A metaheuristic approach for minimizing service creation time in slice-enabled networks

Anastasios-Stavros Charismiadis[×], Dimitris Tsolkas[× +], Nikos Passas[×] and Lazaros Merakos[×]
[×]*National and Kapodistrian University of Athens, Department of Informatics & Telecommunications*
Panepistimiopolis, Ilissia, 15784, Athens, Greece
{anachar, dtsolkas, passas, merakos}@di.uoa.gr
[+]*Fogus Innovations & Services P.C.,*
Michali Karaoli 51-53, 16121, Kaisariani, Greece
dtsolkas@fogus.gr

***Abstract*— A fundamental technology towards the next generation of mobile networks is the Network Slicing, i.e., the capability to "slice" network resources and functions and to offer isolated end-to-end network services over shared physical infrastructures. From the performance perspective, one of the main challenges in network slicing is the minimisation of the slice set up time; a factor that directly effects a key performance indicator for the 5G networks, namely, the service creation time. Treating a network slice as a set of Virtual Network Functions (VNFs) that are installed in different hosts (central/edge clouds) during the slice creation procedure, we examine a nature-inspired metaheuristic approach for scheduling the VNFs installation process. Simulation results prove that the proposed metaheuristic method can offer high quality solutions for VNF scheduling problems and optimize service creation time for a set of requested network slices, by producing a close-to-optimal scheduling plan, for a large set of VNFs, in reasonable execution time.**

Keywords—5G, KPI, service creation time, network slicing, VNF, job-shop scheduling problem, meta-heuristics, genetic algorithm

I. Introduction

The era of fifth generation (5G) networks has already emerged worldwide. The 5G architectural and technical specifications bring a radical change in the way telecommunication systems are built. A paradigm shift is observed where the conventional isolated hardware-based approach is replaced by a flexible software-based one. Inevitably, a critical factor in this shift is the introduction of the network slicing concept [1]. Network slicing is one of the core mechanisms in 5G networks that allows for the instantiation of multiple virtual networks over shared physical infrastructures.

Key role in the realisation of network slicing plays the Network Function Virtualization (NFV) technology. NFV is about decoupling network functions from dedicated hardware and hosting them in virtual machines. Practically, it refers to the transformation of the conventional Physical Network Functions (PNF), that reside at the core and access domains of a mobile network, to Virtual Network Functions (VNF).

Considering network slices as set of VNFs, hosted on several virtual servers of provider's network, many challenging research aspects arise. The major one is how resources, of whatever kind (compute, network, storage etc.), are allocated efficiently to fulfil the functional and performance requirements of the network slices. Reasonably, the challenge is becoming critical when the resources are in scarcity (e.g., in scenarios where edge network nodes are used) and when network providers/operators are called to serve simultaneously multiple and heterogeneous requests for network slice set up.

One metric for assessing the efficiency of the resource management in this context, is a well-known 5G Key Performance Indicator (KPI), called service creation time [2]. Service creation time describes the time needed so as a network service at the "zero" status becomes fully functional. This value is composed of various time-consuming procedures during the service creation, including the slice set up process.

We focus here on the procedure of *instantiation and configuration of VNFs* which is the major operation in the slice set up process. The problem of minimizing the time needed for instantiation and configuration of VNFs, is mapped to the Job Shop Scheduling Problem (JSSP), where each VNF is matched to an operation of a job. With this mapping, the target minimisation problem is translated to the minimisation of the runtime for all the jobs' operations defined in the mapped JSSP [3].

To resolve the above-mentioned problem (i.e., scheduling the VNFs instantiation and configuration procedure), a specific category of optimisation algorithms is examined, named metaheuristics. More precisely, we consider a system that receives requests for setting up network slices and provides a scheduling decision based on a nature-inspired metaheuristic approach by using genetic evolution characteristics.

In order to evaluate the performance of the above-mentioned algorithm, we performed simulations using well validated open datasets for the JSSP. From the evaluation process it became apparent that metaheuristic approaches can address efficiently the slice set up requirements of a 5G system and perform well in terms of service creation time.

