropriate network flows to minimise the total cost of VNF deployment and flow communication for big data processing.

In addition to the above-mentioned set of approaches, online solutions have been proposed as well. For instance, in [24] the authors propose a mechanism of online placement and reallocation of VNFs between core and edge networks, by using a forwarding graph embedding algorithm, with major objective to maximise the number of served users. Similarly, the research study in [25] introduces an online method to orchestrate VNFs and schedule traffic flow for network utility maximisation, using deep reinforcement learning (DRL) assisted by an optimization model.

VNF scheduling. The study in [26] implements Q-learning, a Reinforcement Learning technique, to solve delay-guaranteed VNF scheduling. The problem is handled as a time-slot problem, where in each slot a decision on which VNF will be executed is being made. The authors in [27] present a solution for VNF scheduling problem, based on GA, adopting three variations, i) feed the algorithm with deterministic initial population, ii) use an adaptive crossover probability controlled by the fitness value, and iii) apply mutation controlled by simulated annealing. In [28], VNF scheduling with E2E delay requirements among services is formulated first as MILP and then as a Markov decision process (MDP) solved with Q-learning algorithm. In [29], VNF scheduling is formulated as a flexible job-shop scheduling problem, and the authors propose a deep Q-Learning algorithm, named DQS, to solve the problem. They also compare their approach with other four algorithms, a greedy-based, a matching-based, a fair weighted and GA. A metaheuristic approach, on the other hand, is used by [30], developing two novel genetic-based algorithms, which try to optimise VNF scheduling by maximising the number of accepted service requests, and minimising the number of bottleneck links and the overall processing time. An alternative approach is presented by the authors of [31], who propose a matching-based algorithm to solve both offline and online VNF scheduling.

Joint VNF placement and scheduling. The authors in [32] present a two-phase solution for joint optimization of VNF chain placement and request scheduling problem. The authors apply the theory of open Jackson network to model VNF chains, formulate the VNF chain placement problem as a variant of variable-sized bin-packing problem, and propose a priority-driven weighted algorithm best fit decreasing using smallest used nodes (BFDSU) to optimise resource utilization and a heuristic algorithm reverse complete Karmarkar-Karp (RCKK) to minimise response latency. A similar approach for the joint optimisation problem is proposed in [33]. The authors study stateful VNF placement along with routing scheduling, formulate the problem as a Mixed Integer Linear Programming (MILP) and use a two-phase fast algorithm to deploy VNF instances and select appropriate routing paths. The objective of placing and scheduling VNFs in [34] is to minimise cost incurred by the service provider (SP) to lease computing and networking resources. Also, latency requirements of service demands, which arrive dynamically, should be satisfied. Cost-effective placing and scheduling scheme are compared with a low-latency scheme and is formulated with MILP and with a heuristic that consists of three sub-algorithms, i.e., optimal zone determination (OZD), latency requirement verification (LRV), and service demand provisioning (SDP). The study in [35] proposes an approach to jointly optimise the processes of VNF placement and VNF scheduling. Authors first solve the optimal VNF placement for all flows to simplify the problem, and then they use Lagrangian multipliers to relax and decompose it into a series of subproblems. A dynamic programming method is designed to solve each subproblem and a heuristic method is presented to construct a feasible solution for it. Finally, the solution is optimised through the subgradient optimisation method. The authors in [36] propose a graph-based algorithm to solve VNF placing and scheduling on demand, taking also some software requirements into consideration. On the other hand, the author in [37] evaluates heuristic, metaheuristic and greedy approaches for the joint optimisation of VNF placement and scheduling.

Overall, the VNF placement problem, in practice, is addressed internally in the platform that implements the virtualisation environment (e.g., a cloud computing platform, such as Kubernetes). Thus, the implementation details can affect the efficiency of the solution. In this view, proof-of-concept solutions have been developed, as for example the one presented in [38], where VNF placement and chaining is concerned for the interactive gaming use case. An implementation for VNF placement is presented in [39] as well. The authors introduce an ETSI NFV compliant, scalable, and distributed architecture, called Megalos, which allocates VNFs via a custom variation on top of the Kubernetes native VNF placement scheme. Thus, the VNF scheduling problem seems to be more attractive, as it is platform independent and, also, more critical, when it comes to time optimization criteria, like the slice set up time. Thus, the target of our work is to study the VNF scheduling problem. The essentials of the proposed approach are presented in the next section.

III. System Model

A. Problem Statement

As already mentioned, a network slice is comprised of VNFs which are chained in a specific order. We assume that a network slice provider (NSP) is responsible for fulfilling the performance requirements of a requested service, and for providing the service-related network slice(s) ready (up and running) in the shortest possible time.

Considering the ETSI MANO architecture and the 5G service-based architecture (SBA), the responsibility for the end-to-end deployment of a service-related network slice is upon the network slice management function (NSMF), which resides at the NSP [34]. Also, considering the 5G performance targets initially set by ITU IMT 2020 [40], for the network slicing preparation, the NSP targets to reduce the average service creation time from 90 hours to 90 minutes; The service creation time has been defined as the “time required to provision a service, measured since a new service deployment is requested until the overall orchestration system provides a response” [13], and it can be divided into many time-consuming phases, each of them depending on different factors. In the

TABLE I
THE MAIN PHASES OF SERVICE CREATION AND ACTIVATION TIME.

Service creation & activation phases		
Phase	**Process**	**Description**
Phase 0	Platform provision	Platform configuration, platform deployment
Phase 1	Onboarding	Onboard network slice template, network slice descriptor, etc.
Phase 2	Instantiate, configure & activate	Instantiate network slice, instantiate & activate network service, instantiate & configure VNFs in service chain, etc.
Phase 3	Modify	Modify slice, service or VNF configuration
Phase 4	Terminate	Terminate slice, service, VNF

attempt to identify the various phases of service creation, 5G-PPP has reported a collaborative reference timing flow. This timing flow does not only separate the stages in network service creation and activation process, but also defines clear starting and ending time points. A representation of the timing flow is presented in Table I. From the set of phases defined for the service creation procedure, phase 2, (named as instantiate, configure & activate) seems to be the most time-consuming stage in the whole process, since it includes the processes of instantiating, configuring and activating a slice. As a result, the minimisation of the time spent in this phase would have a positive overall effect in the network slice creation process. In this view, our target is to minimise the time needed for instantiation and configuration of a network slice, by studying fast and robust schedulers that are able to schedule resource-competitive VNFs of various network slices in a limited set of physical machines (hosts).

In this context, let the NSP manage several processing machines, located in multiple compute centers (at network core, edge etc.) on which VNFs can be hosted. We assume the following:

- Chunks of requests for deployment of network slices are periodically received by the NSP.
- Each request includes the required execution plan (VNF order) for each slice.
- The NSP, based on i) monitoring of past set-ups, ii) knowledge of the capabilities of the available processing machines and iii) network constrains and delays, estimates the time needed for setting-up each specific VNF and assigns it in the most suitable machine.
- An optimization algorithm is applied to schedule a chunk of requests, taking all constrains into account. The algorithm exports the total VNF execution plan.

To select the optimisation algorithm for the scheduling of network slice chunks, the problem is mapped to JSSP, as explained below.

