Adaptable and Data-Driven Softwarized Networks: Review, Opportunities, and Challenges

Wolfgang Kellerer* Patrick Kalmbach* Andreas Blenk* Arsany Basta*
Martin Reisslein† Stefan Schmid‡
* Technical University of Munich, Germany †Arizona State University, Tempe, AZ, USA
‡University of Vienna, Austria

***Abstract*—Communication networks are a key enabling technology for our digital society. In order to sustain their critical services in the future, communication networks need to flexibly accommodate new requirements and changing contexts due to emerging diverse applications. In contrast to traditional networking technologies, software-oriented networking concepts, such as Software-Defined Networking (SDN) and Network Function Virtualization (NFV), provide ample opportunities for highly flexible network operations, enabling fast and simple adaptation of network resources and flows. This paper identifies opportunities and challenges of adaptable softwarized networks and introduces a conceptual framework for adaptations in softwarized networks. We first explain how softwarized networks contribute to network adaptability through the functional primitives observation, composition, and control. We review the wide range of options for fine-granular observations as well as fine-granular composition and control provided by SDN and NFV. The multitude of fine-granular "tuning knobs" in adaptable softwarized networks complicates the decision making, which is the main focus of this article. We propose to enhance the functional primitives observation, composition, and control with data-driven decision making, e.g., machine learning (ML) modules, resulting in *deep* observation, composition, and control. The data-driven decision making modules can learn and react to changes in the environment, e.g., new flow demands, so as to support meaningful decision making for adaptation in softwarized networks. Finally, we make the case for employing the concept of empowerment to realize truly "self-driving" networks.**

***Index Terms*—Data-Driven Networking, Empowerment, Machine Learning (ML), Network Function Virtualization (NFV), Self-Driving Networks, Software-Defined Networking (SDN).**

I. Introduction

A. Motivation: Need for Flexible Network Adaptation

Today's communication networks are continuously exposed to new contexts and have to react to new demands. Large-scale software updates, user streaming of social mega events, overnight popularity or sudden drops in popularity of recently introduced apps contribute to unprecedented dynamics in terms of changing traffic patterns and resource demands. Given the critical role that communication networks play in our digital society, it is important to account for and accommodate such dynamics. In other words, communication networks are required to be flexible and to react, supporting fast and simple adaptations of the network resources and flows. Emerging software-oriented networking paradigms, and in particular Software-Defined Networking (SDN) [1]–[3] and Network Function Virtualization (NFV) [4], [5], operating

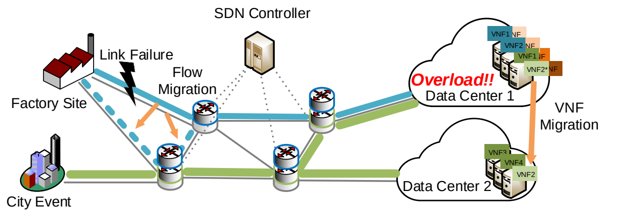


Fig. 1: Example illustration of softwarized network scenario that combines SDN and NFV technologies. *Upper part:* A central SDN controller directs traffic to particular network services upon demand, and sets up a new path (*dashed*) when a physical link fails. *Lower part:* Network services are composed of (virtualized) network functions in different data centers. When one of the data centers becomes overloaded (e.g., due to a mega event), some functions are migrated to another data center to ensure load balancing.

either individually or in combination, promise to provide such flexibilities. SDN and NFV unlock flexibilities through a new level of indirection as well as new interfaces for programming the control plane and for setting up virtual network functions (VNFs) and network flows on demand [6], [7].

We illustrate some example opportunities that are provided by softwarized networks in Fig. 1. To support the connectivity, an SDN-based controller configures the respective flows at runtime and steers the traffic towards the VNFs in a datacenter. Fig. 1 shows two flows: one from a factory site (blue line) towards a network service composed of VNFs in the upper right data center, for some real-time data processing; and one from a mega event in the city (green line). Upon a link failure in the network infrastructure, an adaptation is triggered, i.e., the migration of the factory related flow (dashed blue line), to maintain connectivity. Also, in case the upper right data center becomes overloaded, one of the running VNFs is migrated to the bottom right data center. More generally, softwarized networks require a flexible adaptive VNF lifecycle management that encompasses the initial placement of the VNFs, the migration of VNFs to ensure continuous scalable service, as well as mechanisms for VNF fault tolerance and the coordinated tear-down of VNFs that are no longer needed.

In general, network adaptation involves three main phases [8]–[10]: (1) detection of an event that requires adaptation, e.g., change in the environment, such as traffic conditions, or receipt of a request for a new network function chain, (2) decision on what and how to adapt, e.g., find an optimal placement for a function chain, and (3) execution of the adaptation, e.g., migrating functions and steering flows through them.

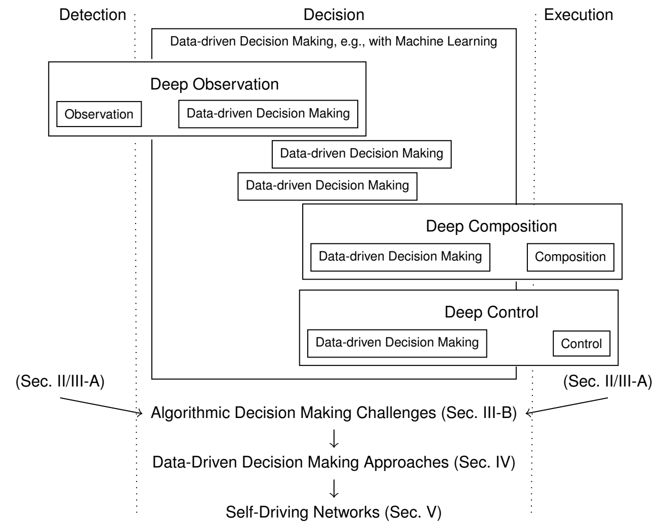

Detection
Decision
Execution
Data-driven Decision Making, e.g., with Machine Learning
Deep Observation
Observation
Data-driven Decision Making
Data-driven Decision Making
Data-driven Decision Making
Deep Composition
Data-driven Decision Making
Composition
Deep Control
Data-driven Decision Making
Control
(Sec. II/III-A)
(Sec. II/III-A)
Algorithmic Decision Making Challenges (Sec. III-B)
Data-Driven Decision Making Approaches (Sec. IV)
Self-Driving Networks (Sec. V)

Fig. 2: Conceptual framework for adaptation in softwarized networks: Adaptation commonly comprises the three phases detection, decision and execution. The detection of events, e.g., a sudden traffic increase, triggers adaptations. The decision phase needs to answer the questions of what and how to adapt in the network. Adaptation tasks are carried out in the execution phase through Network Function Virtualization (NFV), and control actions, e.g., change the flow steering through Software Defined Networking (SDN). SDN and NFV provide functional primitives for observation, composition, and control that mainly map to the detection and execution phases. The fine-grained observation information as well as functions and "knobs" available with SDN and NFV result in highly complex decision making, which calls for data-driven approaches. This article is motivated by the insight that the enhancement of the functional primitives with data-driven concepts leads to deep observation, composition, and control.

B. Functional Primitives for Adaptation in Softwarized Networks: Observation, Composition, and Control

Adaptation is a key feature of SDN and NFV based softwarized networks to accommodate dynamic changes, as illustrated in the example in Fig. 1. For adaptation of a communication network, we not only consider the control of the connectivity and transport of traffic flows across the network, but also the control of the storage and processing capabilities in network, including data center nodes that support the composition of network functions [4], [5].

Adaptations in softwarized networks mainly comprise the *observation* of events, e.g., overload situations, as well as the *composition* and the *control* of network resources [1]–[3]. We denote these three adaptation components as the three functional *primitives* of adaptation in softwarized networks.

The softwarized network technologies that we consider here, namely SDN and NFV, contribute differently to these primitives. Both provide opportunities for data collection through new interfaces and partially centralized views of the network, providing the observation primitive. The main focus of NFV is on adaptation through composition. Network functions can be (de-)activated, configured, placed, composed (and decomposed), migrated, and chained [4], [11]. Control-related adaptation is, on the other hand, rather attributed to SDN. SDN provides a fine-granular view of the flows, thus supporting opportunities for flow steering (e.g., to compose VNFs into more complex services) and traffic engineering [5].

C. Mapping Adaptation Phases (Detection, Decision, and Execution) to Functional Primitives in Softwarized Networks

The functional primitives of softwarized networks (observation, composition, and control) map in a straightforward manner to the general adaptation phases (detection, decision, and execution), as illustrated in Fig. 2. The observation primitive supports the detection phase, while the composition and control primitives support the execution phase. Moreover, the fine-grained observation options provided by SDN and NFV, as well as the fine-grained SDN and NFV composition and control options lead to a highly complex decision making. Hence, new approaches are required for meaningful and efficient decision making in softwarized networks.

The example in Fig. 1 illustrates the adaptation phases: observations can relate to the current demands and link status; this information can then be accounted for in the execution phase where the composition primitive can compose NFVs in new ways and migrate NFVs, while the control primitive can re-route flows. The decision making can be very challenging as it concerns several combinatorial problems which are known to be hard, e.g., when and how to compose and to migrate the VNFs, as well as how to re-route and to traffic engineer the flows. We will elaborate on these and related computationally hard problems in Section III-B, and make the case for attractive solutions based on data-driven approaches.

D. Enhancing Functional Primitives in Softwarized Networks with Data-Driven Decision Making

1) Overview: We believe that data-driven approaches are well suited for meaningful and efficient decision making in softwarized networks. We propose to combine the observation functional primitive of SDN and NFV with data-driven decision making, e.g., ML based decision making, so as to give a "deep observation" functionality that spans the detection phase and a part of the decision phase, see Fig. 2. The "deep observation" functionality should be designed to optimally support network adaptation decision algorithms. Moreover, we propose to combine the composition and control functional primitives of SDN and NFV with data-driven decision making to give "deep composition" and "deep control", see Fig. 2, so as to accomplish meaningful network adaptation.

2) From SDN/NFV Observation to Deep Observation: The observation functional primitive provided by SDN and NFV supports the detection phase by enabling the collection of unprecedented amounts of fine-grained observation data. In particular, SDN and NFV provide tools for the collection of network observation data, ranging from the equivalent of binoculars for long range observations to microscopes for fine-grained local observations.

A critical part of the decision making is to optimize the extraction and processing of the observation data. *Deep observation* combines the SDN and NFV observation functional primitive with data-driven decision making modules. The goal of deep observation is to collect data to detect and interpret an emerging new situation so as to support the decision making and optimization of the execution actions that are carried out by deep composition and control. The decision making modules of deep observation have to select and extract the most useful data for supporting network adaptation decisions, e.g., through ML based dimensionality reduction of relational network data, as elaborated in Section IV-B1.

3) From SDN/NFV Composition and Control to Deep Composition and Deep Control: SDN and NFV provide composition and control functional primitives for executing network adaptations. More specifically, within the adaptation capabilities provided by SDN and NFV, the decision making needs to select the specific optimal adaptation mechanisms, i.e., the optimal composition and control actions, as well as the corresponding optimized composition and control parameter settings. *Composition* refers to the configuration of the network functions and the related network resources. Composition includes function (de-)composition, chaining, placement, and configuration, as well as the configuration of the network topology [4], [5]. In particular, composition includes all functionalities that are accomplished through packet processing in physical or virtual computing elements, such as multi-access edge computing. *Control* refers to instructing the data plane elements how to process packets and how to steer packets and flows. In particular, control encompasses all functionalities that are accomplished through direct instructions to data plane elements, such as OpenFlow (OF) switches, specifying prescribed actions on packets [12], [13].

The vast range of fine-grained composition and control capabilities enabled by SDN and NFV in conjunction with the deep observation capabilities pose highly complex novel optimization problems that need to be solved in the decision phase. That is, the decision phase is critical for effective composition and control in softwarized networks. We propose to combine data-driven decision making modules, e.g., data-driven ML modules, with the composition and control functional primitives of SDN and NFV leading to *deep composition* and *deep control*. More specifically, the decision phase in the middle of Fig. 2 will typically feature several data-driven decision making modules. Some of these modules are not directly associated with the observation, composition, and control functional primitives, such as modules on general long-term network infrastructure planning or other higher-order decision making. On the other hand, there will be several modules that make decisions regarding the composition and control functional primitives. For instance, data-driven ML decision making modules can enhance composition and control by predicting the feasibility of VNF or routing requests and by reducing the search spaces for composition and control decisions (see Section IV-B2). We propose to combine these data-driven decision modules with the composition and control functional primitives to form deep composition and deep control, as illustrated in Fig. 2.