The remaining of the paper is organized as follows. Section II contains a brief presentation on network slicing concept and a definition of the service creation time KPI. In section III, the adopted system model is presented, explaining how the target minimisation problem is mapped to the well-known JSSP. Next, section IV contains the proposed metaheuristic algorithm. Finally, in section VI, conclusions and insights for further study are provided.

II. Network Slicing

A. The concept of network slicing

The high-performance standards and the heterogeneous requirements that are set by emerging vertical services (e.g. Internet-of-Things, automotive communication), cannot be satisfied by the conventional/monolithic network deployments. As a countermeasure, network slicing is 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 various end-to-end logical networks, that: i) run on a common underlying (physical or virtual) network, ii) are mutually isolated, and iii) can have independent control and management [4].

The implementation of the network slicing concept is based on two key technologies, namely, the Software Defined Networks (SDN) and the NFV. On the one hand, SDN concept enables the decoupling between software and hardware in the network, thus it gives the opportunity to dynamically create and configure networks through software. On the other hand, NFV enables the virtualization of physical network functions such as firewall, encryption, routing etc., separating them from dedicated hardware and moving them as software to virtual or physical servers (transforms the network functions to VNFs). Consequently, a network slice can be considered as a set of VNFs that are hosted in compute nodes of a network infrastructure owner and offer an isolated and scalable virtual network to vertical service providers.

B. *Service creation time*

The organizations responsible for 5G standardization and development have set their goals concerning the 5G performance by providing a set of target values for major technical specifications or KPIs. According to the technical annex of the 5G PPP contractual arrangement [5], one of the target KPIs is the so called "*Reducing the average service creation time cycle from 90 hours to 90 minutes*". Service creation time is defined as "*Time required to provision a service, measured since a new service deployment is requested until the overall orchestration system provides a response*" [2].

Service creation time can be divided into many time-consuming phases, each of them depending on different parameters. In the attempt to identify the various phases of service creation, 5G-PPP Phase II projects have 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 summarized version of the timing flow is presented in TABLE I. We focus on the second phase, listed in TABLE I, i.e., the time required for the network slice to be created, named as "*Instantiate, Configure & Activate*". More specifically, we focus on optimizing the time for the instantiation and configuration of a network slice, by trying to find the optimal scheduling for setting-up VNFs in service chain.

TABLE I. SERVICE CREATION &ACTIVATION TIME

Creation & Activation Time		
Phase	**Name**	**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

III. SYSTEM MODEL

Let a 5G infrastructure manager that holds several processing machines, located in multiple data centers, on which VNFs can be hosted. We assume the following constraints:

- Chunks of requests for network service deployment are periodically received at the infrastructure manager.
- Each request includes the required execution plan of each service.
- The infrastructure manager, based on monitoring of past set ups and knowledge of the capabilities of the available processing machines, estimates the time needed for setting-up a VNF to one of the available machines.
- An optimisation algorithm is applied to schedule each chunk of requests, taking all constrains into account, and to export the total VNF execution plan.

To address the problem of selecting an optimisation algorithm for the process described in the last bullet point above, principles of scheduling are studied, as explained below.

A. *Mapping to known scheduling problems*

In the literature, the scheduling mainly deal with the challenge of selecting time slots for performing a set of activities, with respect to a given set of constrains (commonly constraints such as resources limitations or precedence order among the activities), targeting at optimizing a performance metric [6]. In the majority of the network related scheduling approaches, the optimisation metric is either the total processing time of the activities (the well-known make span minimisation [7]) 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 (the functional units that compose slices) to available processing resources. This mapping is based on the fact that, every network service is modelled as a service chain, i.e., a set of network functionalities, implemented as VNFs, in a specific order [8].

The Job-Shop scheduling sets a combinatorial problem-solving approach, used widely in literature to address resource allocation and scheduling problems [9, 10, 11, 12, 13]. Here we assume the Job-Shop scheduling with the following characteristics:

- let a set of n jobs, denoted as J_1, J_2, … J_n of varying processing times
- The jobs need to be scheduled on m machines, denoted as M_1, M_2, M_m
- Within each job there is a set of operations O_1, O_2, ..., O_i
- Each operation must be executed is a specific machine 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, applying to precedence and resource constraints of the system, to minimize the time in which all operations will be executed (find the 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, comprising a network slice.