B. Mapping VNF Scheduling to JSSP

In the literature, the scheduling problem mainly refers to the challenge of selecting time slots for performing a set of activities, with respect to a given set of constrains (such as resources limitations or precedence order among the activities), targeting at optimising a performance metric [41]. In most of the network related scheduling approaches, the optimisation metric is either the total processing time of the activities (i.e., the minimisation of makespan [42]) or the amount of processing resources allocated to run the activities. At the dawn of the slice-enabled networks, scheduling procedures fit well to procedures needed in sliced networks for allocating VNFs to available processing resources. This mapping is clear since every network service is modelled as a service chain, i.e., a set of network functionalities, implemented as VNFs, in a specific order [43].

A well-studied example of scheduling problem is the Job-Shop Scheduling Problem. The JSSP sets a combinatorial problem, where its solving has been used widely in literature to address resource allocation and scheduling problems [44]–[48]. Here we assume the JSSP with the following characteristics:

- Let a set of n jobs, denoted as $J_1, J_2, \cdots, J_n$;
- Within each job there is a set of operations $O_1, O_2, \cdots, O_i$, of varying processing times;
- Each operation of every job must be executed is a specific machine, among a set of m machines, denoted as $M_1, M_2, \cdots, M_m$, according to initial plan;
- Each machine can process one operation at a time and when an operation is assigned to a machine, the machine can not interrupt its execution.

The objective is to find an execution plan of the activities, w.r.t., precedence and resource constraints of the system, that minimises the time in which all operations will be executed (i.e., to find the minimum makespan). To formulate the VNF scheduling, the following mapping is assumed:

- JSSP machines are mapped to network provider's machines that host VNFs;
- Jobs correspond to network slicing requests;
- Operations are mapped to VNFs.

More specifically, based on this mapping, we use the following notation:

- S for the set of network slices;
- s for the sth network slice, where $s \in [1, |S|]$;
- F_s for the set of VNFs of slice s;
- f_{sj} for the jth VNF of the sth slice, where $s \in [1, |S|]$ and $j \in [1, |F_s|]$;
- H for the set of hosts; and h for the hth host, where $h \in [1, |H|]$;
- $h_{s,j}$ for the host where the jth VNF of the sth slice is deployed, $h_{s,j} \in [1, |H|]$;
- $t_{s,j}$ for the starting time of the jth VNF of the sth slice;
- $T_{s,j}$ for the deployment time of the jth VNF of the sth slice.

The constrains of this problem can be formalised as follows:

- $t_{s,j+1} - t_{s,j} \geq T_{s,j+1}$, representing that each VNF must be installed after the completion of the previous VNF of the same network slice;
- $t_{s,j} \geq t_{s',j'} + T_{s',j'}$ or $t_{s',j'} \geq t_{s,j} + T_{s,j}$ for $h_{s,j} = h_{s',j'}$, representing that a host cannot process two or more VNFs simultaneously.

Thus, the minimisation of makespan is defined as:

$$MinMakespan = min[max[t_{s,j} + T_{s,j}]], \tag{1}$$

where the quantity $max[t_{s,j}+T_{s,j}]$ defines the completion time of the last VNF among all network slices.

C. Metaheuristics as a Candidate Solver

Based on the mapping presented in the previous section, a solution of the VNF scheduling problem can be provided by resolving the related JSSP [49]. The JSSP is known as an NP-hard problem, meaning it cannot be solved in polynomial time [50]. Thus, it makes ineffective the use of exact methods (such as the dynamic programming, the branch & bound, etc.) since they search exhaustively to the space of all the potential solutions to find the optimal one.

Also, it is worth noted that the execution time (complexity) of the optimisation algorithm to be used affects the overall slice set-up time as well. In other words, the response time of the algorithm that performs the scheduling must be negligible, compared to the other procedures needed for setting the slice up.

Based on the above observations, heuristic methods fit well to the JSSP, as they can provide a near-optimal solution by constructing solutions according to greedy decisions [51]. Heuristics have some remarkably interesting advantages. To name some, they work well for dynamic problem sizes [52]; they find a solution in reasonable time [44]; and they can be combined with other methods [53]. However, the 'greedy' nature of heuristics is not always preferable when the problem size increases, and the near-optimal solution they provide tends to be worse, or the algorithm may stick on local minima or maxima. To compensate with the greediness of the heuristic methods, metaheuristic methods have emerged.

The metaheuristic category of algorithms refers to high-level heuristics that mimics the biological or physical phenomena. Metaheuristics are refined scientifically to find near-optimal solutions in reasonable computing time, and bring some additional advantages compared with heuristics: i) Simple and easy implementation, ii) avoiding stucking on local optima, and iii) tunning of the execution time through execution parameters that can change based on the input size [54]. Another unique feature that metaheuristics apply is the different search process. As it is analyzed in [55], these algorithms apply two phases of search: intensification and diversification. The intensification phase finds the local best solution within its adjacent location, called as local search. The diversification phase starts the search process globally in the provided search space which intend to attain the global solution, called as global search. The most challenging task in the development of any metaheuristic algorithm is to find a suitable balance between intensification and diversification.

IV. Selected Metaheuristic Algorithms

Digging into the plethora of metaheuristic methods, there is a set of algorithms called 'nature-inspired'. The nature-inspired methods, as their name implies, adopt their behavior from various nature functionalities and have been extensively used in the last decade to solve various optimisation problems in many research fields such as cloud computing [52], power consumption [53] and data-mining [54]. According to [51], nature-based methods can be classified into four divisions: Evolution-based, inspired by the theory of natural evolution, Physics-based, mimicking physical rules in the universe, Swarm-based, mimicking the social behavior of groups of animals, and Human-based, inspired by the advancement in level of searching strategy. In this study, we focus on the most efficient metaheuristic algorithms of the Evolution-based and the Swarm-based categories, namely the genetic algorithm and ant colony optimisation.

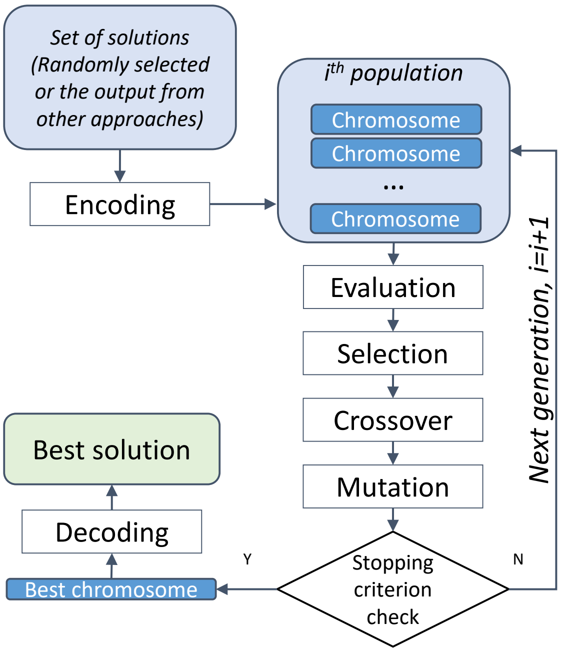


Fig. 1. Block diagram of evolution-based approach.

A. Genetic Algorithm (GA)

The GA is a genetic or evolution-based method to solve combinatorial optimisation problems such as the one studied in this paper. The block diagram of the adopted method is depicted in Fig. 1, while each one of the blocks is explained below.