We briefly note that adaptive softwarized network operation involves continuous cycling through the deep observation as well as deep composition and deep control. In particular, the effects of deep composition and control actions are monitored through the deep observation to inform the decision making for the next adaptation steps.

E. Towards Data-Driven Empowered Networks

SDN and NFV do not only improve adaptation, e.g., through resource and flow steering, but also open up new dimensions of data collection and global views. Such data and views can be used to further improve network management, and in particular to devise novel algorithms "bypassing" computationally hard network optimization problems. In other words, the availability of new data sets through softwarized network technologies is not only beneficial for adaptation in general, but also enables fundamentally new approaches for managing, operating, and hence adapting networks. We envision that data-driven softwarized networks integrate the knowledge of the past for faster and more efficient executions of network algorithms in the future. The use and processing of the increasing amount of new data poses novel challenges, e.g., how to efficiently represent data and how to efficiently implement the data gathering process.

Taking the data-driven approach for supporting the adaptation of softwarized networks one step further, we arrive at the currently emerging paradigm of self-driving networks [14]. Self-driving networks measure, analyze, and control themselves in an automated manner, even without a pre-defined objective function. The concepts for such a self-driving network operation are in the early development stages and require extensive future research. We propose the concept of *empowerment* as a promising feature for self-driving networks.

F. Article Structure

Fig. 2 illustrates the structure of this article. Section II elaborates on the observation, composition, and control functional primitives of SDN and NFV. These SDN and NFV functional primitives are employed in the detection and execution phases of network adaptations. Section III-A reviews the adaptation potentials of softwarized networks based on SDN and NFV along with state-of-the-art adaptation examples. Section III-B discusses the challenges that arise in the decision phase that links detection (observation) with execution (composition and control) in softwarized networks. Section IV gives an overview of recently emerging new ML concepts that can be employed as data-driven decision making modules in deep observation, deep composition, and deep control. We emphasize that this article is *not* concerned with enhancing the SDN/NFV capabilities. Instead, our main focus is on the judicious use of the SDN/NFV capabilities through the combination of the SDN/NFV capabilities with data-driven decision making modules. Section V discusses the forward looking concept of future empowered, intelligent data-driven networks. Section VI concludes this article outlining further open research challenges.

II. Enablers of Adaptable Softwarized Networks: Functional Primitives of SDN and NFV

This section provides a brief overview of key concepts that enable high levels of flexibility and adaptability in softwarized networks, in particular, Software-Defined Networking (SDN) and Network Function Virtualization (NFV). The novel abstraction and generalization introduced by SDN and NFV, provide extensive opportunities for adaptation. Leveraging a combination of SDN and NFV further enhances the adaptation potential.

As introduced in the previous section, we identify three main SDN and NFV functional primitives that enable fine-grained adaptations: network observation, network composition, and network control.

A. Software Defined Networking

1) Overview: At the heart of SDN lies the idea of outsourcing and consolidating control over data plane elements (e.g., OF switches) to a logically centralized software controller. With SDN, a controller configures the forwarding behavior (e.g., forwarding tables) of data plane elements *directly*, through a standardized interface (API) [12], [13], [15], [16]. The forwarding tables of the switches store a set of *match-action rules*: the match part of the rule is defined over packet headers (e.g., specific header fields, but it may also be protocol field independent), and the action part of the rule defines what to do with the packets matched by a certain rule, e.g., forward to a specific port, drop, or forward to another table for additional classifications.

This decoupling of control from the data plane introduces a novel abstraction and provides novel opportunities for adaptation. Perhaps most importantly, SDN enables the evolution of the control plane independently from the constraints and life-cycles of the data plane [17]. This comes with two main benefits:

- *Faster innovation:* software often outpaces hardware in terms of innovation speed. SDN makes it possible to quickly replace and update control plane logic.
- *Specialization:* SDN makes it possible to flexibly *tailor* control logic to specific organizational needs. Many organizations, such as Google [18], have specific control plane requirements that are not readily provided by existing (legacy) equipment that has the control plane integrated with the data plane.

The SDN adaptation functionalities fall mainly into the primitives of observation and control of the data plane. Whereas the composition primitive is mostly restricted to the SDN control plane.

2) Observation: Access to fine-grained insights into network configurations and operations is a pre-requisite for softwarized network adaptations. The de facto standard protocol for interfacing data plane nodes with the SDN control plane is the OF protocol [12]. The OF protocol can be used to observe behaviors, e.g., of packets as well as flows and their performance, and to collect data from the data plane. For instance, OF provides the interface to collect network statistics with several granularities, e.g., flow as well as switch port statistics on individual or group basis. The entities observed by SDN, such as switch ports, queues, and flows, can be abstracted through information models, e.g., graph-oriented database constructs, as elaborated in Section IV-B1.

Importantly, due to the centralized nature of SDN control, the collected network observations can be centrally viewed and analyzed. This centralized perspective can further simplify the detection of events, such as failures, and can serve as a basis for optimizing the network. Generally, the (logically) centralized SDN control enables fast adaptation: with a centralized view, it is possible to overcome inefficiencies, e.g., from distributed reconvergence after failures, and to predefine different algorithms for reacting to events in the data plane through fine-grained control.

3) Composition: The SDN composition primitive is mainly focused on the SDN control plane. A logically centralized SDN control plane can be formed in a scalable manner through interacting distributed controllers. The composition of the distributed SDN controllers providing the logically centralized SDN control functionality can adapt in response to varying loads or application demands. One aspect of the composition is to operate a meaningful number of SDN controllers, through control plane adaptation, to avoid overload and hence performance degradation and to provide a judicious load balancing so as to achieve scalability [19].

Another aspect of the composition is SDN controller placement. To meet performance requirements, such as control plane delay constraints, distributed SDN controllers can be strategically added, removed, and placed in the underlying network substrate, close to the data plane events (and data plane hot spots) that these controllers handle.

Availability can be further improved through clustering: distributed SDN controllers can be clustered and mapped to data plane nodes to meet resilience and efficiency targets [20]. For example, multi-controller composition assigns multiple distributed SDN controllers to a given data plane node to compose the SDN control functionality. Clearly, more controllers may also increase synchronization overhead among controllers, which can be considered as the cost of adaptation. This needs to be accounted for when determining the number of controllers and their placement.

The SDN control plane composition and controller placement can be adapted dynamically to react to the current network status or emerging events, such as a sudden traffic increase or a change of the flow distribution in the traffic mix. As a result, controllers may be migrated at runtime so as to implement optimal placement decisions.

4) Control: The main focus of the SDN functionalities is on the control of data plane adaptations. Unlike traditional switches whose behavior can only depend on Layer-2 header fields, an OF switch can match Layer-2, Layer-3, and Layer-4 header fields, or may in the future go beyond the existing header field structure entirely. In other words, the distinction between layers becomes blurry, and network management tasks, such as traffic engineering, become more flexible: e.g., packets may be forwarded depending on the application data they carry; `http` packets may be forwarded to a different port (e.g., toward a web cache) than `ftp` traffic headed towards the same IP destination address.

In traditional communication networks, flows are usually defined by their destination address (e.g., IP routing), a combination of source and destination addresses (e.g., oblivious routing), or an MPLS label, using pattern matching over the header. In contrast, OF makes it possible to arbitrarily tune the granularity of rules, from very fine-grained, e.g., microflows for specific combinations of IP addresses and TCP ports, to very coarse-grained, e.g., using wildcards over the header fields [21]. Moreover, OF offers the possibility to define whether certain flows should be handled rather proactively or reactively, introducing further flexibilities but also overheads [22].

Traditional traffic engineering is rather *indirect*. Traditional control planes are typically limited to shortest-path routing according to the link weights in the network. The only way to influence routing paths and to perform traffic engineering is through changing these link weights, which is a rather indirect way of influencing routes. In contrast, with SDN, paths can be influenced directly, by defining the forwarding rules. In fact, the resulting paths do not have to be shortest paths and do not even have to be loop-free.

Given these opportunities to react and adapt to new requirements, it is not surprising that the problem of how to reconfigure (i.e., to update and adapt) SDN networks quickly at runtime has been studied extensively over the last years [22]. In Section IV-B2, we describe how machine learning can enhance the control of networks to achieve deep control, e.g., through reactive flow rules that dynamically adapt the network control.

B. Network Function Virtualization

1) Overview: Network Function Virtualization (NFV) is an enabler towards highly adaptable and scalable networks responding to dynamically changing needs of network services and customers. Whereas current network function deployments rely on hardware middleboxes, NFV promotes the deployment of VNFs that are instantiated and operated on commercial off-the-shelf (COTS) hardware. As for SDN, moving from hardware to software-based implementations not only enables faster innovation, but also more efficient network resource utilization. In particular to achieve higher resource utilization, the NFV concept, as introduced by the European Telecommunications Standards Institute (ETSI) [23], [24], virtualizes hardware and networking resources, i.e., computing, storage, and networking resources of commodity hardware, e.g., provided by data centers.

One widely accepted architecture view of NFV is promoted by ETSI. The ETSI architecture consists of three building blocks: the Network Function Virtualization Infrastructure (NFVI), the VNFs, and the management and orchestration block (NFV MANO). The main driver for adaptability and scalability is provided by the NFVI, which virtualizes processing (CPU), storage, and networking resources. The NFVI provides the capability to operate VNFs on virtualized hardware; hence, it inherits all the advantages from virtualization, i.e., flexible provisioning, scaling, and migrating, while not relying on specific hardware implementations.

NFV has many potential applications [25] that mainly fall into the adaptation primitives of observation and composition.

2) Observation: Softwarization not only provides a given set of interfaces for monitoring VNF instances, but also enables fast ways of adding additional interfaces to further improve observability. Deep observation enabled by NFV provides access to an extensive pool of fine-grained observational data that has not been previously available with fixed hardware monitoring and event detection implementations. VNFs facilitate the access to fine-grained network function data and parameters, e.g., through benchmarking metrics for VNFs and the monitoring of NFVIs. These fine-grained network function observations can be exploited to improve the network function implementation, operation, and adaptation.

Moreover, considering the commodity server resources, hypervisors can report CPU utilization or network throughput at runtime—information that can be used to quantify the performance of VNFs that are deployed on virtualized commodity servers. Such information can help trigger adaptations, which are executed by VNF orchestrators.

3) Composition: Mobile network functions, including the Mobile Core Network and the Radio Access Network functions, are a prominent example benefitting from the fine-grained composition of network functions. Hardware functions, e.g., of the LTE Mobile Core Network, i.e., of the Evolved Packet Core (EPC), can be put into software according to their 3GPP standardization [26]. For instance, the Mobility Management Entity (MME) and the Serving and PDN-Gateways (S/P-GW) functions can be readily softwarized. The MME and S/P-GW functions provide time-critical adaptations in mobile networks: the MME manages the mobility of users, while the S/P-GWs forward and control user data traffic. By virtualizing these functions, operators gain the ability to scale their networks according to traffic fluctuations, e.g., daily user traffic patterns [27], [28]. Further promising use cases for mobile network function compositions are, for example, the softwarization of content distribution networks (CDNs) or set top boxes for multimedia applications.

In addition to the abstraction of network functions from the underlying hardware through virtualization, one important generalization that comes with NFV is the generalization of a given network function itself. This means that a given network function can be composed of multiple VNFs, where each VNF is implemented and deployed in a tailor-made manner to specifically support its performance needs. Therefore, a judicious decomposition of network functions into potential components is an important and often neglected task in NFV. Returning to the example of a Mobile Core Network, the S-/P-GWs require control plane and data plane related functions to forward and control user traffic. A judicious decomposition can support the separate placement of decomposed functions to meet two main objectives: enforcing latency constraints while exploiting the resource efficiency of VNFs [29]. The composition of VNFs is referred to as (service) function chaining. The decomposition of functions into numerous components enables tailor-made service adaptations as well as the fine-grained orchestration of networking resources. Ideally, an optimal function chain should connect all VNFs in the correct order. As SDN is a candidate technology to set up the flow control for such chains, we elaborate in Section II-C on the combination of NFV and SDN.

The composition provided through NFV can generalize the packet processing. While VNFs need to be compliant with standardized network functions, NFV opens further ways to process network packets. For instance, while hardware implementations for tunneling match only on pre-specified fields, software implementations can extend matches to arbitrary header fields. Beside this matching aspect, new optimization and adaptation opportunities arise with NFV: technologies such as Intel DPDK [30] or P4 [31] make it possible to accelerate existing software-based VNFs with hardware-close implementations, still compatible with COTS hardware. While recently introduced technologies, such as Intel DPDK, rely on closed hardware, there are new initiatives, such as Stratum [32], that seek to combine the P4 capabilities with open source hardware. This delivers not only the abstraction capabilities of P4, but also provides packet processing pipelines that are tailor-made for network functions. Accordingly, NFV opens the range of network function implementations from pure-software to software implementations accelerated with hardware.