More specifically, based on this mapping and the principles of the model presented in [13], we use the following notation:

- S for the set of network slices, $s_i, i \in [1, |S|]$
- F for the set of VNFs, $f_i, i \in [1, |F|]$
- H for the set of hosts, $h_i, i \in [1, |H|]$
- f_{ij} for the representation of the j^{th} VNF in the i^{th} network slice.
- T_{ij}^{pq} for the installation time of the q^{th} VNF to its corresponding host h_p

Thus, the minimisation of makespan is defined as:

$$MinMakespan = \min\left[\max\left[t_{ij}^{pq} + T_{ij}^{pq}\right]\right]$$

where max $[t_{ij}^{pq} + T_{ij}^{pq}]$ is the completion time of the last VNF among all network slices.

B. Algorithmic approach

Based on the mapping presented in the previous section, a resolving of the VNF scheduling problem can be provided by resolving the related JSSP [14]. The JSSP is known as an NP-hard (Non-Deterministic Polynomial-time) problem, meaning it cannot be solved in polynomial time [15]. Thus, it makes ineffective the use of exact methods, such as the Dynamic Programming, the branch 'n bound, etc., since they search exhaustively to 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 resolving methods fit well to the JSSP. Heuristic methods provide a near-optimal solution by constructing solutions according to greedy decisions [16]. Heuristics have some very interesting advantages. To name some, they work well for dynamic problem sizes [17]; they find a solution in reasonable time [9]; and they can be combined with other methods [18]. However, the 'greedy' nature of heuristics is not always preferable (since as the problem size increases, the near-optimal solution they provide tends to be even 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 brings an additional advantage. The execution time can be tuned through execution parameters that can change based on the input size [19]. 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 [20], power consumption [21] and data-mining [22]. According to [23], nature-based methods can be classified into four divisions: *Evolution-based, Physics-based, Swarm-based, and Human-based.*

Here, we adopt from the Evolution-based category a proven to be efficient and adaptive method, known as genetic or evolutionary method.

IV. The Proposed Algorithm

Genetic or evolutionary method, as the name implies, is based on Darwin's theory of evolution, adopting also some aspects from genetic science. Genetic Algorithm (GA) is an effective iterative 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.

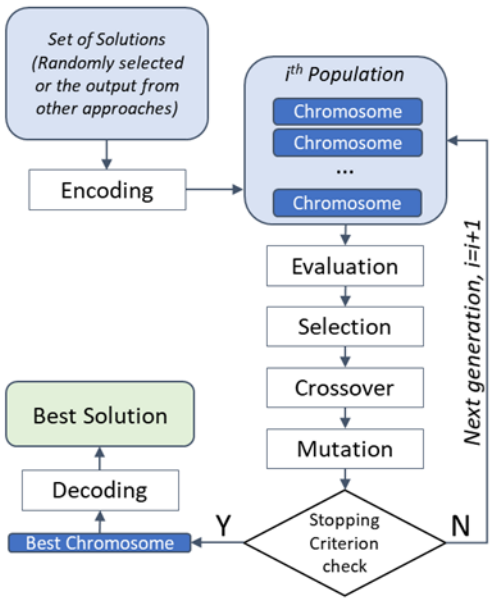


Fig. 1. Block diagram of the adopted approach

Encoding

Solutions of a prodder study or randomly selected ones are encoded as series of VNFs, referred here as ***chromosomes***. In a more abstract view, solutions are represented as arrays of VNFs or ***genes***, with specific order, as the example that follows.

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

Crossover

Crossover is a genetic operator analogous to biological reproduction, which aims at breeding new acceptable solutions from existing ones. This procedure, requires two VNF chains (chromosomes) referred to as 'parents'. First, for both 'parents' some points are randomly selected across their chain length and the chains are split into a number of parts. Then, a new chain of the same length is created by completing its parts from corresponding parts selected either from the one or from the other patent's chain. An example is presented in TABLE II.

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}	$f_{41}, f_{22}, f_{11}, f_{33}, f_{42}, f_{23}, f_{34}, f_{12}, f_{43}$	$f_{44}, f_{13}, f_{24}, f_{14}$

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.