1) Encoding: The encoding process refers to the process of representing a set of initial (randomly selected) solutions to arrays of VNFs with specific order. Using genetic-based terminology, each VNF is a ***gene***, and a solution (series of VNFs) is a ***chromosome***, as depicted in the example that follows:

$$Chromosome: \{f_{31}, f_{21}, f_{32}, f_{41}, f_{22}, f_{11}, f_{33}, f_{42}, f_{23}, f_{34}, f_{12}, f_{43}, f_{13}, f_{44}, f_{14}, f_{24}\}$$

2) Crossover: Crossover is a genetic operator analogous to biological reproduction, and it aims at breeding new acceptable solutions from existing ones. This procedure requires two VNF chains (chromosomes) referred to as "parents". First, from both the parent chromosomes, a portion of the same lengh is selected. Then this portion is

TABLE II
CROSSOVER EXAMPLE.

Partitioned chromosome of parent 1		
f_{31}, f_{21}, f_{32}	$f_{41}, f_{22}, f_{11}, f_{33}, f_{42}, f_{23}, f_{34}, f_{12}, f_{43}$	$f_{13}, f_{44}, f_{14}, f_{24}$
Partitioned chromosome of parent 2		
f_{21}, f_{31}, f_{32}	$f_{11}, f_{41}, f_{33}, f_{42}, f_{22}, f_{23}, f_{34}, f_{12}, f_{43}$	$f_{44}, f_{13}, f_{24}, f_{14}$
Crossover result for child 1		
f_{31}, f_{21}, f_{32}	$f_{11}, f_{41}, f_{33}, f_{42}, f_{22}, f_{23}, f_{34}, f_{12}, f_{43}$	$f_{13}, f_{44}, f_{14}, f_{24}$
Crossover result for child 2		
f_{21}, f_{31}, f_{32}	$f_{41}, f_{22}, f_{11}, f_{33}, f_{42}, f_{23}, f_{34}, f_{12}, f_{43}$	$f_{44}, f_{13}, f_{24}, f_{14}$

TABLE III
MUTATION EXAMPLE.

Chromosome of child 1 with selected genes for mutation
$f_{31}, f_{21}, \boldsymbol{f_{32}}, f_{11}, f_{41}, f_{33}, \boldsymbol{f_{42}}, f_{22}, f_{23}, f_{34}, f_{12}, f_{43}, f_{13}, f_{44}, f_{14}, f_{24}$
Chromosome of mutated child 1 [Non-acceptable]
$f_{31}, f_{21}, \boldsymbol{f_{42}}, f_{11}, f_{41}, f_{33}, \boldsymbol{f_{32}}, f_{22}, f_{23}, f_{34}, f_{12}, f_{43}, f_{13}, f_{44}, f_{14}, f_{24}$
Chromosome of mutated child 1 [Acceptable]
$f_{31}, f_{21}, \boldsymbol{f_{22}}, f_{11}, f_{41}, f_{33}, f_{42}, \boldsymbol{f_{32}}, f_{23}, f_{34}, f_{12}, f_{43}, f_{13}, f_{44}, f_{14}, f_{24}$

exchanged between the chromosomes to create two new chromosomes (children), as depicted in Table II. A portion is valid for exchange, only if duplications and invalid crossovers are avoided.

3) Mutation: Mutation is the genetic operator that preserves the genetic diversity from one generation to the next. This is succeeded by changing randomly the order of the VNFs in the chain. However, a mutation can also produce unacceptable solutions that violate constrains. To prevent this, in our implementation, the randomness of mutation is restricted to chromosome areas that the constrains are not violated.

4) Evaluation (Fitness function): The process of evaluation or fitness function is used on evaluating the solutions produced from genetic operations. This evaluation happens by calculating for every solution the objective value, that is $max[t_{s,j} + T_{s,j}]$ as defined in the objective function (1).

5) Selection: The selection mechanism is used for keeping the most "genetically robust" children of a generation, thus the solutions with the optimal service creation time. The selected children serve as parents in the next generation.

B. Ant Colony Optimisation (ACO)

In the swarm-based metaheuristic category, the ACO is the most representative paradigm. Marco Dorigo was the first to introduce ACO in his PhD thesis in 1992 [56]. The idea behind the algorithm was inspired by the foraging process of ants. When an ant of a colony searches for food, it deposits a special substance on every path that walks called pheromone. The shorter the path, the bigger the portion of pheromone in the path. Ants coming later, choose with greater probability the path with increased amount of pheromone previously deposited by other ants, thus a shorter path. As a result, ants succeed to find the shortest path to the food source. The block diagram of the algorithm is depicted in Fig. 2.

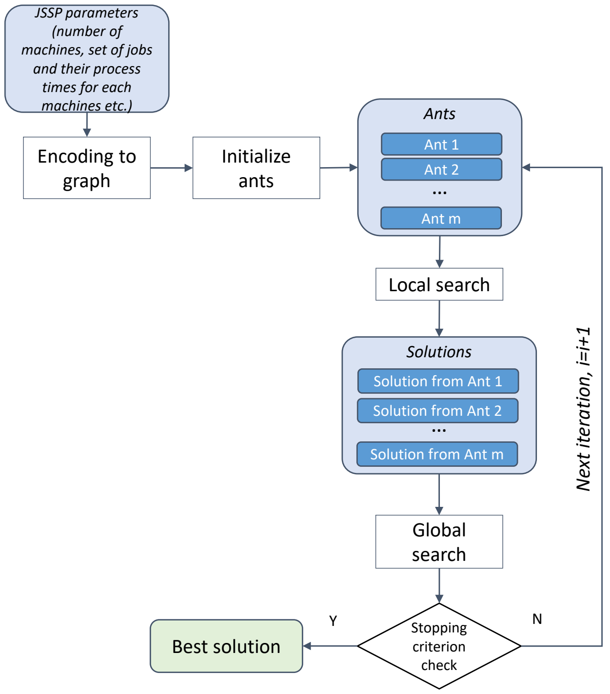


Fig. 2. Block diagram of swarm-based approach.

1) Encoding to graph: The problem must first be encoded as graph where:

- Each VNF is represented by a node.
- VNFs belonging to the same network slice request are connected with unidirectional edges, respecting the order of processing.
- The rest of the VNFs are connected with bidirectional edges.
- The cost of its edge is an ant-inspired parameter $\tau_{j_1 \to j_2}$, called pheromone concentration. The pheromone concentration quantity refers to the amount of pheromone deposited by each ant for transition from VNF/node j_1 to VNF/node j_2. The contribution of each ant m that has passed from the VNF/node j_1 to VNF/node j_2 is given by the following quantity:

$$\Delta\tau^m_{j_1 \to j_2} = \begin{cases} \dfrac{Q}{t_{s,j_2} + T_{s,j_2}}, & \text{if ant } m \text{ used path from } j_1 \text{ to } j_2 \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where Q is a constant, selected to tune the size of the ratio $Q/(t_{s,j_2} + T_{s,j_2})$, based on the volumes of the denominator (i.e., is a problem specific quantity). Thus, the $\tau_{j_1 \to j_2}$ is updated as follows:

$$\tau'_{j_1 \to j_2} \leftarrow (1 - \rho)\tau_{j_1 \to j_2} + \sum_{m}^{M} \Delta\tau^m_{j_1 \to j_2}, \tag{3}$$

where $\tau_{j_1 \to j_2}$ current value and $\tau'_{j_1 \to j_2}$ the updated one. Also, ρ is the pheromone evaporation coefficient, M is the number of ants, and the sum $\sum_m^M \Delta\tau^m_{j_1 \to j_2}$ is the contribution of the ants that passed from the representing edge, as defined in (2).