Moreover, composition can generalize how software implementations use hardware resources. That is, computing capacity, storage, and network resources can always be used arbitrarily by any VNF—the general advantage of virtualization. Moreover, the use of resources can be adapted over time; the composition concept can simply tear down and bring up VNFs to support different networking services on a given server infrastructure.

A further generalization is location-independence. VNFs are not bound to specific physical locations anymore; instead, they can be deployed anywhere by the VNF orchestrator according to current demands. Having been deployed, VNFs can also be quickly adapted, even at runtime, again due to their software implementation nature.

Generally, with the migration from hardware to software-based implementations, and the deployment on virtualized infrastructures, it is expected that NFV will lower the cost not only of equipment (capex) but also the operation cost (opex) due a more fine-granular resource usage. In any case, VNFs should only rely on and make use of as much server (CPU, storage) and networking resources (link capacities) as currently needed.

C. Combination of NFV and SDN

Both, SDN and NFV rely on the concepts of virtualization and softwarization. Their combination opens further opportunities for adaptation and offers additional design options for highly adaptive softwarized networks. SDN is a key enabler for the realization of adaptive traffic routing for function chains composed by NFV. On the other hand, the control opportunities provided by SDN can be leveraged to realize network functions directly on the SDN data plane.

1) Composition and Control—Chaining: Using SDN in an NFV-based infrastructure can leverage NFV's full potential. For instance, network traffic can be steered to traverse multiple network functions to compose novel and innovative network services: e.g., firewalls, network address translation (NAT), and deep packet inspection (DPI). As discussed above, such function chains can also emerge from a judicious decomposition of a larger network function.

In any case, in order to realize such chains, a routing path has to be set up. The routing path has to traverse all functions in the correct order and has to fulfill all imposed requirements. In the context of this composition, SDN provides the way to accomplish flexible and adaptive routing through multiple network functions.

2) Composition and Control—Push Function to Data Plane: Combining SDN and NFV introduces new design opportunities; hence, it further increases adaptability. For instance, network functions can also be directly implemented through SDN by integrating the control logic in the SDN controller and pushing the VNF functionalities to the data plane. Consider for example the implementation of a stateless firewall. A pure VNF implementation places a firewall as a VNF in a data center; all traffic is then routed to the data center for the firewall VNF processing. However, firewall rules can also be directly implemented on the SDN data plane nodes by installing respective OF rules to block the traffic that matches prescribed conditions, e.g., prescribed port numbers. In this way, the SDN controller effectively pushes the firewall functionality onto the SDN data plane nodes. Based on the fine-grained rule sets applied by an SDN controller the firewall realization option can vary for different traffic flows [33].

This SDN realization of VNF functionalities, i.e., the pushing of functionalities by the SDN controller to the data plane nodes, is not limited to a static realization. Instead, the pushed functionalities can be differentiated and dynamically adapted. For instance, SDN control can be dynamically used to steer the part of the traffic that matches a prescribed set of header fields to the data center for the firewall VNF processing, while the firewall functionality for the other part of the traffic is directly implemented on the data plane. Hence, the SDN control offers extended adaptability for the implementation of network functionalities. This also offers the possibility to adapt the network design according to changing performance requirements, e.g., bringing network functionalities closer to the user on data plane nodes for lower latency or steering traffic to VNFs in case more processing (CPU) is needed.

3) Composition and Control—Hardware Programming: Novel concepts, such as programmable data planes, support the programming of packet processing hardware and provide additional design options for adaptation. For instance, P4 provides an abstract model for protocol independent programming of the network data plane. P4 can define packet headers and specify packet parsing and processing behaviors that extend the adaptability of SDN data plane devices and VNF data plane pipelines.

The combination of SDN and programmable data planes provides the opportunity to adapt not only the control rules/policies of the data plane traffic but also the data plane pipeline processing behavior itself. This extended adaptability can be achieved through different approaches. For example, one approach is to extend the OF controller with a P4 plugin. The P4 plugin enables the controller to work in conjunction with P4 targets to populate tables and handle new packets. An example for such an approach is the activity of OpenDaylight[1]. Another approach is to include an OF agent in the P4 data plane device which acts as a controller facing agent that is

[1]https://wiki.opendaylight.org/view/Project_Proposals:P4_Plugin

TABLE I: Examples of adaptations with their corresponding support by the adaptation enablers for softwarized networks, i.e., SDN and NFV, and their relation to the three functional primitives of adaptation. (•: main target).

	Enabler(s)		**Primitive(s)**		
Adaptation (see Sec. II)	**SDN**	**NFV**	**Obs.**	**Comp.**	**Ctl.**
Monitoring	•	•	•		
Event detection	•	•	•		
Adaptive measurements	•	•	•		
Function configuration		•		•	
Fct. cfg. (push to SDN node)	•			•	•
Fct. res. dimen. (scale)		•		•	
Add/remove f.ct./ctl. (scale)	•	•		•	
Function placement		•		•	
Controller placement	•			•	
Function (de-)composition		•		•	
Function chaining		•	•	•	•
HW / SW configuration	•	•	•	•	•
Flow steering (direct rout.)	•				•
Traffic engineering	•				•
Rule/policy adaptation	•	•		•	•
Consistent network update	•		•		•
Admit and embed services and virtual networks	•	•	•		•

responsible for the mapping between OF and P4 [2]. Such data plane programming and the integration with SDN control extend the composition and control functional primitives of softwarized networks to offer a wide range of adaptation opportunities.

III. Adaptation Potentials and Challenges of SDN and NFV

This section presents overviews of the potential adaptation benefits enabled by SDN and NFV as well as the corresponding challenges. We organize the adaptation potentials and challenges according to our three functional primitives for adaptation: Observation, composition, and control. Table I gives an overview. We highlight these aspects with selected examples from state-of-the-art adaptive networks; we do not aim to provide an exhaustive survey. General surveys of SDN and NFV based concepts (not necessarily focused on adaptation) can be found in [1], [11], [34]–[37].

A. *Brief Review of SDN and NFV Adaptation Potentials*

We consider the three functional primitives supporting adaptation in turn.

1) Observation: Observation includes the following main networking tasks related to adaptation.

a) Monitoring: SDN and its OF protocol provide ideal means to collect flow data and monitor network traffic. A logically centralized network control provides the opportunity to collect entire traces and packet histories either "out-of-band" by collection through an out-of-band control channel, e.g., NetSight [38] leverages SDN to trace entire packet histories (without sampling), or "in-band", e.g., CherryPick [39] uses packets to carry information of SDN paths "in-band" (namely, a subset of links along the packet trajectory). VNF management systems can provide additional measurement points inside VNFs that can potentially enhance the level of detail and accuracy of network monitoring [40], [41]. Software-defined adversarial trajectory sampling [42] additionally provides a direct passive measurement method to infer packet routes, even in adversarial environments.

b) Event Detection: The fast detection and handling of (link or node) failures was one of the main motivations for Google's move to SDN: the logically centralized perspective overcomes distributed reconvergence, and supports centralized, and hence optimized, failover routing [43].

c) Adaptive Measurements: Routing flexibilities can be exploited to assist traffic monitoring. As traffic characteristics and measurement objectives can dynamically change over time, a fixed placement of traffic monitors can be suboptimal. While it is not always feasible to dynamically redeploy/reconfigure the measurement infrastructure to cater to evolving measurement requirements, an alternative solution may be to strategically route traffic subpopulations over *fixed* monitors: the idea of *MeasuRouting* [44]. MeasuRouting monitors transit traffic at one or more points in a network. Network operators can employ MeasuRouting for traffic accounting, debugging or troubleshooting, forensics, and traffic engineering.

2) Composition:

a) Function Chaining: Network flows may not only need to traverse one VNF, e.g., Network Address Translation (NAT), but multiple VNFs in a pre-defined order, whereby multiple functions may need to be added or removed on demand. For instance, network traffic should be inspected and blocked before being translated by a NAT. Function chains incorporate a wide range of services that are typically provided by networks, from in-network packet processing functions (e.g., virtualized middleboxes) to applications (such as data storage or application servers) [11].

b) Function Configuration: With the implementation of network functions in software, VNFs offer great potential for configuration optimization, even at runtime. Accordingly, to deliver optimal performance, the right configuration settings need to be found for varying network conditions. Taking the example of a Mobile Radio Access Network (RAN), FlexRAN [45] offers such flexible and tailored network configuration to meet prescribed performance requirements. FlexRAN pushes the mobile radio processing functions to programmable hardware of individual eNBs, e.g., to achieve ultra-low latencies, or to a central RAN cloud serving multiple eNBs, e.g., for extensive processing power. Hence, an adaptive function configuration can meet previously hardly feasible performance requirements with a static configuration.

c) Function Resource Dimensioning: The performance of VNFs relies on the available hardware resources. Thus, to meet the demands of network users, cloud resources need to be acquired according to the users' networking demands. This dimensioning involves two steps: when designing a VNF and when provisioning VNF chains at runtime [46]. Resource allocation and dimensioning of VNFs have been widely evaluated, e.g., a sharing scheme for cloud computing resources to improve radio clustering and dynamic radio cooperation in a Cloud-RAN has been proposed in [47], while compute resource sharing for RAN functions has been examined in [48].

[2]https://github.com/p4lang/p4ofagent

Generally, there is a wide range of granularity levels of networking resources that can be considered in resource dimensioning, e.g., fine-grained buffer and link resources, or more coarse-grained subnetwork resources. Moreover, advances in awareness and control of computing and networking resources, such as CPU pinning, non-uniform memory access (NUMA), and single-root input/output virtualization (SR-IOV), allow a wide range of granularity levels of resource allocations.

d) Function Placement: VNFs may serve many network services, i.e., they may process the network packets of many network flows. As those flows may originate from different locations in a network while demanding, e.g., strict Quality of Service (QoS) guarantees, network functions need to be placed optimally to meet constraints and reduce costs [49]. As a candidate example of VNFs that serve a multitude of user flows, [50] evaluates the optimal placement, i.e., the location, of mobile core network functions with the target objective of minimizing the VNF deployment cost while satisfying the latency performance requirements for all user traffic flows.

3) Control: With respect to control, SDN and NFV can improve the network adaptation in the following main categories.

a) Traffic Engineering: In order to improve traffic engineering metrics (e.g., minimizing the maximal link load), a system administrator or operator may decide to reroute (parts of) the traffic along different links. For example, Internet Service Providers may switch between multiple routing patterns during the day, depending on the expected load. These patterns may be precomputed offline, or may be computed as a reaction to an external change (e.g., due to a policy change of a Content Distribution Provider) [51].

b) Consistent Network Updates and Reconfigurations: Any dependable network needs to be flexible and support reconfigurations and updates at runtime, e.g., to improve traffic engineering, to support maintenance work, or add and remove services. For example, in order to replace a faulty router, or to upgrade an existing router, it may be necessary to temporarily reroute traffic. The ability to flexibly change routes is also useful to support new network policies, e.g., in security policies [22].

c) Saving Energy: Reacting to current traffic conditions by rerouting traffic and powering down as many unneeded links and switches as possible, can lead to significant energy savings. For example, *ElasticTree* [52] relies on prediction methods to decide which subset of links and switches to use to obtain significant energy savings.

d) Admitting and Embedding Services and Virtual Networks: When fully leveraging the available flexibilities, it is not only possible to flexibly route or deploy network functions, but also to combine the two: emerging network services such as service chains or virtual networks can come in various shapes, connecting network functionality (e.g., middleboxes) by provisioning routes with prescribed bandwidth. The resulting problems can be seen as variants of virtual network embedding problems, Rost et al. [53] provide an overview of the complexities of this problem. An overview of admission and control complexities in SDN virtual networks is provided in [54].

B. Algorithmic Challenges of SDN and NFV Adaptations

The increased flexibilities afforded by SDN and NFV not only bring many potential advantages, but also introduce new challenges. These include implementation related issues, such as virtualization and isolation, reactiveness, scalability, as well as optimization problems. The optimization problems relate, in particular, to the decision phase as described in the introduction and in Fig. 2. As an example and since they are studied intensively, we concentrate on the optimization challenges in this subsection. With every additional configuration possibility, the complexity of existing optimization algorithms may double. Thus, new algorithmic techniques may be required to fully exploit the new flexibilities as introduced by SDN and NFV.

Complexities emerge when aiming to support efficient observation (e.g., sampling strategies with an optimal tradeoff between the overhead of sampling and the benefits of coverage) and control (e.g., reconfiguring flows in a network consistently, quickly, and with minimal controller interactions). In the following, we consider the functionality of composition to illustrate the algorithmic challenges arising from SDN and NFV adaptations.