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

Evaluation (Fitness function)

The process of Evaluation or Fitness function is used on evaluating the solutions produced from genetic operations. This evaluation happens by estimating the objective value that is chosen to be minimized, in every solution. This makes fitness function a problem-dependent issue. For JSSP, makespan is the objective value, that we have to minimize. In our implementation in order to correctly calculate fitness value for every scheduling solution, we had to take into consideration the set up time of the VNFs as well as the time periods that the hosts are available to set up the VNFs allocated to them.

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.

Having presented the major functional blocks of the proposed approach, the pseudo code of the proposed GA is presented below.

Algorithm: The Proposed Genetic Algorithm (GA)

```
1:  $initialPopulation, minimumMakespan \leftarrow$
    $ChooseInitialParents\ (schedulingPlanFile)$
2:  $define\ values\ for$:
3:      $crossover_{threshold}$,
4:      $mutation_{threshold}$,
5:      $numberOfGenerations$,
6:      $numberOfPopulation$
7:  $for\ par_x, par_y\ in\ initialPopulation$:
8:      $for\ m\ in\ [1, numberOfGenerations]$:
9:          $for\ n\ in\ \left[1, \frac{numberOfPopulation}{2}\right]$:
10:             $crossover_{possibility} \leftarrow random(0,1)$
11:             $if\ crossover_{possibility} \leq crossover_{threshold}$
12:                 $child_i, child_j \leftarrow crossover(par_x, par_y)$
13:             $mutation_{possibility} \leftarrow random(0,1)$
14:             $if\ mutation_{possibility} \leq mutation_{threshold}$
15:                 $child_i \leftarrow mutation(child_i)$
16:                 $child_j \leftarrow mutation(child_j)$
17:             $selectionList.append\begin{pmatrix} fitnessValue(child_i), \\ fitnessValue(child_j) \end{pmatrix}$
18:             $par_x \leftarrow child_i,\ par_y \leftarrow child_j$
19:         $sort(selectionList)$
20:         $if\ first\ element\ of\ selectionList \leq minimumMakespan$
21:             $minimumMakespan \leftarrow first\ element\ of\ selectionList$
22:         $par_x \leftarrow first\ element\ of\ selectionList$
23:         $par_y \leftarrow second\ element\ of\ selectionList$
24: $MinimumMakespan$
```

V. PERFORMANCE EVALUATION

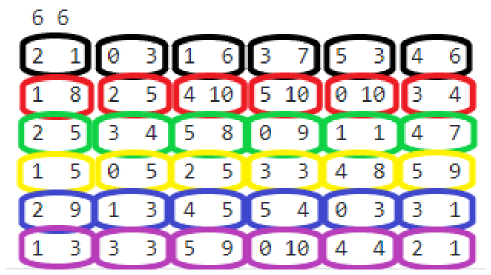
A. Simulations set up

In order to verify whether metaheuristic methods are efficient enough to be used on VNF scheduling, the proposed genetic algorithm is implemented on Python 3.6 and tested with benchmark instances formulated as it is exemplified in Fig. 2. More precisely, the format of every instance follows the rules listed below:

1) The first row defines the number of network slices requests and number of VNF hosts for the specific instance, e.g. "6 6".

2) Each row contains a set of pairs whose number is dependent on the number of VNFs of each network slice.

3) 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 e.g. "2 1" means that the VNF is hosted by machine 2 and its execution will last 1 time-unit.



```
6 6
2 1 0 3 1 6 3 7 5 3 4 6
1 8 2 5 4 10 5 10 0 10 3 4
2 5 3 4 5 8 0 9 1 1 4 7
1 5 0 5 2 5 3 3 4 8 5 9
2 9 1 3 4 5 5 4 0 3 3 1
1 3 3 3 5 9 0 10 4 4 2 1
```

Fig. 2. JSSP instance format

The instances are used as input to our Python program, among other important experimental parameters, such as population size, number of generations, crossover and mutation possibility. Firstly, the instance's data is translated to a more Python-friendly structure, using lists, tuples and dictionaries. Then, genetic algorithm is being executed on this structure, following the steps analyzed in section IV. The optimal plan of VNF scheduling is produced as output together with the makespan, the population size, the number of generations, the crossover and the mutation possibility, and the execution time of the algorithm.