2) Initialise ants: In this stage, the number of the ants (usually being equal to the number of network slice requests) is defined. For our analysis, we denote by M the number of ants/agents in the system. For each ant, m, the number of iterations is also defined, i.e., the number of times that an ant is going to run through the graph (apply a local search as explained below).

3) Local search: The local search refers to the path followed by an ant in the graph. Each ant, starting from a VNF/node j_1 of the graph, moves to the adjacent node j_2 with probability:

$$P^{m}_{j_1 \to j_2} = \frac{[\tau_{j_1 \to j_2}]^{\alpha}[\eta_{s,j_2}]^{\beta}}{\sum_{l \in AllowedTransitions_{j_1}} [\tau_{j_1 \to l}]^{\alpha}[\eta_{s,l}]^{\beta}}, \tag{4}$$

where,

- Attractiveness $\eta_{s,j}$, represents a heuristic value of the cost of selecting the move, and it is equal to the reverse of the duration of VNF deployment, i.e., $1/T_{s,j}$.
- $\sum_{l \in AllowedTransitions_{j_1}} [\tau_{j_1 \to l}]^{\alpha}[\eta_{s,l}]^{\beta}$ is the cumulative product of pheromone concentration and heuristic value for all possible node transitions from j_1.
- α and β parameters, are arbitrary values, $\alpha \in [0, \infty)$, $\beta \in [1, \infty]$, that determine the degree of influence from the heuristic and pheromone values, respectively.

4) Global search: When the ants finish their job and produce their solutions, that is a valid sequence of VNFs, the algorithm keeps the solution with the best makespan. Then the pheromone trails in each edge are globally updated as defined in (3). When one of the stopping criteria is met, then the output of the algorithm is the VNF sequence with the best makespan until that time.

V. Performance Evaluation

The two algorithms were tested for a wide range of values for their configurable parameters, while a numerous set of runs has been executed to extract each evaluation result. To evaluate the algorithms, we adopted a 10×10 (i.e., #slices $|S| = 10$, and #VNFs $|F_s| = 10$ per slice) and a 20×5 dataset (i.e., #slices $|S| = 20$, and #VNFs $|F_s| = 5$ per slice), named "orb04" and "ft20" (from [57]), with known minimum makespans equal to 1005 and 1165 time-units, respectively. We also assume known deployment times and machines for the input, i.e., known values for the $T_{s,j}$, and $h_{s,j}$, parameters. More precisely, the format of every dataset follows the rules listed below (we use a 6×6 example for convenience):

- The first row defines the number of network slices requests and number of VNF hosts for the specific instance, e.g.: "6 6", "10 10" or "20 5".

Algorithm 1 Genetic algorithm (GA)

```
$initialPopulation, minimumMakespan \leftarrow ChooseInitialParents(schedulingPlanFile)$
$define\ value\ for$ :
    $crossover_{threshold}$,
    $mutation_{threshold}$,
    $numberOfGenerations$,
    $numberOfPopulation$
for $par_x$, $par_y$ $in\ initialPopulation$ do
    for $m\ in\ [1, numberOfGenerations]$ do
        for $n\ in\ [1, \frac{numberOfPopulation}{2}]$ do
            $crossover_{possibility} \leftarrow random(0,1)$
            if $crossover_{possibility} \leq crossover_{threshold}$ then
                $child_i, child_j \leftarrow crossover(par_x, par_y)$
            end if
            $mutation_{possibility} \leftarrow random(0,1)$
            if $mutation_{possibility} \leq mutation_{threshold}$ then
                $child_i \leftarrow mutation(child_i)$
                $child_j \leftarrow mutation(child_j)$
            end if
            $selectionList.append\begin{pmatrix} fitnessValue(child_i), \\ fitnessValue(child_j) \end{pmatrix}$
            $par_x \leftarrow child_i, par_y \leftarrow child_j$
            $sort(selectionList)$
        end for
    end for
end for
if $first\ element\ of\ selectionList \leq minimumMakespan$ then
    $minimumMakespan \leftarrow first\ element\ of\ selectionList$
    $par_x \leftarrow first\ element\ of\ selectionList$
    $par_y \leftarrow second\ element\ of\ selectionList$
end if
$MinimumMakespan$
```

Algorithm 2 Ant colony optimisation (ACO)

```
$graph \leftarrow Create\_Graph(schedulingPlanFile)$
$ants \leftarrow InitializeAnts(numberOfAnts)$
$define\ value\ for$ :
    $iterations$,
    $\alpha$,
    $\beta$
for $i\ in\ iterations$ : do
    $node_{present} \leftarrow InitialNode(0)$
    $makespan_{global} \leftarrow emptyList()$
    for $ant\ in\ ants$ do
        $path_{ant} \leftarrow list(node_{present})$
        $Until\ all\ nodes\ are\ passed$ :
        for $node_{possibleNext}\ in\ listNextNodes$ do
            $Calculate$ :
            $P^{k}_{i \to j}, i \leftarrow node_{present}, j \leftarrow node_{possibleNext}$
            $P_{min},\ node_{next} = findMinimum(P^{k}_{i \to j})$
            $path_{ant} \leftarrow path_{ant}.add(node_{next})$
            $node_{present} \leftarrow node_{next}$
        end for
    end for
end for
$makespan_{ant} \leftarrow estimateMakespan(path_{ant})$
$makespan_{global} \leftarrow makespan_{global}.add(makespan_{ant})$
$Global\_pheromone\_update()$
$MinimumMakespan \leftarrow findMinimum(makespan_{global})$
```

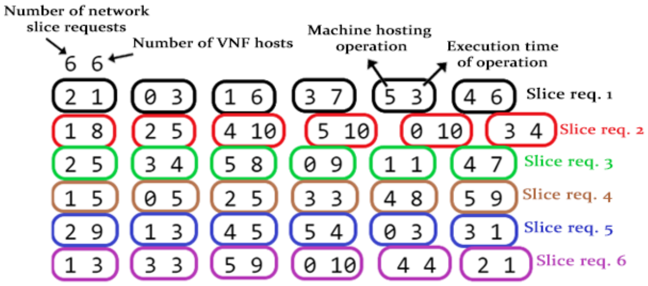

Number of network slice requests
Number of VNF hosts
Machine hosting operation
Execution time of operation
6 6
2 1 | 0 3 | 1 6 | 3 7 | 5 3 | 4 6 Slice req. 1
1 8 | 2 5 | 4 10 | 5 10 | 0 10 | 3 4 Slice req. 2
2 5 | 3 4 | 5 8 | 0 9 | 1 1 | 4 7 Slice req. 3
1 5 | 0 5 | 2 5 | 3 3 | 4 8 | 5 9 Slice req. 4
2 9 | 1 3 | 4 5 | 5 4 | 0 3 | 3 1 Slice req. 5
1 3 | 3 3 | 5 9 | 0 10 | 4 4 | 2 1 Slice req. 6

Fig. 3. JSSP dataset format.

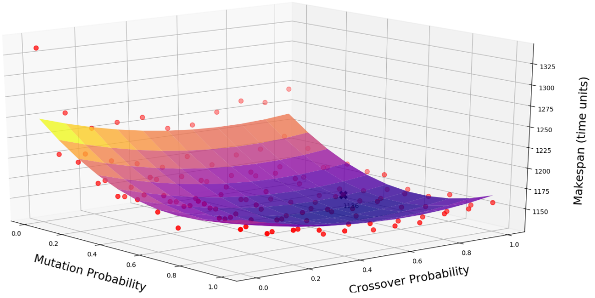


Fig. 4. Average service creation time in GA for “orb04” (5 runs per crossover-mutation probability combination).