In the context of composing network functions, we currently witness increasing deployments of middleboxes, e.g., firewalls, proxies, and traffic optimizers in computer networks [55]. Moreover, the functions of middleboxes are increasingly virtualized, making the middlebox functions available as VNFs. VNFs can be quickly and flexibly deployed through a variety of computing infrastructures, e.g., multi-access edge computing while reducing costs. Importantly, over the last few years, innovative new network services have been promoted by industry and standardization organizations [24], by *composing* service chains from VNFs [56], [57].

The benefits and technological challenges of composing complex network services from individual VNFs through SDN and NFV have been intensely studied. The flexibilities of composition are typically achieved by adaptively steering traffic through service chains that are flexibly composed from VNFs. However, much less is known today about the algorithmic challenges underlying the routing through such VNFs, which are also referred to as *waypoints* in the algorithm theory context. In a nutshell, the underlying algorithmic problem is: How to route a flow (of a certain size) from a given source s to a destination t, *via a sequence of k waypoints* $(w_1, \ldots, w_k)$? The allocated flow needs to respect capacity constraints, and ideally, be as short as possible, in terms of the number of hops (to minimize the bandwidth and/or energy consumption and minimize latency).

The waypoint routing problem comes in many different flavors, e.g., depending on whether a shortest or just a feasible route needs to be computed, depending on the number k of waypoints, and depending on the type of the underlying network, e.g., directed vs. undirected, or Clos vs. arbitrary topologies. Moreover, some VNFs (waypoints), such as security and performance related waypoints, may or may not be *flow-conserving*: e.g., a tunnel entry point may *increase* the packet size by adding an encapsulation header, whereas a wide-area network optimizer may *decrease* the packet size by compressing the packet.

In contrast to classic traffic engineering and routing problems, the basic waypoint routing problem arising from deep composition comes with a fundamental twist: routes are not restricted to form simple paths, but can rather form arbitrary *walks* (i.e., routes which may include links and nodes repeatedly), as long as capacity constraints in the underlying network are respected. Indeed, often feasible routes do not exist if restricted to a simple path.

However, merely computing a shortest route through a *single* waypoint in a capacitated network turns out to be a non-trivial task. The problem is related to a classic combinatorial problem: the disjoint path problem [58], [59] and the k-cycle problem [60]. Amiri et al. [61] recently provided an overview of hardness results and algorithmic techniques for this problem. It is known that in some (undirected) networks and for a single waypoint, *shortest* routes can be computed efficiently, e.g., using Suurballe's algorithm. Waypoints which change the flow size however are more challenging, and directed links make it hard as well. For multiple waypoints, *feasible* routes through a fixed number of waypoints can be computed in polynomial time: This result follows by a reduction to a classic result by Robertson and Seymour [62]. For general k, even on undirected graphs, the decision problem (whether a feasible route *exists*) is NP-hard. For multiple waypoints, the required algorithmic techniques depend on whether waypoints need to be traversed in a specific order [63] or not [64].

Another dimension is the flexible deployment of middleboxes: Emerging flexible networks do not only allow to steer traffic more flexibly through middleboxes, but also to flexibly deploy middleboxes. However, also the problem of deploying middleboxes is challenging. Lukovszki et al. [65] recently showed that the problem is NP-hard as well. On the positive side, however, polynomial-time algorithms for an optimized *incremental* deployment of middleboxes is possible if one resorts to approximation algorithms: the problem features a submodular structure.

Besides these examples, several other optimization problems arise, e.g., in terms of efficient network updates [22]. Existing algorithms often have in common that they are oblivious to the actual workload, past performance or additional aspects which are not part of the model for which the algorithms were designed. In the next section, we will argue that data-driven algorithms may provide an interesting alternative.

IV. Exploiting Adaptation Potentials of SDN and NFV: A Data-Driven Approach

While software-based and virtualized networks introduce many flexibilities and optimization opportunities, exploiting these flexibilities is challenging. Indeed, as illustrated in Section III-B, the judicious exploitation of the flexibilities afforded by softwarized networks based on SDN and NFV poses hard optimization problems. For instance, embedding virtual networks and service function chains is a computationally hard problem [53]; hence, the corresponding objective functions are difficult to optimize, even for computers. Even seemingly simple tasks, such as computing a shortest route through a *given single* (virtualized) network function [61], *placing a single* network function [65], or deciding which requests to admit [66], are all hard.

In addition to the computational complexity, the question arises how accurate our underlying models actually are: communication networks are complex and are compromised of many different aspects. For instance, when combining SDN and network virtualization [67], requirements change continuously, e.g., in terms of demands and workloads, and objective functions are typically complex. This network complexity makes real-time management almost impossible for human operators with todays tools; moreover, the complexity makes network modeling very challenging.

Hence, new ways for improving existing network management algorithms are needed. In this section, we argue that data-driven approaches provide such an opportunity. First, data-driven approaches facilitate interesting and fast heuristics that can outperform existing rigorous optimization or approximation algorithms. Second, and more importantly, data-driven approaches adapt to and optimize for the *actual* state of the network, as characterized by the network observation data (e.g., workloads, performance measurements, interference), rather than for some hypothetical worst-case network state. Thus, existing theoretical upper bounds on the maximal theoretically possible performance may no longer apply.

This motivates us to explore the use of ML in softwarized networks in this section while particularly focusing on the decision phase. In Section IV-A, we outline how data-driven network operation may lead to "self-driving" networks. Subsequently, in Section IV-B we focus on data-driven ML strategies for improving the solution of network optimization problems.

Indeed, there is an increasing consensus that network operations should be supported by data-driven ML-based models revolving around high-level goals and a holistic view of the underlying network, see the surveys [68]–[70] and position papers [14], [71]–[76]. Interestingly, we note a particular lack (1) of investigations on the adaptation potentials of the combinations of SDN and NFV in [14], [69], [72], and (2) thorough investigations of the increasing complexity of the decision phase arising from SDN and NFV in [68], [73], [75].

A. Why is Data-based ML Useful for Softwarized Networks?

ML for networking relies on one main principle: integrating knowledge and experience from the past. Existing solutions can be seen as data-driven, albeit only to a very limited degree. For instance, existing traffic engineering approaches are based on current network conditions, such as the current link utilization. Going further, we argue for always learning from decisions that have been made in the past as well as all relevant past and current data that may be relevant for a given task, e.g., Quality of Experience (QoE) when conducting routing for QoS-based measures, such as latency. Consider again a traffic engineering example: suppose that a routing algorithm sees recurring network conditions while making routing decisions for requests that have also been observed in the past. A data-driven approach that integrates knowledge from the past may group the requests into categories based on applications or IP domains. As a result, applications from a given domain may follow similar network paths.

Jiang *et al.* [77] coined the term *Data Driven Networking (DDN)*. Their main argument for DDN is the ability of computing to process the available large amounts of data. Hence, in a DDN-based control loop, automatically tuned algorithms base their decisions on data collected in real time. While Jiang *et al.* do not directly point to SDN and NFV as enablers, we believe that SDN and NFV will help facilitate a highly flexible realization of the DDN paradigm.

One decade ago, David Clark actually proposed the *Knowledge Plane (KP)*: an entity that integrates *intelligence* into "the fabric of the Internet itself" [78]. Mestres *et al.* [72] have recently re-visited the KP concept; in particular they wanted to answer the question why the *KP* has not yet been fully realized. Their answer was quite simple: the missing pieces of more flexible network control were softwarization paradigms, such as Software-Defined Networking and Network Function Virtualization. Hence, with the advent of flexible softwarization technologies, they (re)promoted the *Knowledge Defined Networking*—a paradigm relying on software-defined networking, network analytics, and artificial intelligence (AI). Yao *et al.* [73] followed up on the KP concept by proposing *NetworkAI*: a network architecture that makes use of SDN, network monitoring technologies, and reinforcement learning technologies.

Ayoubi *et al.* [71] have further pursued this direction. They outline how ML can be used to cognitively manage networks; cognitively in the sense that architectures should support autonomic, self-managing network operation. Ayoubi *et al.* argue that with the introduction of SDN and NFV, there is high potential for new sources of valuable data, which need to be handled by ML due to their inherent complexity. The work of Ayoubi *et al.* is based on the notion of *cognitive networks* as introduced by Thomas *et al.* [79]. Cognitive networks are one example of a system description that explicitly relies on learning and data gathering to adapt the configurable parameters of a network [79]. Cognitive networks are primarily applied in the context of wireless networks and try to overcome the limitations introduced by layering, i.e., control decisions in a given layer are based not only on the information as observed by that layer but on information from all layers [79], [80]. In contrast to the preceding studies we provide an overarching conceptual framework for exploiting the different functional primitives of SDN and NFV for network adaptations.

Wu *et al.* [76] have extended the notion of knowledge-defined networking to *Knowledge Centric Networking*. Wu *et al.* considered an IoT scenario, where ML can extract useful information from sensors at the edge of the network to reduce the burden in the core of the network. They propose the usage of SDN together with ML to deploy and manage network slices.

Latah *et al.* [70] have provided an extensive overview of artificial intelligence techniques that have been used in the context of SDN, including various placement problems and security problems. *Xie et al.* [69] have provided a comprehensive survey on ML techniques applied to SDN without NFV. Moysen *et al.* [75] have provided a survey on self-organized network management, noting that for *Self-organizing Networks*, self-organization should be analyzed from an end-to-end perspective. Boutaba *et al.* [68] have comprehensively surveyed ML for networking.

All these preceding papers differ from our article in that we consider the additional degrees of freedom introduced by SDN and NFV with strong focus on the decision phase. We make concrete suggestions on how ML can help to tackle the increasing complexity of the decision phase.

Hence, the softwarization of communication networks can be seen as an *enabler* for next-generation, self-driven networks: Increased automation has the potential to not only simplify network operations, but also to enable fine-grained optimizations. The fine-grained optimizations can fully leverage the available *network observation data* rather than relying on predefined models [14]. Thus, a vision emerges of fully *Self-Driving Networks* that continuously measure, analyze, and control themselves [14], [81]. Before we elaborate on the concept of self-driving networks, we explain data-driven ML concepts that consider past knowledge.

In particular, SDN and NFV introduce new challenges with respect to network optimization. With the increasing decision possibilities due to SDN and NFV, optimization algorithms are forced to consider high-dimensional optimization problems. These highly complex optimization problems cannot be solved within reasonable runtimes with the traditional approaches that only consider the current network state and ignore the knowledge of the past.

B. ML for Improving SDN/NFV-related Network Optimization

This section reviews ML strategies to enhance the three functional primitives of softwarized networks in support of adaption.

1) Deep Observation: The large amount of information that will be available through deep observation is a promising opportunity, but also poses new challenges, which can be effectively addressed with ML strategies. The main challenges relate to the efficient acquisition of deep observation data and the data representation.

Data Acquisition: While the availability of fine-grained observation data is generally beneficial, the data collection and transmission to a central entity causes additional network load. In addition, fine-grained measurements for all flows in a system at the same time may not be feasible due to hardware limitations, e.g., the available TCAM limiting the number of flow-rules [82]. ML can mitigate the additional load by predicting future states [83], [84], and provide more fine-grained data to downtream applications through learned models than would otherwise be possible [82]. A system can now obtain fine-grained data even for large networks. Predicted future states can also be used to detect deviations from expected behaviors and thus potentially be directly used for anomaly, novelty, and fault detection [85].

Data Representation: Two main challenges in the representation of data acquired from deep observations in softwarized SDN and NFV networks are (1) reducing the dimensionality of the observed data, and (2) capturing the spatial relations of the network data. Accordingly, the following paragraphs highlight examples from ML that enable the exploitation of the data becoming available with SDN and NFV.