For the simulations, the algorithm was executed on a Dell Optiplex 7050, equipped by Intel Core i5-6500 processor (3.20 GHz x 4), running Ubuntu 18.04 (Bionic Beaver) Desktop. The output was stored in a Mongo database running on the same host. MongoDB was a NoSQL, document database, which offers an efficient, simple and powerful interface with Python language (PyMongo).

Each evaluation result of the proposed algorithm was extracted after a numerous set of runs. Also, the algorithm was tested for a wide range of values for its configurable parameters. With no loss of generality, we adopted a 10x10 dataset, named “orb04” (from [24]), with a known minimum makespan equal to 1005 time-units.

B. Numerical results

We first examine whether and in what extend the number of generations and population size affect the provided makespan (slice set up time). We set the possibility that a genetic operation (crossover and mutation) is applied to 0.7. The choice of this value is based on other experiments, described below, showing that is the specific range of values, makespan is optimized. The results are depicted on Fig. 3.

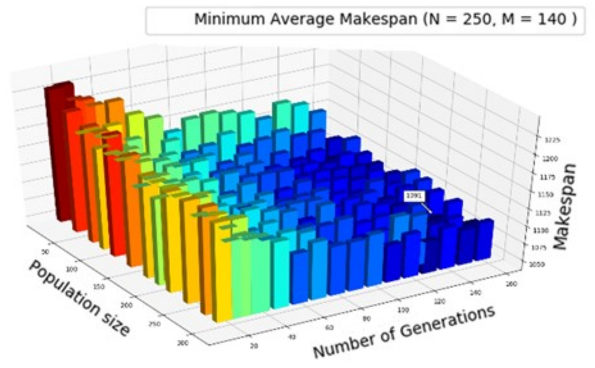


Fig. 3. Average service creation time for “orb04” (5 runs per generation-population size combination)

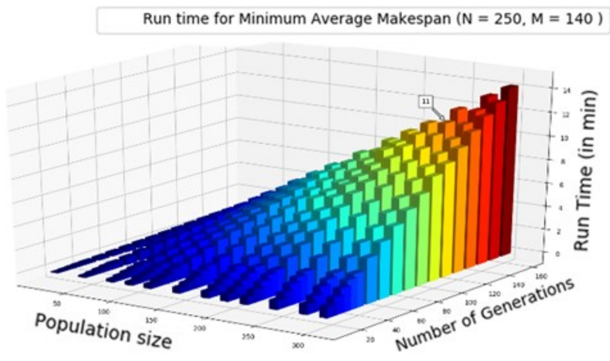


Fig. 4. Genetic algorithm run time for “orb04”

Reasonably, the higher the number of generations and populations, the higher the solution space that the algorithm examines. However, as depicted in Figure 3, as the number of generations and populations increases, shorter schedules are discovered, and optimal solution is approximated. In Fig. 4, we measure also the time in which every result is produced. According to the results, the optimal solution was found in 11 minutes using a common low-power computer as described in the previous section.

Another interesting aspect of the proposed approach is that the genetic operators can be applied wisely to improve the results. As resulted from our study, the efficiency of the genetic operators depends on their matching possibility, which is an input of the algorithm. In the simulated example, in order to verify that the possibilities of the genetic operators are applied beneficially in the set of solutions examined (e.g. if the mutation possibility is set to 0.5, 50% of the whole population size in all generations must have been through mutation process), we tested various combinations of generation and population number and the results are depicted on Fig. 5.

Consequently, taking the results of Fig. 5 into consideration, we examine whether genetic operators have impact on makespan searching in a set of solution composed by 40 generations and 100 children in each generation. According to Fig. 6, when the proposed algorithm is executed with higher values of mutation and crossover possibilities, makespan is decreasing (presented with even darker blue bar in the figure), thus optimizing scheduling.