- Each row contains a set of pairs whose number is equal to the number of VNFs of each network slice.
- Each pair of numbers describes a specific VNF by defining the machine hosting the operation and the duration of executing the operation in time units estimated by the NSP e.g. “2 1” means that the VNF is hosted by machine 2 and its execution will last 1 time-unit.

A. Numerical Results

The performance of the GA is tightly correlated with its tunning parameters, namely, the number of generations, the population size, the crossover and the mutation probability. We first study the impact of crossover and the mutation probabilities on the achievable makespan. The actions of crossover and mutation help the algorithm to overcome the problem of stucking on local optima by changing the order of the VNFs. As it is depicted on Fig. 4, for 40 generations and population size equal to 100, the performance of the algorithm is improved as more crossover and mutation actions are operated. However, for probabilities higher than 0.6, the slope of the performance remains rather stable. In order to define more exactly the best values for these parameters, we conducted another experiment, shown at Fig. 5. In this experiment we tried to clarify which is the best combination of the two parameters. According to the graph, the algorithm performs better when the number of crossovers and mutations tends to be equal. It can be observed in Fig. 5 that the mutation action results in reduced makespan compared to crossover, thus having a greater impact on the overall algorithm. As a result, for the rest of the experiments, we choose to set the values of crossover and mutation equal to 0.7.

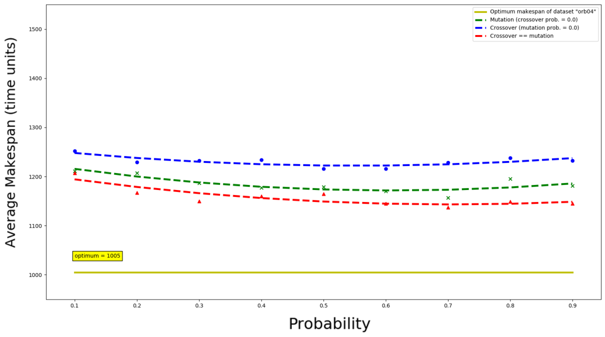


Fig. 5. Average service creation time in GA for “orb04” per probability value.

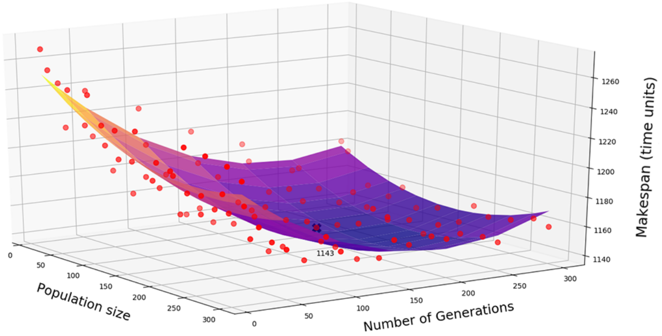


Fig. 6. Average service creation time in GA for “orb04” (5 runs per generation-population size combination).

The next parameters we examined on the GA were the number of generations and the population size. These parameters define the extend that the algorithm is going to search for a near-optimal solution. As their number increases, a wider part of the solution space is searched, and therefore, the probability of finding a better solution is also increasing. However, increasing recklessly the number of these parameters cannot be considered as a viable strategy, because it leads to enormous and unacceptable execution time. Hence it is important to choose the referred parameters in a way that both quality and acceptable time are satisfied. According to the results depicted in Fig. 6, the increase in the number of population and generations is indeed helping the performance, though it can be observed that makespan is minimised for population and generation values between 150 and 200, respectively.

ACO on the other hand, as we have already mentioned in the previous section, follows a swarm-based strategy, comprised of many agents trying to find the optimal execution order of VNFs. The most important factors in the specific algorithm are the number of iterations, that is how many times the agents will run, and the pheromone decay rate, which defines the rate of system memory for previous runs of the agents. In order to examine the impact of the above parameters on the overall performance, we run the algorithm for multiple iterations and for specific decay rates (0, 0.3, 0.7, 1) covering its whole value space. According to the results presented in Fig. 7, decay rate plays an especially important role on the performance of the algorithm. More specifically, better results seem to come out when rate is close to 0.7, while at the same

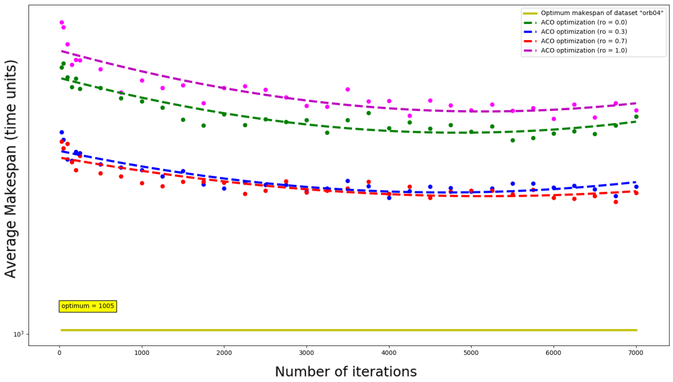


Fig. 7. Average service creation time in ACO for "orb04" per iteration number.

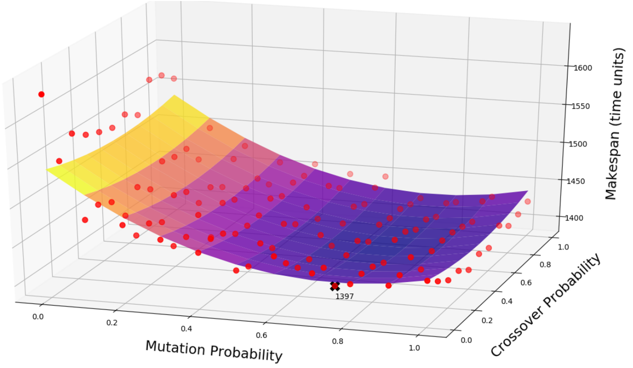


Fig. 8. Average service creation time in GA for "ft20" (5 runs per crossover-mutation probability combination).

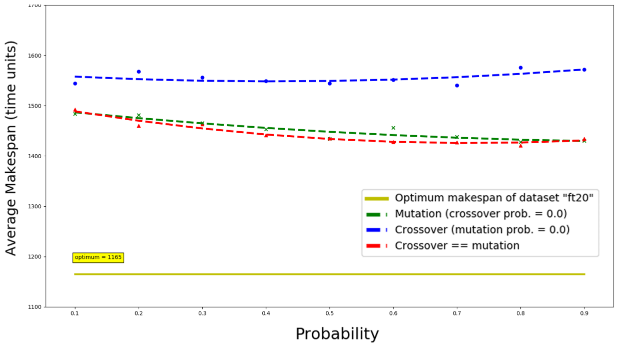


Fig. 9. Average service creation time in GA for "ft20" per probability value.