Concretely, an input model from one set of network optimization problems should be generalizable and applicable as input to other similar network optimization problems, i.e., solutions for small networks could potentially provide insights for large problem instances. Moreover, the representations need to be storage-efficient. Simply memorizing all data without any compression may be infeasible due to space constraints.

a) Dimensionality Reduction: Networks are large systems, as a consequence data is usually high dimensional. A data point could for example be a traffic matrix with N^2 elements, or a vector of CPU utilization of nodes. Often, a compressed representation of the data can be obtained through *representation learning* [86]. The goal of representation learning is the transformation of data into a representation that is more suitable for classifiers or other predictors. Well-known examples for such a transformation are Principal Component Analysis (PCA), auto-encoders, and Convolutional Neural Networks (CNNs) for spatial data. Especially CNNs can extract task-specific representations by providing an end-to-end differentiable learning framework. A disadvantage of such techniques is their black-box behavior. A user generally cannot directly comprehend the inner workings or interpret the outputs of such models. This is especially true for representations learned through neural networks. Interpretable models can be obtained through the use of probabilistic modeling. So-called stochastic block models (SBMs) [87] can extract compressed representations for relational data that can be interpreted by humans. The SBM assigns nodes to groups and models the connectivities among the groups. This allows the visualization of the network, the role discovery of nodes, and the identification of the organizational structure of the network. Probabilistic models also provide a principled way to learn representations through neural-networks [86].

b) Representing Relational Network Data: A main challenge in the application of ML techniques for the dimensionality reduction of data representations of SDN and NFV networks is the inherent relational nature of the networking data, which violates basic assumptions of standard ML approaches and makes them difficult to apply. ML techniques usually operate on vector valued data under the independent and identically distributed (IID) assumption. In contrast, the typical underlying data structures in networking are graphs. In order to use ML techniques, such as Decision Trees, Neural Networks, or Support Vector Machines, graphs first have to be transformed into a representation that can be used by the ML techniques. Of course, various data structures exist to represent attributes of graphs: e.g., the adjacency matrix, where the attributes of edges and nodes are represented as vectors. However, such a representation has shortcomings: the network may change over time in case of failures or when adding or removing nodes in an overlay network—accordingly the representation, i.e, the input for ML techniques would have to change. In such a case, an adjacency matrix

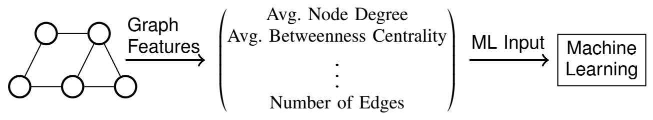


Fig. 3: From graph to feature vector: The feature vector contains topological attributes, such as average (avg.) node degree, average betweenness centrality, and number of edges.

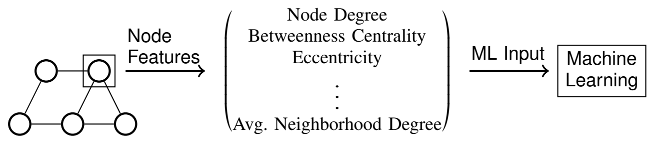


Fig. 4: Constructing node feature vector: The box indicates the node for which a node topological feature vector is constructed. Features can, for instance, be the node degree, the betweenness centrality, the eccentricity, or the average neighborhood degree.

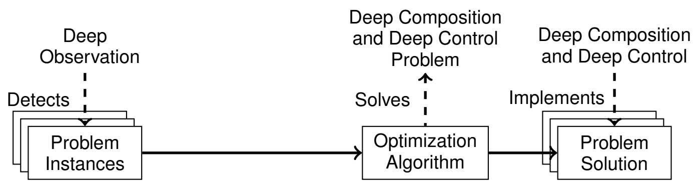
would have to change its dimension, i.e., the input of the ML algorithm would no longer be scale free. Furthermore, such representations may not allow for compression; for instance, an ML algorithm may only require knowledge of the current number of nodes and edges with enough capacity for making simple feasibility checks when deciding on the admission of service function chains. Accordingly, using an adjacency matrix would introduce an information overhead that is simply not needed.

c) Network Graph and Node Transformations: The importance of graphs not only in the field of communication networks but also in other fields, such as sociology and biology, has led to a number of techniques and approaches that are able to transform graphs and nodes in a graph. To just name a few examples, graph kernels take a graph and transform it into a (high-dimensional) Hilbert space [88]. Another concept could be to use topological feature attributes, such as degree, betweenness, or closeness centrality, and to put these attributes in a feature vector [89], [90]. Such representations not only compress the graph data, but also provide a scale-free graph representation. Fig. 3 shows an example of the procedure for constructing a graph feature vector, which can be used as input to an ML algorithm.

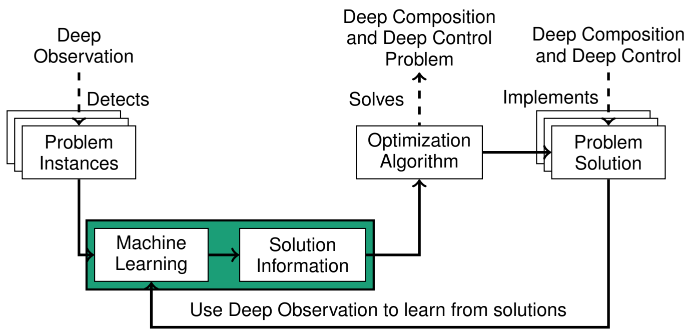
Similar to graph representations, different node representation concepts have been developed in the graph and ML research fields. Again, topological node features, such as the degree or the betweenness centrality of a node, can represent information about, e.g., the importance of a node, as shown in Fig. 4.

2) Deep Composition and Deep Control: Deep composition and deep control in softwarized networks pose problems with structures akin to conventional optimization problems. Accordingly, we describe in this subsection how ML-based strategies can exploit the data from deep observation to enhance the decision phase in Fig. 2. That is, how can ML-based strategies speed up the solution finding process or the solution quality of composition and control-related optimization problems.

Figure 5 compares the traditional way of solving optimization problems (Fig. 5(a)) with the ML-enhanced solution procedure (Fig. 5(b)). In Fig. 5(a), problems are solved by

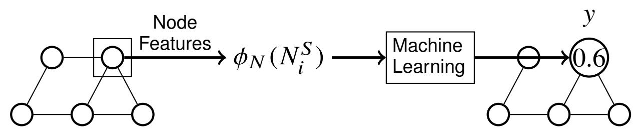


(a) Traditional problem optimization.

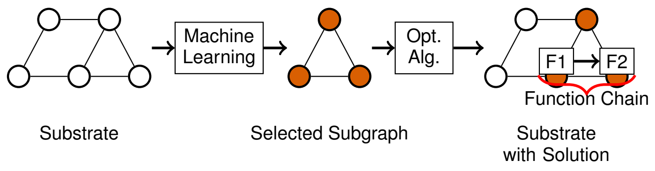


(b) ML-enhanced problem optimization.

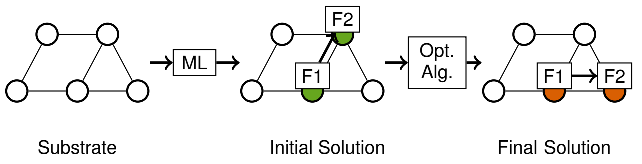
Fig. 5: Comparison between traditional and ML-enhanced solution of optimization problems. The traditional approach neglects solutions of previous problem instances. ML speeds up the decision process for composition and control-related optimization problems by taking into account data from past problem solutions with deep observation techniques. The advantage of deep observation is that it provides a means to detect or see changing environments and inputs. Furthermore, ML techniques can efficiently represent and store problem instances from the past. Efficient deep composition and control rely on the ML-advanced solving of optimization problems.



(a) Node representation: $\phi_N(N_i^S)$ maps a node into a scale-free feature representation. This representation serves as input for the ML algorithm. The ML algorithm outputs a probability value y, which, e.g., provides the probability of this node being part of a subgraph for search space reduction or being part of an initial solution.



(b) Search space reduction example: The ML algorithm determines a subset of nodes (subgraph), which contains nodes with high potential to be part of an optimization solution. For instance, the selected nodes should host a network function chain. In the depicted example, the ML algorithm selects three nodes, and the optimization algorithm then selects two nodes, e.g., for a function chain.



(c) Initial solution example: Based on previously placed function chains, the ML module provides an initial solution, e.g., for a function chain (F1 and F2 deployed on the green nodes). An optimization algorithm, such as Simulated Annealing, then starts searching in the vicinity of the given solution. Here, only one node then needs to be replaced.

Fig. 6: From graph nodes to ML input to probabilities: ML predicts for each node a probability. The probability values can be used and interpreted for different use cases: e.g., search space reduction as shown in Fig. 6(b) or to provide an initial solution as illustrated in Fig. 6(c).

an optimization algorithm. In contrast, in Fig. 5(b), data available from the previous problem solutions are fed into an ML component. The ML component generates additional information that can be used by the optimization algorithm to find better solutions faster. We can distinguish the following use cases for such a "boosting" approach.

a) Predicting Problem Feasibility: SDN/NFV-based systems receive requests, e.g., for virtual networks connecting VNFs or for flow routing, that are arriving or changing over time. An arriving or changing request may not be served or adapted at all over time—the request may be infeasible. Reasons for infeasibility are, for instance, that a system simply does not have enough remaining capacities (CPU or data rate) to serve a specific demand. While predicting feasibility may be trivial in some cases, e.g., a network does not have at least one node with enough remaining capacity to serve a request, there are also cases in which complex requests cannot simply be decided due to their combinatorial nature. For instance, for a function chain request with n links, it is possible that $n-1$ links can be embedded while the nth virtual link cannot be served. Checking such demands for feasibility requires in the worst case the checking of all possible 2^n combinations. ML could detect patterns within requests that are hard to be served [91]; hence, requests could be quickly rejected. Such quick rejections could potentially avoid time-intensive modeling and solution tasks of, e.g., optimal solver software.

b) Predicting Objective Values: Is it worthwhile to serve a specific feasible request? Predicting the cost of a networking problem may help to answer this question. Moreover, in cases where multiple infrastructure providers compete for hosting requests, quickly evaluating the revenue to cost relationships of requests can provide a competitive advantage. Aside from such visionary use cases, conventional use cases, such as routing, may also benefit form predictions. A controller that does not explicitly rely on shortest path routing could simply evaluate a predetermined set of alternative paths with predicted costs.

c) (Problem) Search Space Reduction: NFV increases the placement possibilities of network functions due to virtualization; any available data center may be a potential host for a VNF. Accordingly, not only the potential search space for locations is increasing, but also the demand to make fast decisions for such optimization problems. VNF users may expect short provisioning times. Indeed, the time to turn on VNFs is determined, e.g., by the time to turn on a virtual machine—an action that can be accomplished within seconds [92]. While many similar optimization problems exist in SDN- and NFV-based networks, many underlying problems relate to classical optimization tasks: bin packing and set covering. As these problems are generally hard to solve, ML potentially provides ways to reveal dependencies between, e.g., current network states and the requested VNFs. Thus, locations could be pre-excluded from an algorithm's search space [93]. Based on solutions to past problems, ML could learn models that guess the probabilities of nodes or links to be used for a solution. Beside using data and learning models, it has been demonstrated that exploiting the power of neural computations yields similar results, i.e., can efficiently shrink the search space [94]. Fig. 6 illustrates two examples. In Fig. 6(a), a node is characterized by the likeliness that it is selected, e.g., for function placement. A subgraph preselection containing likely selected nodes is shown in Fig. 6(b).

d) Provide Initial Solutions: Many heuristic and optimal algorithms rely on initial starting solutions. Hence, if repeating problems appear, such as routing or VNF chains in SDN and NFV, ML may provide a performance boost. Based on observations, there may be potential to construct valuable starting solutions: initial controller or virtual functions placements, or initial routes, or rule settings in SDN networks. Such information could be extracted from already served demands. Different angles of learning from problem solutions have been demonstrated for SDN/NFV-related composition and control problems [29], [95]. For instance, a neural network predicted initial solutions for a simulated annealing approach solving the dynamic controller placement problem [95]. Fig. 6(c) sketches the procedure for function chaining.

Not directly targeting SDN/NFV-related use cases, many concepts have emerged in the ML area that promise to improve general and network-specific optimization problems. For completeness, we point the reader to the literature on utilizing deep reinforcement learning and on recent advances in the area of deep-reinforcement learning [96]–[104].

3) Summary: To summarize, SDN and NFV provide new means of gathering data that is potentially useful for the decision phase of network adaption. However, the inherent problems of networks, such as transmitting data over low-bandwidth or high-delay links in order to process the state at a centralized location, still exist when applying ML solutions in the decision phase. Moreover, all challenges existing in traditional fields of ML applications also arise when applying ML to networking problems. For instance, rare events, e.g., flash crowds, can complicate ML-based decision making. As such events occur very rarely, the overall observation data may be biased. Hence, ML algorithms tend to incorrectly classify those rare events for the sake of simplicity and high accuracy. Furthermore, data representations need to scale freely in case of dynamically changing networks. ML techniques may require special adaptations for their use in networking: e.g., similar to how convolutional neural networks advanced image classification, specialized neural network structures may help to solve networking problems.

V. Towards Autonomous Networks

All examples of deep composition and control have in common that they are target-driven: operators define specific goals such as low latency for their network operations; ML can speed up algorithms solving such problems. In this section, we outline a concept called *empowerment*, which we envision as a promising basis for making networks truly self-driving. Networks should be equipped with mechanisms that allow them to discover concepts that prepare them for (= let them adapt to) novel situations, e.g., for events or changing objects that were not a priori hard-wired into their operation by a human (such as setting a specific objective).