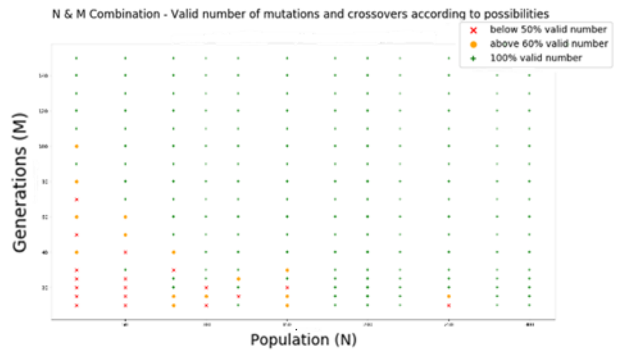


Fig. 5. Validation of genetic operators’ possibility in the dataset

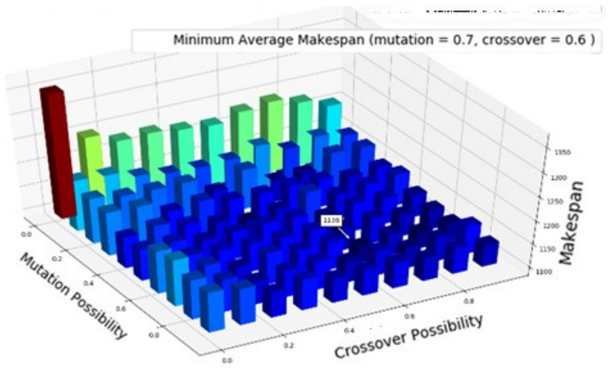


Fig. 6. Average service creation time for “orb04” (5 runs per crossover-mutation possibility combination)

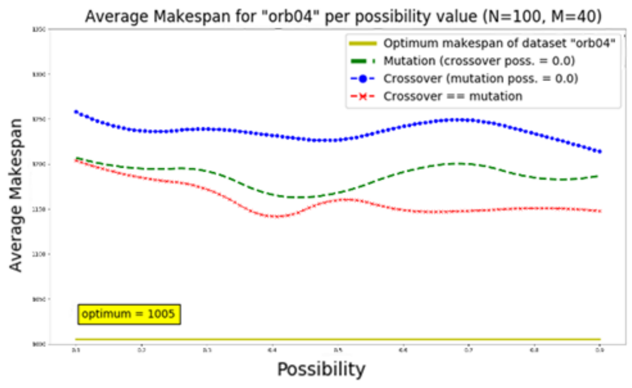


Fig. 7. Average service creation time for “orb04” per possibility value

One last, very interesting, result, is presented in Fig. 7. Mutation and crossover are compared on which one produces better results when the other has zero possibility to be implemented. The facts show that solely mutation gives way shorter makespan than crossover. However, the combination of them, brings even better results.

VI. Conclusions

Service creation time is one of the fundamental KPIs in 5G systems, but also one of the most complicated, as it can be affected by multiple computational or network parameters. Its optimisation can have a great impact on the overall system performance, and eventually to the quality of provider's network slice infrastructure services. Hence, in this paper, we presented a complete solution framework that implements a genetic algorithm, i.e., a meta-heuristic method, responsible to schedule the VNFs from various service requests in a 5G system. From the evaluation process revealed that meta-heuristic algorithms can efficiently contribute towards a real-time optimisation of service creation time. More precisely, the proposed genetic algorithm provided close-to-optimal result in adequately low processing time for the simulated example. Next steps in the direction of service creation time minimisation could possibly be the mapping of other meta-heuristic algorithms to the specific problem and their implementation/evaluation in real slice-enabled systems.

Acknowledgment

The work described in this paper is co-funded by the 5GENESIS project under the European Union's Horizon 2020 research and innovation programme (grant agreement No 815178) and also by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH – CREATE – INNOVATE (project CityZEN, code:T1EDK 02121).

References

[1] 5G-PPP, "View on 5G Architecture," 2019.

[2] F. Messaoudi and L. Valcarenghi, "5G-TRANSFORMER Project, Experimentation results and evaluation of achievements in terms of KPIs," 2019.

[3] J. F. Riera, E. Escalona, J. Batallé, E. Grasa and J. A. García-Espín, "Virtual network function scheduling: Concept and challenges," in International Conference on Smart Communications in Network Technologies (SaCoNeT), Vilanova i la Geltru, Spain, 2014.