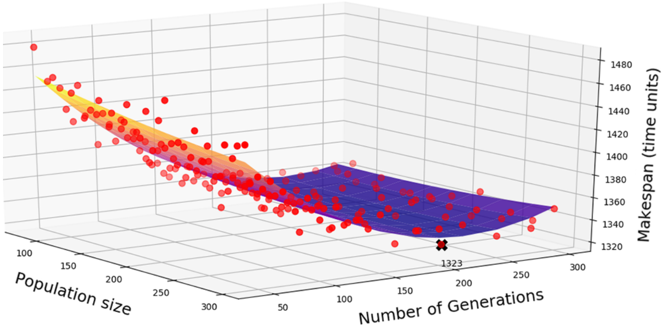


Fig. 10. Average service creation time in GA for "ft20" (5 runs per generation-population size combination).

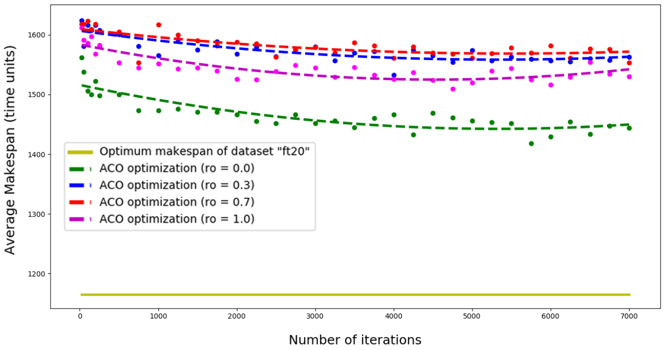


Fig. 11. Average service creation time in ACO for "ft20" per iteration number.

time, both full-memory (rate equal to 0) and no-memory (rate equal to 1) schemes downgrade the performance. Concerning the number of iterations, it can be deducted from the graph that their increment improves the results. However, makespan seems to stabilize after 4500 iterations. As a result, keeping the iterations around 4500 guarantees satisfying results at acceptable time.

A factor we had not included in our previous work [15], was whether the characteristics of the problem play some role in choosing the best parameters in each algorithm. To fill that blank, we run GA and ACO for a 20×5 JSS problem. The 20×5 problem is consisted of 20 slice requests, with 5 VNFs each, that must be allocated in 5 hosts. While the number of different VNFs is the same as the 10×10 problem, however, the distribution of VNFs to hosts has changed. Following the same approach with the previous series of experiments, we first examine the impact of GA's probability parameters on the makespan. The results in Fig. 8, show a small difference compared to the corresponding graph in 10×10. While the increment in mutation probability improves the performance, crossover probability does not influence the results that much. The above can be observed in Fig. 9 as well, where the change in crossover probability (blue line) does not have any crucial effect on the makespan. Due to this fact, as it is also depicted in Fig. 9, the total absence of crossover probability (green line) is close to the condition that crossover and mutation probabilities are equal (red line).

Generation and population parameters seem to follow the same pattern in 20×5 problem as in 10×10. However, it can be observed in Fig. 10 that the threshold defining the result stabilization has been slightly increased to 200 generations while the population parameter seems to have less effect on the makespan than in the 10×10 problem.

The execution of ACO for 20×5 problem brought up an interesting result, shown in Fig. 11. While in the 10×10 problem, the performance improved when the decay rate was between 0.3 and 0.7, in 20×5 the situation has been inverted. The total absence of decay seems to minimise the makespan, which means that for a limited number of machines it is better to keep a full-memory strategy among run of the agents.

B. Overall Assessment and Insights

To assess which metaheuristic approach fits better to this kind of scheduling problems, Table IV depicts the optimal makespan for three sizes of the target problem (10 × 10, 20 × 5, and 20 × 20). Those optimal values are compared to the actual makespan values for the two target algorithms. Both approaches seem to achieve near-optimal results. The deviation between the optimal and the actual results of the algorithms is related to the number of slice requests. As it can be observed, for the 10 × 10 problem, both GA and ACO, can achieve a close to optimal makespan. According to the experiments, GA leads to slightly better results than ACO. Specifically, GA offers approximately an 8% performance improvement but with longer execution times in exchange.

Based in the theoretical study and the performance evaluation results:

- While both evaluation-based and swarm-based algorithms perform well on small and medium problems (less than 100 VNFs), they are inappropriate for bigger problems (more than 100 VNFs) due to the high increase of their execution time. As a countermeasure, ACO, due to its agent-based architecture, can be efficiently implemented as a distributed /multi-threaded application to reduce the execution time.
- The number of slice requests plays more crucial role on the algorithms' performance than the number of machines used to host the VNFs. This observation reveals the critical role of the way that the requests for network service deployment are organized for feeding the scheduling process. In other words, the performance can be increased if we split the input pool of requests and address them in small groups rather by adding more hosts in the system.
- ACO approach can be easily adapted to function on dynamic arrival of network slice requests, in contrast to GA. This is because the GA approach starts from a random solution from a given number of requests, while the ACO progressively builds a result, which allows the addition of more requests by dynamically expanding the representation graph that is used for the ants' local search.
- In terms of complexity, the nature of metaheuristic algorithms does not facilitate an absolute way of comparing them. Indeed the complexity in both the selected algorithms is affected by their tuning parameters. As such, the GA's complexity could be expressed as:

$$O\left(\begin{array}{r} initialPopulation \times \\ numberOfGenerations \times \\ \dfrac{numberOfPopulation}{2} \end{array}\right)$$

and ACO's as:

$$O\left(\begin{array}{r} iterations \times \\ numberOfAnts \times \\ numberOfNodes \end{array}\right)$$

To better reflect the relation between complexity and efficiency for the selected metaheuristic algorithms, Table IV includes the tuning parameters used for each experiment.

TABLE IV
PERFORMANCE COMPARISON OF THE PROPOSED APPROACHES.

Problem size	Optimal makespan	GA performance		ACO performance	
		Results	Parameters	Results	Parameters
10 × 10	1005	1042	N = 200	1126	iter = 4500
			M = 140		
			crossover = 0.7		r = 0.7
			mutation = 0.7		
			time = 8 min		time = 2 min
20 × 5	1165	1277	N = 280	1388	iter = 3000
			M = 220		
			crossover = 0.7		r = 0
			mutation = 0.7		
			time = 36 min		time = 3 min
20 × 20	826	1070	N = 80	1162	iter = 6750
			M = 120		
			crossover = 0.7		r = 0.7
			mutation = 0.7		
			time = 76 min		time = 37 min

VI. CONCLUSION

Service creation time is one of the fundamental KPIs in 5G systems, since it can have a great impact on the overall system performance, and eventually to the quality of the services that an operator provides to vertical industries. The optimisation of the service creation time is a challenging task, as it can be affected by multiple parameters that reside at the implementation choices, as well as at the management methods (algorithmic processes) that are used. In this context, we examined metaheuristic methods as enablers for resolving the VNF scheduling under various service requests in a 5G system. The evaluation process revealed that metaheuristic algorithms can efficiently contribute towards the minimization of service creation time. From the various categories of metaheuristic algorithms, the evolution-based approach (in our case the GA) provides an overall better performance than the swarm-based one (in our case the ACO). However, considering that the slice set-up process is be real-time process, swarm-based algorithms, such as the ACO, have the advantage of short execution time, compared to the evolution-based ones.

Although metaheuristics seem to be an efficient solution for VNF scheduling problem, the analysis revealed some drawbacks, such as their low-fitting to dynamic scheduling (on-line) problems (that is the case when new network slice requests arrive and have to be scheduled on-line). Building on top of this study, we plan to relax the assumption on the way that the slice requests arrive and examine machine learning approaches toward an automated alignment of the scheduling to the dynamically changing input.