Already today, communication networks are notoriously difficult to manage. The capabilities emerging with deep observation, composition, and control will make the network management even harder. The novel concept of self-driving networks [14] offers an opportunity to escape this dilemma by utilizing the capabilities of deep observation, composition, and control to their full effect. In particular, self-driving networks tightly incorporate measurement and control, i.e., self-driving networks could use the large amount of information provided by deep observation to manage the network and to react to external events through the capabilities provided by deep composition and control, which are enabled by the SDN and NFV functional primitives.

While it is presently unclear how exactly self-driving networks will be designed, it is certain that they will have to cope with uncertainty, e.g., uncertainty in the arrival of new flows, uncertainty about the traffic volume of individual flows, and uncertainty regarding the hardware availability. Certainly, a desirable property of a self-driving network is being *prepared* towards requirements that arise in the near future. Being prepared can refer to the network being robust to environmental changes and open for new requests (e.g., new flows). The network performance should not degrade significantly due to small environmental changes, such as slight variations in a routing matrix, and networks should be able to accommodate a wide range of new requests at any time. This requires for instance active resource management to avoid resource fragmentation. With active resource management, networks can accept additional requests, or reduce the amount of reconfigurations necessary to cope with new requests. The rationale is that the network should bring itself into a state from which it can react faster to new situations. Hence, self-driving softwarized networks can be regarded as the ultimate adaptable networks.

A semblance to preparedness is currently achieved through *robust optimization* or *stochastic optimization* [105], [106]. Robust optimization is concerned with providing worst-case guarantees for uncertainty in the parameters, represented as sets. Stochastic optimization can be used in a similar fashion in cases where the objective function contains noise. Both approaches prepare the network against environmental uncertainty, e.g., changes in a routing matrix, but do not include active resource management as illustrated in [107]. In addition, both require optimization paradigms usually a rigorous mathematical model of the system that should be optimized, where an objective function serves usually a specific purpose. In contrast, empowerment may facilitate a way of “improving” network operations further by not relying on a specific objective, which is actually the case for unpredictable future use cases.

A. Empowerment

Empowerment is an information-theoretic measure motivated by the observation that living organisms strive for states that give them maximum control or influence over their

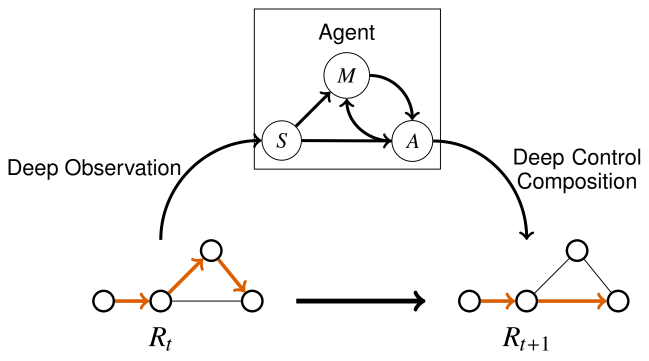


Fig. 7: Illustration of self-driving network operating with empowerment: An agent consists of a sensors S, actuators A, and memory M. The actuators use deep control and composition to influence the environment which is in state R_t. The environment transitions into a new state R_{t+1}, which is observed by the agent through its sensors using capabilities from deep observation.

$$\begin{tikzcd}
{} \arrow[rr, dashed] & & M_t \arrow[rrr] \arrow[drr] & & & M_{t+1} \arrow[rrr] \arrow[drr] & & & M_{t+2} \arrow[rr, dashed] \arrow[dr, dashed] & & {} \\
{} \arrow[r, dashed] \arrow[urr, dashed] & A_{t-1} \arrow[ur] \arrow[dr] & & S_t \arrow[r] \arrow[urr] & A_t \arrow[ur] \arrow[dr] & & S_{t+1} \arrow[r] \arrow[urr] & A_{t+1} \arrow[ur] \arrow[dr] & & S_{t+2} \arrow[r, dashed] \arrow[ur, dashed] & {} \\
{} \arrow[rr, dashed] & & R_t \arrow[rrr] \arrow[ur] & & & R_{t+1} \arrow[rrr] \arrow[ur] & & & R_{t+2} \arrow[rr, dashed] \arrow[ur] & & {}
\end{tikzcd}$$

Fig. 8: Network showing the dependencies between variables in the time unrolled perception action loop of an agent with memory. The variables R_t correspond to environment (network) states, the variables S_t to sensor readings, the variables A_t to actions, and the variables M_t correspond to the agent's memory. The figure shows that an action depends not only on the current sensor readings and thus environment state, but also on the memory the agent acquired over its lifetime.

environment: Everything else being equal, states are preferable which (1) keep as many options open as possible, or (2) in which actions have the largest influence on the direct environment [108], [109]. The concept of empowerment is an attempt to formalize and quantify the influence an agent (the entity interacting with its environment) has on its environment. This contrasts with current approaches in communication systems, such as robust/stochastic optimization and general optimization (which does not include parameter uncertainty).

Empowerment is tightly coupled to the concepts of *embodiment* and *situatedness* of an agent in an environment. Situatedness means that the agent has to deal with sensory and motor contingencies of the environment. Embodiment means that the agent experiences the environment directly through its sensors and influences it through its actuators. Empowerment is then measured as the mutual information between actions and sensory information obtained at a later time step. In particular, SDN with its centralized view provides great opportunities for sensing (monitoring through deep observation) the network in a fashion that has not been available with legacy technologies.

The interplay of agent and environment can be represented as a perception-action loop, where the agent influences the environment through its actuators, and receives a perception of the resulting environmental state through its sensors. In addition, the agent can be equipped with a memory, storing past sensor-motoric data and use it to make decisions [110]. In networking, actuators could, e.g., change the topology or adapt routing weights (through deep composition and control). Sensors could measure the QoS, e.g., the perceived packet latency, or could—in an abstract sense—correspond to statistics, e.g., port- or flow level statistics (obtained through deep observation), as shown in Fig. 7. Fig. 8 illustrates the perception-action loop as a time unrolled Causal Bayesian Network (CBN). The CBN contains the following random variables:

- the sensor S taking values $s \in \mathcal{S}$,
- the actuator A taking values $a \in \mathcal{A}$,
- the memory M taking values $m \in \mathcal{M}$,
- the environment R taking values $r \in \mathcal{R}$,

Fig. 8 shows that sensors and actuators are connected through the environment. An agent selects an action for the next time step based on the sensory information of the current time step, and possibly past experience that the agent has stored in its memory. The action influences the state of the environment, which in turn affects the sensor input.

The perception-action loop can be understood as a probabilistic channel, and empowerment $\mathfrak{E}$ is defined as the channel capacity between the agent's actuator A_t and sensor S_{t+1}:

$$\mathfrak{E} := C(A_t \to S_{t+1} \mid M_t) = \max_{p(a_t|m_t)} I(S_{t+1}, A_t \mid M_t), \qquad (1)$$

where $I(S_{t+1}, A_t \mid M_t)$ is the mutual information between the sensor reading in the next time step S_{t+1} and the current action A_t conditioned on the agent's memory M_t [108], [110]. Mutual information measures the average information one can gain about S_{t+1} by observing A_t [108]. Thus, empowerment can be understood as a measure of how much *influence* an agent has on its environment given its actuator, sensor, and, possibly, memory. If all actions result in the same states, then the agent has no empowerment, i.e., $\mathfrak{E} = 0$. The same holds if actuators have no perceivable influence on the environment, e.g., when the environment changes randomly. To obtain $\mathfrak{E} > 0$, the effect of the actuators must be perceivable by the sensors.

Fig. 8 also illustrates the need for a causal model of how the action A_t in the current state of the environment R_t influences the sensor reading S_{t+1} in order to calculate empowerment. This model tells the agent how its actions influence future readings for the actual states of the environment. Depending on the concrete use-case, existing system dynamics obtained through a control-theoretic framework can be leveraged. For instance, control theory has been used to describe the system dynamics of congestion control algorithms [111] and to directly control flow rates [112]. In settings with highly non-linear or even unknown system dynamics, a causal model can be learned directly from observations by the agent itself [113].

It is important to note that empowerment represents the *potential* information flow. The agent calculates how it *could* affect the environment, and does not necessarily materialize its potential [114]. A way to act upon empowerment could be to greedily select an action resulting in the state with the highest empowerment.

Empowerment has a number of desirable properties [110]:

- Empowerment is **agent-centric**: Only information accessible to the agent is used, i.e., samples from the perception action loop (sensi-motoric data).
- Empowerment features **locality**: No global knowledge of the environment is necessary.
- Empowerment is **well-defined and computable**: Due to the channel formulation, standard information-theoretic quantities and established methods can be used for its calculation.
- Empowerment is **semantically unbiased**: No external reward system is introduced.

Especially the last point sets empowerment apart from traditional Reinforcement Learning (RL) approaches [103] and mathematical optimization. In RL and optimization, the designer has to define a specific reward signal or objective (e.g., related to QoS or QoE). Defining a specific objective function is often non-trivial. The objective may be multi-dimensional and potentially depend on many aspects (e.g., on routing latency and resilience). In contrast, empowerment depends only on the agent's embodiment and the environment. This makes empowerment readily applicable to self-driving networks since no extensive environment (network) model is required, and empowerment can be used to maximize future options. Specific measures, such as latency or throughput, may be implicitly optimized as well [109]. In fact, empowerment was motivated in [109] in part by the difficulty of crafting problem-specific utility functions for evolutionary algorithms. Empowerment was proposed as a universal utility function that implicitly results in desirable behaviors of artificial agents.

B. Empowerment and Deep Observation

The empowerment framework relies strongly on the capabilities provided by deep observation in order to utilize deep composition and control. Only if the changes introduced into the environment are reflected in the sensory information, then the agent can utilize the empowerment framework [108]. SDN and NFV open new interfaces for deep observation of sensing data that can be used for improving network operations. However, gathering this information comes at a cost. Therefore, the question arises of how much and what information is necessary to utilize deep composition and control?

The empowerment framework can help in this regard through sensor evolution [115]. An agent evolves its own sensors such that they are suitable for the agent's environment (and the agent's actuators). For example, evolving the sensors of an agent constructing communication networks led to structures that reduced the path length of routed requests, and increased the number of served requests in the resulting network [107].

C. Empowerment and Deep Composition and Control

Returning to deep composition and control, as enabled by SDN and NFV, how can empowerment help to leverage the additional degrees of freedom introduced by those concepts? Deep composition and control can execute the actions of an agent residing in a network environment. The agent thus could execute deep composition actions, e.g., place network functions with NFV and reconfigure the network topology, e.g., through optical devices [116], [117], as well as deep control actions, e.g., route fine-granular flows with SDN.

1) Empowerment Based Development: The control of a network often requires the execution of a sequence of configurations to obtain a specific goal. For example, re-routing a flow requires a careful scheduling of updates at forwarding devices to avoid mis-configurations [118]. Deep control and composition enable fine-grained configurations. How can these capabilities be effectively used by a self-driving network?

Here empowerment can help by providing a mechanism to *learn* sequences of actions, i.e., configurations, that lead to different outcomes. In this way, a self-driving network can *develop* a set of diverse skills that aid network operation. Since empowerment is not task specific, the learned skills could then be useful for a range of tasks, thus allowing the network to adapt more easily to changing conditions [113], [119].

2) Empowerment Guided Decision Making: Due to the versatility of deep observation and control, multiple ways may exist to realize a specific task. For instance, multiple paths could be chosen for a specific flow. Here empowerment can serve as a secondary objective to select from among the possible choices the one choice, that *keeps as many options open as possible.* Empowerment is generally not suited to optimize towards a specific goal. Using it as a secondary objective however, could result in decision making that keeps options open and thus improves the overall performance.

3) Empowerment Based Preparation: Empowerment maximizing agents can effectively restructure their environment to increase the amount of options they have in the future [108]. In the context of self-driving networks, the agent could utilize deep composition and control to reconfigure the network towards a more desirable state. For example, the self-driving network could re-route specific flows to increase the number of flows that can additionally be routed. A state with high-empowerment can then be a good starting point for future requests [107].

Also, empowerment has the built-in tendency to avoid "dangerous" situations [114]. In networking, dangerous situations could correspond to network configurations that result in serious performance degradations, e.g., a configuration leading to significant congestion. Intuitively, such a configuration would reduce empowerment; thus, an empowerment maximizing agent would reconfigure the network to avoid such situations.

D. Remarks

We believe that empowerment is an important building block towards autonomous and self-driving networks built with SDN and NFV. In turn, empowerment may actually be required to fully exploit the opportunities provided by modern and future technologies, such as SDN and NFV. While empowerment itself is generally not suited to optimize towards specific objectives, we envision empowerment as a basic drive of the network that motivates continuous improvements, which could avoid potentially catastrophic failures.