[4] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca and J. Folgueira, "Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges," IEEE Communications Magazine, May 2017.

[5] 5G-PPP, "Contractual Arrangement," 2013.

[6] P. Brucker and S. Knust, Complex Scheduling, Springer, 2012.

[7] Anshulika and L. A. Bewoor, "A Genetic Algorithm approach for solving a Job Shop," in International Conference on Computer Communication and Informatics, Coimbatore, India, 2017.

[8] L. Ruiz, R. J. Durán, I. De Miguel, P. S. Khodashenas, J.-J. Pedreno-Manresa, N. Merayo, J. C. Aguado, P. Pavon-Marino, S. Siddiqui, J. Mata, P. Fernández, R. M. Lorenzo and E. J. Abril, "A Genetic Algorithm for VNF Provisioning," Applied Sciences, December 2018.

[9] X. Xi, L. Jiang and Q. Zhang, "Optimization for Multi-Resources-Constrained Job Shop Scheduling Based on Three-level Heuristic Algorithm," in International Asia Conference on Informatics in Control, Automation and Robotics, Bangkok, Thailand, 2009.

[10] W. Wu, J. Wei and X. Guan, "A hybrid algorithm for scheduling in job shop problem with flexible resources," in International Conference on Control and Automation (ICCA), Xiamen, China, 2010.

[11] C. Kurera and P. Dasanayake, "New Approach to Solve Dynamic Job Shop Scheduling Problem Using Genetic Algorithm," in 3rd International Conference on Information Technology Research (ICITR), Moratuwa, Sri Lanka, 2018.

[12] 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 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall), Honolulu, USA, 2019.

[13] Q. Li, X. Wang, T. Zhao, Y. Wang, Z. Li and L. Rui, "An Improved Genetic Algorithm for the Scheduling of Virtual Network Functions," in 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), Matsue, Japan, 2019.

[14] Z. Shen and L. Smalov, "Comparative Performance of Genetic Algorithm, Simulated Annealing and Ant Colony Optimisation in solving the Job-shop Scheduling Problem," in 26th International Conference on Systems Engineering (ICSEng), Sydney, Australia, 2018.

[15] M. Garey, D. Johnson and R. Sethi, "The Complexity of Flowshop and Jobshop Scheduling," Mathematics of Operations Research, vol. I, pp. 117-129, 1976.

[16] A. Mungwattana and K. Ploydanai, "Future Makespan Heuristic for job shop scheculing problem," in The 40th International Conference on Computers & Indutrial Engineering, Awaji, Japan, 2010.

[17] J. Cui and T. Li, "A Hybrid Heuristic Neighborhood Algorithm for the Job Shop Scheduling Problem," in 2008 Fourth International Conference on Natural Computation, Jinan, China, 2008.

[18] 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 2nd International Conference on Measurement, Information and Control, Harbin, China, 2013.

[19] C.-W. Tsai and J. J. P. C. Rodrigues, "Metaheuristic Scheduling for Cloud: A Survey," IEEE Systems Journal, vol. 8, no. 1, pp. 279 - 291, 2014.

[20] S. Kumar, S. Mittal and M. Singh, "Metaheuristic based workflow scheduling in cloud environment," in 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2016.

[21] G. Phatai, S. Chiewchanwattana and K. Sunat, "A Comparative of Neural Network with Metaheuristics for Electricity Consumption Forecast Modelling," in 22nd International Computer Science and Engineering Conference (ICSEC), Chiang Mai, Thailand, 2018.

[22] W. Liu and J. Wang, "A Brief Survey on Nature-Inspired Metaheuristics for Feature Selection in Classification in this Decade," in IEEE 16th International Conference on Networking, Sensing and Control (ICNSC), Banff, AB, Canada, 2019.

[23] R. Rajakumar, P. Dhavachelvan and T. Vengattaraman, "A survey on nature inspired meta-heuristic algorithms with its domain specifications," in International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2016.

[24] [Online]. Available: https://github.com/tamy0612/JSPLIB. [Accessed July 2019].