REFERENCES

[1] "5G Market Status: Snapshot January 2020," GSA. [Online]. Available: https://gsacom.com/paper/5g-market-status-snapshot-january-2020/

[2] "5G-PPP Progress Monitoring Report - 2019," 5G Public-Private Partnership, Report, Aug. 2020. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2020/10/5G-PPP-PMR2019v1-6.pdf

[3] D. Wu, Z. Zhang, S. Wu, J. Yang, and R. Wang, "Biologically inspired resource allocation for network slices in 5G-enabled Internet of things," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9266–9279, 2019.

[4] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1409–1434, 2019.

[5] B. Dzogovic, V. T. Do, B. Feng, and T. van Do, "Building virtualized 5G networks using open source software," in *Proc. IEEE ISCAIE*, 2018.

[6] Q. Ye *et al.*, "Dynamic radio resource slicing for a two-tier heterogeneous wireless network," *IEEE Trans. Veh. Technol.*, vol. 67, no. 10, pp. 9896–9910, 2018.

[7] L. Liang, S. Xie, G. Y. Li, Z. Ding, and X. Yu, "Graph-based resource sharing in vehicular communication," *IEEE Trans. Wireless Commun.*, vol. 17, no. 7, pp. 4579–4592, 2018.

[8] G.-K. Chang and Y.-W. Chen, "Key fiber wireless integrated radio access technologies for 5G and beyond," in *Proc. IEEE OECC/PSC*, 2019.

[9] X. Ling, J. Wang, T. Bouchoucha, B. C. Levy, and Z. Ding, "Blockchain radio access network (b-ran): Towards decentralized secure radio access paradigm," *IEEE Access*, vol. 7, pp. 9714–9723, 2019.

[10] "Contractual Agreement," 5G-PPP. [Online]. Available: https://5g-ppp.eu/contract/

[11] "5G-PPP Architecture Working Group, View on 5G Architecture Version 2.0," 5G Public-Private Partnership, White Paper, Dec. 2017. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2018/01/5G-PPP-5G-Architecture-White-Paper-Jan-2018-v2.0.pdf

[12] "5G-PPP Architecture Working Group, View on 5G Architecture Version 3.0," 5G Public-Private Partnership, White Paper, Feb. 2020. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2020/02/5G-PPP-5G-Architecture-White-Paper_final.pdf

[13] "5G PPP phase II KPIs – Annex to Programme Management Report," 5G Public-Private Partnership, Report, Jan. 2020. [Online]. Available: https://bscw.5g-ppp.eu/pub/bscw.cgi/312793

[14] J. F. Riera, E. Escalona, J. Batallé, E. Grasa, and J. A. García-Espín, "Virtual network function scheduling: Concept and challenges," in *Proc. IEEE SaCoNeT*, 2014.

[15] A.-S. Charismiadis, D. Tsolkas, N. Passas, and L. Merakos, "A metaheuristic approach for minimizing service creation time in slice-enabled networks," in *Proc. IEEE ICC*, 2020.

[16] J. Ordonez-Lucena *et al.*, "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 80–87, 2017.

[17] "What is NFV?," RedHat official website. [Online]. Available: https://www.redhat.com/en/topics/virtualization/what-is-nfv

[18] "Network Function Virtualization (NFV); Architectural Framework," European Telecommunications Standards Institute, Group Specification, Oct. 2013. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/\\gs_nfv002v010101p.pdf

[19] R. Kang, F. He, T. Sato, and E. Oki, "Virtual network function allocation to maximize continuous available time of service function chains," in *Proc. IEEE CloudNet*, 2019.

[20] J. J. A. Esteves, A. Boubendir, F. Guillemin, and P. Sens, "Location-based data model for optimized network slice placement," in *Proc. IEEE NetSoft*, 2020.

[21] D. Sattar and A. Matrawy, "Optimal slice allocation in 5G core networks," *IEEE Netw. Lett.*, vol. 1, no. 2, pp. 48–51, 2019.

[22] D. Dietrich, C. Papagianni, P. Papadimitriou, and J. S. Baras, "Network function placement on virtualized cellular cores," in *Proc. IEEE COMSNETS*, 2017.

[23] L. Gu, J. Hu, D. Zeng, S. Guo, and H. Jin, "Service function chain deployment and network flow scheduling in geo-distributed data centers," *IEEE Trans. Netw. Sci. Engineer.*, vol. 7, no. 4, pp. 2587–2597, 2020.

[24] I. Sarrigiannis *et al.,* "Online VNF lifecycle management in an MEC-enabled 5G IoT architecture," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4183–4194, 2020.

[25] L. Gu *et al.*, "Intelligent VNF orchestration and flow scheduling via model-assisted deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 279–291, 2020.

[26] J. Li, W. Shi, N. Zhang, and X. S. Shen, "Reinforcement learning based VNF scheduling with end-to-end delay guarantee," in *Proc. IEEE ICCC*, 2019.

[27] Q. Li *et al.*, "An improved genetic algorithm for the scheduling of virtual network functions," in *Proc. IEEE APNOMS*, 2019.

[28] J. Li, W. Shi, N. Zhang, and X. Shen, "Delay-aware VNF scheduling: A reinforcement learning approach with variable action set," *IEEE Trans. Cognitive Commun. Netw.*, vol. 7, no. 1, pp. 304–318, 2021.

[29] T. Wang, J. Zu, G. Hu, and D. Peng, "Adaptive service function chain scheduling in mobile edge computing via deep reinforcement learning," *IEEE Access*, vol. 8, pp. 164 922–164 935, 2020.

[30] M. Gamal, S. Jafarizadeh, M. Abolhasan, J. Lipman, and W. Ni, "Mapping and scheduling for non-uniform arrival of virtual network function (VNF) requests," in *Proc. IEEE VTC*, 2019.

[31] C. Pham, N. H. Tran, and C. S. Hong, "Virtual network function scheduling: A matching game approach," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 69–72, 2018.

[32] Q. Zhang *et al.*, "Joint optimization of chain placement and request scheduling for network function virtualization," in *Proc. IEEE ICDCS*, 2017.

[33] J. Shi *et al.*, "Joint optimization of stateful VNF placement and routing scheduling in software-defined networks," in *Proc. IEEE ISPA/IUCC/BDCloud/SocialCom/SustainCom*, 2018.

[34] T. Gao *et al.*, "Cost-efficient VNF placement and scheduling in public cloud networks," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4946–4959, 2020.

[35] B. Ren, S. Gu, D. Guo, G. Tang, and X. Lin, "Joint optimization of VNF placement and flow scheduling in mobile core network," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1900–1912, 2022.

[36] G. M. dos Santos and D. M. Batista, "On-demand placement and scheduling of virtual network functions with software requirements," in *Proc. IEEE LATINCOM*, 2020.

[37] R. Mijumbi *et al.*, "Placement and scheduling of functions in network function virtualization," 2015, *arxiv:1512.00217*.

[38] J. J. Alves Esteves, A. Boubendir, F. Guillemin, and P. Sens, "Optimized network slicing proof-of-concept with interactive gaming use case," in *Proc. IEEE ICIN*, 2020.

[39] M. Scazzariello, L. Ariemma, G. D. Battista, and M. Patrignani, "Megalos: A scalable architecture for the virtualization of network scenarios," in *Proc. IEEE/IFIP NOMS*, 2020.