VI. Conclusion and Research Challenges

We have argued that softwarized networks in general, and SDN and NFV in particular, provide the *adaptability* that is required for future dependable and highly flexible communication networks serving our society. We proposed to structure adaptability around the three functional primitives of SDN/NFV networks, namely observation, composition, and control. We reviewed the algorithmic optimization problems arising from the flexibilities offered by SDN and NFV. Motivated by the computational hardness as well as the complex models that make it challenging to fully exploit such flexible technologies for adaptation, we argued for the benefits of data-driven decision making, e.g., decision making based on machine learning (ML). We proposed a conceptual adaptation framework that enhances the functional primitives of SDN and NFV with data-driven decision making, so as to result in deep observation, deep composition, and deep control. Finally, we argued that fully "self-driving" networks may not only be driven by classic objective functions, but also by the amount of flexibility and "preparedness" they preserve. We posited that empowerment is highly promising and requires urgent future research to fulfill the vision of self-driving networks towards ultimate adaptable and data-driven softwarized networks.

Future networks are expected to provide even more adaptation opportunities than the current versions of SDN and NFV. This increased adaptation flexibility will further increase the decision making complexity in deep observation, deep composition and deep control. Thus, the feasibility of decisions/optimization, scalability and reactiveness in real-time will continue to pose challenging research questions [19], even with the emergence of data-driven adaptation and self-driving networks up to fully empowered networks. In addition to those general challenges, we identify the following exemplary research challenges:

- Adaptation is a process over time. Hence, time-related questions, such as how fast can softwarized networks adapt or what are the time constraints for algorithm executions, need to be clarified as a basic foundation for research on adaptation in softwarized networks.
- At which scale of network sizes do the new concepts provide meaningful improvements for adaptation?
- To what extent are adaptation functionalities provided by the control plane or by the data plane?
- To what extent can adaptation be supported by new hardware concepts? Or is network adaptation a pure software concept?
- How can system performance be guaranteed based on adaptation? Can the analysis of adaptable softwarized networks be combined with formal methods? How can deterministic guarantees be provided?
- How to measure the adaptation potential in relation to the provided flexibility [6]?
- How can ML be supported by hardware? ML is great at data abstraction, however, we also need hardware that puts efficient implementations into effect.
- How far can we go with empowerment as a concept for ultimate self-driving adaptable softwarized networks? How do deployments perform in reality?

Overall we can observe that the study of highly adaptable softwarized communication network designs has just started and is likely to remain a key research problem for many years to come.

Acknowledgment

This work has been funded in part by the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation program (grant agreement No. 647158 - FlexNets "Quantifying Flexibility for Communication Networks").

References

[1] D. Kreutz, F. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[2] F. A. Lopes, M. Santos, R. Fidalgo, and S. Fernandes, "A software engineering perspective on SDN programmability," *IEEE Commun. Surv. & Tut.*, vol. 18, no. 2, pp. 1255–1272, Second Qu. 2016.

[3] N. Zilberman, P. M. Watts, C. Rotsos, and A. W. Moore, "Reconfigurable network systems and Software-Defined Networking," *Proceedings of the IEEE*, vol. 103, no. 7, pp. 1102–1124, Jul. 2015.

[4] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Communications Surveys & Tutorials, in print*, pp. 1–1, 2019.

[5] H. Hantouti, N. Benamar, T. Taleb, and A. Laghrissi, "Traffic steering for service function chaining," *IEEE Communications Surveys & Tutorials, in print*, pp. 1–1, 2019.

[6] W. Kellerer, A. Basta, P. Babarczi, A. Blenk, M. He, M. Klugel, and A. M. Alba, "How to measure network flexibility? A proposal for evaluating softwarized networks," *IEEE Commun. Mag.*, vol. 56, no. 10, pp. 186–192, Oct. 2018.

[7] M. He, A. M. Alba, A. Basta, A. Blenk, and W. Kellerer, "Flexibility in softwarized networks: Classifications and research challenges," *IEEE Commun. Surv. & Tut.*, pp. 1–1, 2019.

[8] A. Leinwand and K. F. Conroy, *Network Management: A Practical Perspective*. Addison-Wesley, Boston, MA, 1993.

[9] W. Stallings, *SNMP, SNMPv2, and RMON: Practical Network Management*. Addison-Wesley, Boston, MA, 1996.

[10] M. Subramanian, *Network Management: Principles and Practice*. Pearson Education India, Dehli, India, 2010.

[11] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, First Qu. 2016.

[12] N. McKeown *et al.*, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[13] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using OpenFlow: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 493–512, First Qu. 2014.

[14] N. Feamster and J. Rexford, "Why (and how) networks should run themselves," in *arXiv Technical Report*, 2017.

[15] Open Networking Foundation, "SDN Architecture, onf tr-502," Jun. 2014, https://www.opennetworking.org/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf, Last accessed Dec. 17, 2018.

[16] A. S. Thyagaturu, A. Mercian, M. P. McGarry, M. Reisslein, and W. Kellerer, "Software defined optical networks (SDONs): A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2738–2786, Fourth Qu. 2016.

[17] M. Jarschel, T. Zinner, T. Hoßfeld, P. Tran-Gia, and W. Kellerer, "Interfaces, attributes, and use cases: A compass for SDN," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 210–217, 2014.

[18] A. Vahdat, D. Clark, and J. Rexford, "A purpose-built global network: Google's move to SDN," *ACM Queue*, vol. 13, no. 8, pp. 1–26, 2015.

[19] O. Hohlfeld, J. Kempf, M. Reisslein, S. Schmid, and N. Shah, "Guest editorial scalability issues and solutions for Software Defined Networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 12, pp. 2595–2602, Dec. 2018.

[20] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. R. Kompella, "ElastiCon; an elastic distributed SDN controller," in *Proc. ACM/IEEE Symp. on Arch. for Netw. and Commun. Sys. (ANCS)*, 2014, pp. 17–27.

[21] J. W. Guck, A. Van Bemten, M. Reisslein, and W. Kellerer, "Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation," *IEEE Commun. Surv. & Tut.*, vol. 20, no. 1, pp. 388–415, First Qu. 2018.

[22] K.-T. Foerster, S. Schmid, and S. Vissicchio, "Survey of consistent software-defined network updates," *IEEE Commun. Surv. & Tut., in print*, 2019.

[23] ETSI. (2014, Dec.) Network functions virtualisation (nfv); management and orchestration, etsi gs nfv-man 001. Last accessed Dec. 17, 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[24] ——, "Network Functions Virtualisation – Introductory White Paper," *White Paper*, Oct. 2013.

[25] ETSI, "Network Function Virtualization (NFV); Use Cases, ETSI GS NFV 001, V1.1.1," Oct. 2013, last accessed Dec. 17, 2018. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf

[26] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "OpenAirInterface: A flexible platform for 5G research," *ACM SIGCOMM Computer Commun. Rev.*, vol. 44, no. 5, pp. 33–38, Oct. 2014.

[27] A. Basta, A. Blenk, M. Hoffmann, H. J. Morper, K. Hoffmann, and W. Kellerer, "SDN and NFV dynamic operation of LTE EPC gateways for time-varying traffic patterns," in *Proc. Int. Conf. on Mobile Networks and Management*. Springer, 2014, pp. 63–76.

[28] A. S. Thyagaturu, Y. Dashti, and M. Reisslein, "SDN-based smart gateways (Sm-GWs) for multi-operator small cell network management," *IEEE Trans. Netw. and Svc. Manag.*, vol. 13, no. 4, pp. 740–753, 2016.

[29] A. Basta, A. Blenk, K. Hoffmann, H. J. Morper, M. Hoffmann, and W. Kellerer, "Towards a cost optimal design for a 5G mobile core network based on SDN and NFV," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1061–1075, 2017.

[30] "Intel Data Plane Development Kit," http://dpdk.org/.

[31] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.

[32] "Stratum: Developing an open source reference implementation for white box switches supporting next-generation sdn interfaces," 2018, https://stratumproject.org, Last accessed Dec. 10, 2018.

[33] C. Lorenz, D. Hock, J. Scherer, R. Durner, W. Kellerer, S. Gebert, N. Gray, T. Zinner, and P. Tran-Gia, "An SDN/NFV-enabled enterprise network architecture offering fine-grained security policy enforcement." *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 217–223, Mar. 2017.

[34] R. Amin, M. Reisslein, and N. Shah, "Hybrid SDN networks: A survey of existing approaches," *IEEE Commun. Surveys & Tut.*, vol. 20, no. 4, pp. 3259–3306, Fourth Qu. 2018.

[35] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[36] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[37] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Commun. Surveys & Tutorials*, vol. 17, no. 1, pp. 27–51, First Qu. 2015.

[38] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown, "I know what your packet did last hop: Using packet histories to troubleshoot networks." in *Proc. NSDI*, vol. 14, 2014, pp. 71–85.

[39] P. Tammana, R. Agarwal, and M. Lee, "Cherrypick: Tracing packet trajectory in software-defined datacenter networks," in *Proc. ACM SIGCOMM Symp. on Softw. Def. Netw. Res.*, 2015, pp. 23–1–23–7.

[40] J. Nam, J. Seo, and S. Shin, "Probius: Automated approach for VNF and service chain analysis in software-defined NFV," in *Proc. ACM Symp. on SDN Research*, 2018, pp. 1–13.

[41] R. J. Pfitscher *et al.*, "A model for quantifying performance degradation in virtual network function service chains," in *Proc. IEEE/IFIP Netw. Operat. and Managm. Symp. (NOMS)*, 2018, pp. 1–9.

[42] K. Thimmaraju, L. Schiff, and S. Schmid, "Software-defined adversarial trajectory sampling," *arXiv preprint arXiv:1705.00370*, 2017.

[43] A. Vahdat, D. Clark, and J. Rexford, "A purpose-built global network: Google's move to SDN," *ACM Queue*, vol. 13, no. 8, p. 100, 2015.

[44] S. Raza, G. Huang, C.-N. Chuah, S. Seetharaman, and J. P. Singh, "Measurouting: A framework for routing assisted traffic monitoring," *IEEE/ACM Trans. on Netw.*, vol. 20, no. 1, pp. 45–56, 2012.

[45] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, "FlexRAN: A flexible and programmable platform for software-defined radio access networks," in *Proc. ACM Int. on Conf. on Emerging Netwo. Exp. and Techn.*, 2016, pp. 427–441.

[46] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[47] T. X. Tran and D. Pompili, "Dynamic radio cooperation for user-centric cloud-RAN with computing resource sharing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 4, pp. 2379–2393, 2017.

[48] P. Shantharama, A. Thyagaturu, N. Karakoc, L. Ferrari, M. Reisslein, and A. Scaglione, "LayBack: SDN management of multi-access edge computing (MEC) for network access services and radio resource sharing," *IEEE Access*, vol. 6, pp. 57 545–57 561, 2018.

[49] X. Li and C. Qian, "A survey of network function placement," in *Proc. IEEE Consumer Commun. & Netw. Conf. (CCNC)*, 2016, pp. 948–953.

[50] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann, "Applying NFV and SDN to LTE mobile core gateways, the functions placement problem," in *Proc. ACM Workshop on All Things Cellular: Operations, Applications, & Challenges*, 2014, pp. 33–38.

[51] V. Kotronis, R. Klöti, M. Rost, P. Georgopoulos, B. Ager, S. Schmid, and X. Dimitropoulos, "Stitching inter-domain paths over IXPs," in *Proc. ACM Symp. on SDN Res. (SOSR)*, 2016, pp. 1–12.

[52] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastictree: Saving energy in data center networks." in *Proc. USENIX NSDI*, vol. 10, 2010, pp. 249–264.

[53] M. Rost and S. Schmid, "Charting the complexity landscape of virtual network embeddings," in *Proc. IFIP Networking*, 2018, pp. 1–9.

[54] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Commun. Surv. & Tut.*, vol. 18, no. 1, pp. 655–685, First Qu. 2016.

[55] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," *ACM SIGCOMM Computer Commun. Rev.*, vol. 42, no. 4, pp. 13–24, Oct. 2012.

[56] R. Hartert *et al.*, "A declarative and expressive approach to control forwarding paths in carrier-grade networks," in *ACM SIGCOMM Computer Commun. Rev.*, vol. 45, no. 4, Oct. 2015, pp. 15–28.

[57] R. Soulé, S. Basu, P. J. Marandi, F. Pedone, R. Kleinberg, E. G. Sirer, and N. Foster, "Merlin: A language for provisioning network resources," in *Proc. ACM Int. Conf. on Emerg. Netw. Exp. and Techn. (CoNEXT)*, 2014, pp. 213–226.

[58] A. Björklund and T. Husfeldt, "Shortest two disjoint paths in polynomial time," in *Proc. Int. Col. on Automata, Lang., and Progr. (ICALP)*, 2014, pp. 211–222.