[40] "Minimum requirements related to technical performance for IMT-2020 radio interface(s)," International Telecommunication Union - Radio, Report, Nov. 2017. [Online]. Available: https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-M.2410-2017-PDF-E.pdf

[41] P. Brucker and S. Knust, *Complex Scheduling*. Springer Berlin, Heidelberg, 2012.

[42] Anshulika and L. A. Bewoor, "A genetic algorithm approach for solving a job shop scheduling problem," in *Proc. IEEE ICCCI*, 2017.

[43] L. Ruiz *et al.,* "A genetic algorithm for VNF provisioning in NFV-enabled cloud/MEC RAN architectures," *Appl. Sci.*, vol. 8, no. 12, 2018.

[44] X. Xi, L. Jiang, and Q. Zhang, "Optimization for multi-resources-constrained job shop scheduling based on three-level heuristic algorithm," in *Proc. IEEE CAR*, 2009.

[45] W. Wu, J. Wei, and X. Guan, "A hybrid algorithm for scheduling in job shop problem with flexible resources," in *Proc. IEEE ICCA*, 2010.

[46] C. Kurera and P. Dasanayake, "New approach to solve dynamic job shop scheduling problem using genetic algorithm," in *Proc. ICITR*, 2018.

[47] M. Gamal, S. Jafarizadeh, M. Abolhasan, J. Lipman, and W. Ni, "Mapping and scheduling for non-uniform arrival of virtual network function (VNF) requests," in *Proc. IEEE VTC*, 2019.

[48] Q. Li *et al.,* "An improved genetic algorithm for the scheduling of virtual network functions," in *Proc. IEEE APNOMS*, 2019.

[49] Z. Shen and L. Smalov, "Comparative performance of genetic algorithm, simulated annealing and ant colony optimisation in solving the job-shop scheduling problem," in *Proc. IEEE ICSEng*, 2018.

[50] "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976. [Online]. Available: http://www.jstor.org/stable/3689278

[51] A. Mungwattana and K. Ploydanai, "Future makespan heuristic for job shop scheculing problem," in *Proc. IEEE CIE*, 2010.

[52] J. Cui and T. Li, "A hybrid heuristic neighborhood algorithm for the job shop scheduling problem," in *Proc. IEEE ICNC*, 2008.

[53] Y. Li, S. Wang, L. Ding, and X. Xie, "A dynamic programming based heuristic in max-algebra for solving a blocking flow-shop problem," in *Proc. IEEE ICMICA*, 2013.

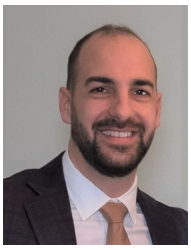
[54] C.-W. Tsai and J. J. P. C. Rodrigues, "Metaheuristic scheduling for cloud: A survey," *IEEE Syst. J.*, vol. 8, no. 1, pp. 279–291, 2014.

[55] R. Rajakumar, P. Dhavachelvan, and T. Vengattaraman, "A survey on nature inspired meta-heuristic algorithms with its domain specifications," in *Proc. IEEE ICCES*, 2016.

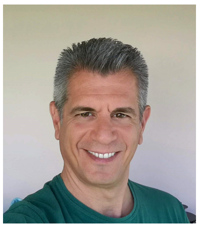
[56] M. Dorigo, "Optimization, learning and natural algorithms," 1992.

[57] "Benchmark instances for the job-shop scheduling problem (minimizing makespan)," Github. [Online]. Available: https://github.com/tamy0612/JSPLIB

Anastasios-Stavros Charismiadis received the B.S. degree and the M.S. degree from the Department of Informatics and Telecommunications, National and Kapodistrian University of Athens (NKUA), in 2015 and 2018, respectively. He is currently pursuing a Ph.D. degree in the area of resource allocation in 5G networks. He works as a Research Associate at NKUA, participating in EC-funded research and development projects, and his current research interests include resource allocation schemes and optimization algorithms for network slicing-enabled networks in the 5G and beyond era.



Dimitris Tsolkas received the Ph.D. degree from the Department of Informatics and Telecommunications, National and Kapodistrian University of Athens (NKUA). He has long experience in research and development and project management, working for a plethora of EC-funded research and innovation actions. He also has teaching experience as a Lecturer and an Assistant Instructor with the Department of Informatics and Telecommunications, NKUA, and the Department of Computer Science and Engineering, University of Ioannina. He is currently a Senior Research Fellow at NKUA. So far, he has published more than 40 articles in peer-reviewed journals, international conferences, and book chapters, while he has served as a reviewer for high-quality journals and magazines. With key expertise in 5G network functions optimization and end-to-end network emulation and testing, his current research interests include wireless/mobile communication systems, radio access networks design and analysis, and user experience management in service provisioning.



Nikos Passas received the Diploma degree (Hons.) from the Department of Computer Engineering, University of Patras, Greece, in 1992, and the Ph.D. degree from the Department of Informatics and Telecommunications, University of Athens, Greece, in 1997. He is currently a member of the Teaching Staff with the Department of Informatics and Telecommunications, University of Athens. He is a Group Leader of the Green, Adaptive and Intelligent Networking (GAIN) Research Group. Over the years, he has participated and coordinated a large number of national and European research projects. He has served as a Guest Editor and a Technical Program Committee Member in prestigious magazines and conferences, such as IEEE Wireless Communications Magazine, Wireless Communications and Mobile Computing journal, IEEE Vehicular Technology Conference, IEEE PIMRC, and IEEE GLOBECOM.

Dionysis Xenakis is Assistant Professor in the area of "Management of Networks and Computing Infrastructures" at the Department of Digital Industry Technologies of the National and Kapodistrian University of Athens (NKUA), Greece. Dionysis received the Ph.D. degree from the Department of Informatics and Telecommunications at NKUA in 2014 and has participated in numerous EU-funded projects since 2010. He is co-author of more than 40 peer-reviewed journal and conference papers, while he has chaired and served as TPC member in numerous top-tier IEEE conferences. Dionysis has been reviewer to almost all high-ranking IEEE journals in Computer Science - Data Networks and is currently Associate Editor in IEEE Networking Letters. His current research interests lie in the design and analysis of 5G/B5G mobile data networks with a focus on multi-access edge computing and distributed ledger technologies.

Lazaros Merakos received the Diploma in Electrical and Mechanical Engineering from the National Technical University of Athens, Greece, and the M.S. and Ph.D. degrees in Electrical Engineering from the State University of New York, Buffalo, NY. He has been on the Faculty of the University of Connecticut (1983–1986), and Northeastern University, Boston (1986–1994). Since 1994, he is a Professor with the Department of Informatics and Telecommunications, National and Kapodistrian University of Athens (NKUA), where he has been the Scientific Director of the Network Operations and Management Center. Since June 2020, he has been serving as the Chair for the newly founded Department of Digital Industry Technologies, NKUA. He has been the Principal Investigator for NKUA in many EU funded research and development projects in the area of mobile/wireless networks. His research interests include network technologies, services, and applications, where he has authored more than 300 publications in international journals and conferences. He was a recipient of the Guanella Award for the Best Paper presented at the 1994 International Zurich Conference on Mobile Communications and the Best Paper Award in the 2008 IEEE International Symposium on Communications, Control and Signal Processing. He is the Chairman of the Board of the Greek Universities Network, a nonprofit organization for the design and development of advanced ICT services for the 25 universities in Greece.