[59] P. D. Seymour, "Disjoint paths in graphs," *Discrete Mathematics*, vol. 29, no. 3, pp. 293–309, 1980.

[60] A. Björklund, T. Husfeld, and N. Taslaman, "Shortest cycle through specified elements," in *Proc. ACM-SIAM SODA*, 2012, pp. 1747–1753.

[61] S. A. Amiri, K.-T. Foerster, R. Jacob, and S. Schmid, "Charting the algorithmic complexity of waypoint routing," *ACM SIGCOMM Computer Commun. Rev.*, vol. 48, no. 1, pp. 42–48, Jan. 2018.

[62] N. Robertson and P. D. Seymour, "Graph Minors .XIII. The Disjoint Paths Problem," *J. Comb. Theory, Ser. B*, vol. 63, no. 1, pp. 65–110, 1995.

[63] S. A. Amiri, K.-T. Foerster, R. Jacob, M. Parham, and S. Schmid, "Waypoint routing in special networks," in *Proc. IFIP Netw.*, 2018, pp. 1–9.

[64] S. A. Amiri, K.-T. Foerster, and S. Schmid, "Walking through waypoints," in *Proc. Latin Am. Theor. Inform. Symp.*, 2018, pp. 37–51.

[65] T. Lukovszki, M. Rost, and S. Schmid, "It's a match! near-optimal and incremental middlebox deployment," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 46, no. 1, pp. 30–36, 2016.

[66] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," in *Proc. Int. Colloquium on Structural Inform. and Commun. Compl. (SIROCCO)*, 2015, pp. 104–118.

[67] A. Blenk, A. Basta, L. Henkel, J. Zerwas, W. Kellerer, and S. Schmid, "perfbench: A tool for predictability analysis in multi-tenant software-defined networks," in *Proc. ACM SIGCOMM Posters and Demos*, Aug. 2018, pp. 1–3.

[68] R. Boutaba *et al.*, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *J. Internet Serv. and Appl.*, vol. 9, no. 16, pp. 1–99, Jun. 2018.

[69] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Commun. Surv. & Tut., in print*, 2019.

[70] M. Latah and L. Toker, "Artificial intelligence enabled software defined networking: A comprehensive overview," *CoRR*, vol. abs/1803.06818, 2018.

[71] S. Ayoubi, N. Limam, M. A. Salahuddin, N. Shahriar, R. Boutaba, F. Estrada-Solano, and O. M. Caicedo, "Machine learning for cognitive network management," *IEEE Commun. Mag.*, vol. 56, no. 1, pp. 158–165, Jan. 2018.

[72] A. Mestres *et al.*, "Knowledge-Defined Networking," *ACM SIGCOMM Computer Commun. Rev.*, vol. 47, no. 3, pp. 2–10, Jul. 2017.

[73] H. Yao, T. Mai, X. Xu, P. Zhang, M. Li, and Y. Liu, "NetworkAI: An intelligent network architecture for self-learning control strategies in software defined networks," *IEEE Internet of Things J.*, vol. 5, no. 6, pp. 4319–4327, Dec. 2018.

[74] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine learning for networking: Workflow, advances and opportunities," *IEEE Network*, vol. 32, no. 2, pp. 92–99, Mar.-Apr. 2018.

[75] J. Moysen and L. Giupponi, "From 4G to 5G: Self-organized network management meets machine learning," *arXiv:1707.09300 [cs]*, 2017.

[76] D. Wu, Z. Li, J. Wang, Y. Zheng, M. Li, and Q. Huang, "Vision and challenges for knowledge centric networking (KCN)," *arXiv:1707.00805*, Jul. 2017.

[77] J. Jiang, V. Sekar, I. Stoica, and H. Zhang, "Unleashing the Potential of Data-Driven Networking," in *Proc. Int. Conf. Commun. Systems and Networks (COMSNETS)*, 2017, pp. 110–126.

[78] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the internet," in *Proc. ACM SIGCOMM*, 2003, pp. 3–10.

[79] R. W. Thomas, D. H. Friend, L. A. DaSilva, and A. B. MacKenzie, "Cognitive Networks," in *Cognitive Radio, Software Defined Radio, and Adaptive Wireless Systems*, H. Arslan, Ed. Dordrecht: Springer Netherlands, 2007, pp. 17–41.

[80] M. Amjad, F. Akhtar, M. H. Rehmani, M. Reisslein, and T. Umer, "Full-duplex communication in cognitive radio networks: A survey," *IEEE Commun. Surv. & Tut.*, vol. 19, no. 4, pp. 2158–2191, Fourth Qu. 2017.

[81] P. Kalmbach, J. Zerwas, P. Babarczi, A. Blenk, W. Kellerer, and S. Schmid, "Empowering self-driving networks," in *Proc. ACM SIGCOMM 2018 Workshop on Self-Driving Networks (SDN)*, 2018.

[82] A. Lazaris and V. K. Prasanna, "DeepFlow: A deep learning framework for software-defined measurement," in *Proc. ACM Workshop on Cloud-Assisted Netw.*, 2017, pp. 43–48.

[83] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt, "Deep variational Bayes filters: Unsupervised learning of state space models from raw data," *arXiv preprint 1605.06432*, 2016.

[84] K. S. Xu and A. O. I. Hero, "Dynamic stochastic blockmodels: Statistical models for time-evolving networks," *CoRR*, vol. abs/1304.5974, 2013.

[85] P. Kalmbach, A. Blenk, W. Kellerer, and S. Schmid, "Themis: Data driven approach to botnet detection," in *Proc. IEEE Infocom Posters*, Apr. 2018, pp. 1–2.

[86] Y. Bengio, A. C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *CoRR*, vol. abs/1206.5538, 2012.

[87] B. Karrer and M. E. Newman, "Stochastic blockmodels and community structure in networks," *Physical Review E*, vol. 83, no. 1, pp. 016 107–1–016 107–10, 2011.

[88] T. Gärtner, P. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," in *Learning Theory and Kernel Machines*, B. Schölkopf and M. K. Warmuth, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 129–143.

[89] G. Li, M. Semerci, B. Yener, and M. J. Zaki, "Graph classification via topological and label attributes," in *Proc. Int. Workshop on Mining and Learning with Graphs (MLG)*, vol. 2, 2011, pp. 1–9.

[90] M. Berlingerio, D. Koutra, T. Eliassi-Rad, and C. Faloutsos, "NetSimile: a scalable approach to size-independent network similarity," *arXiv preprint arXiv:1209.2684*, 2012.

[91] A. Blenk, P. Kalmbach, P. V. D. Smagt, and W. Kellerer, "Boost online virtual network embedding: Using neural networks for admission control," in *Proc. IFIP/IEEE CNSM*, Oct. 2016, pp. 10–18.

[92] C. Clark *et al.*, "Live migration of virtual machines," in *Proc. USENIX Symp. on Netw. Sys. Des. & Impl.*, vol. 2, 2005, pp. 273–286.

[93] A. Blenk, P. Kalmbach, S. Schmid, and W. Kellerer, "*o'zapft is:* tap your network algorithm's big data!" in *Proc. ACM SIGCOMM Workshop Big-DAMA*, Aug. 2017, pp. 19–24.

[94] A. Blenk, P. Kalmbach, J. Zerwas, M. Jarschel, S. Schmid, and W. Kellerer, "NeuroViNE: A neural preprocessor for your virtual network embedding algorithm," in *Proc. IEEE Infocom*, Apr. 2018, pp. 1–9.

[95] M. He, P. Kalmbach, A. Blenk, S. Schmid, and W. Kellerer, "Algorithm-data driven optimization of adaptive communication networks," in *Proc. IEEE ICNP Workshop on Machine Learning and Artificial Intelligence in Computer Netw.*, Oct. 2017, pp. 1–6.

[96] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[97] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2015.

[98] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv:1611.09940*, Nov. 2016.

[99] M. Nazari, A. Oroojlooy, L. V. Snyder, and M. Takc, "Deep reinforcement learning for solving the vehicle routing problem," *CoRR*, vol. abs/1802.04240, 2018.

[100] A. Mirhoseini, A. Goldie, H. Pham, B. Steiner, Q. V. Le, and J. Dean, "A hierarchical model for device placement," in *Proc. Int. Conference on Learning Representations*, 2018, pp. 1–11.

[101] P. Mohapatra, J. C.V, and M. P. Kumar, "Learning to round for discrete labeling problems," in *Proc. Int. Conference on Artificial Intelligence and Statistics*, ser. Proc. of Machine Learning Research, A. Storkey and F. Perez-Cruz, Eds., vol. 84, Apr. 2018, pp. 1047–1056.

[102] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proc. ACM Workshop on Hot Topics in Networks*, 2016, pp. 50–56.

[103] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with Pensieve," in *Proc. ACM SigComm*, 2017, pp. 197–210.

[104] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to route," in *Proc. ACM Worksh. Hot Topics in Netw.*, 2017, pp. 185–191.

[105] D. Bertsimas, D. B. Brown, and C. Caramanis, "Theory and applications of robust optimization," *SIAM Rev.*, vol. 53, no. 3, pp. 464–501, Aug. 2011.

[106] J. C. Spall, *Introduction to Stochastic Search and Optimization*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 2003.

[107] P. Kalmbach, J. Zerwas, P. Babarczi, A. Blenk, W. Kellerer, and S. Schmid, "Empowering self-driving networks," in *Proc. ACM SIGCOMM 2018 Workshop on Self-Driving Netw.*, 2018, pp. 8–14.

[108] C. Salge, C. Glackin, and D. Polani, "Empowerment - an Introduction," *CoRR*, vol. abs/1310.1863, 2013.

[109] A. S. Klyubin, D. Polani, and C. L. Nehaniv, "Empowerment: a universal agent-centric measure of control," in *Proc. IEEE Congress on Evolutionary Computation*, vol. 1, Sep. 2005, pp. 128–135.

[110] Phillipe Capdepuy, "Informational Principles of Perception-Action Loops and Collective Behaviours," Ph.D. Dissertation, University of Hertfordshire, Mar. 2010.

[111] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "A control theoretic analysis of RED," in *Proc. IEEE Infocom*, vol. 3, Apr. 2001, pp. 1510–1519.

[112] S. Keshav, "A control-theoretic approach to flow control," *SIGCOMM Comput. Commun. Rev.*, vol. 21, no. 4, pp. 3–15, Sep. 1991.

[113] M. Karl, M. Soelch, P. Becker-Ehmck, D. Benbouzid, P. van der Smagt, and J. Bayer, "Unsupervised real-time control through variational empowerment," *ArXiv, vol. 1710.05101*, Oct. 2017.

[114] C. Salge, C. Glackin, and D. Polani, "Changing the environment based on empowerment as intrinsic motivation," *CoRR*, vol. abs/1406.1767, 2014.

[115] A. S. Klyubin, D. Polani, and C. L. Nehaniv, "Keep your options open: An information-based driving principle for sensorimotor systems," *PLOS ONE*, vol. 3, no. 12, pp. 1–14, 12 2008.

[116] S. Jia, X. Jin, G. Ghasemiesfeh, J. Ding, and J. Gao, "Competitive analysis for online scheduling in software-defined optical WAN," in *Proc. IEEE INFOCOM*, 2017, pp. 1–9.

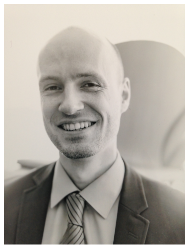
[117] S. Das, G. Parulkar, and N. McKeown, "Rethinking IP core networks," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 5, no. 12, pp. 1431–1442, Dec. 2013.

[118] M. Kuźniar, P. Perešíni, D. Kostić, and M. Canini, "Methodology, measurement and analysis of flow table update characteristics in hardware openflow switches," *Comp. Netw.*, vol. 136, pp. 22–36, May 2018.

[119] K. Gregor, D. J. Rezende, and D. Wierstra, "Variational intrinsic control," *CoRR*, vol. abs/1611.07507, 2016.

Wolfgang Kellerer (S96A02M03SM11) received the Dipl.-Ing. (Masters) and Dr.-Ing. (Ph.D.) degrees from the Technical University of Munich, Germany, in 1995 and 2002, respectively, where he is a Full Professor, heading the Chair of Communication Networks with the Department of Electrical and Computer Engineering. He was with European Research Laboratories, NTT DOCOMO, for ten years in leading positions, contributing to research and standardization of LTE-A and 5G technologies. In 2001, he was a Visiting Researcher with the Information Systems Laboratory, Stanford University, CA, USA. His research has resulted in over 200 publications and 35 granted patents. He was awarded with an ERC Consolidator Grant from the European Commission for his research project FlexNets Quantifying Flexibility in Communication Networks in 2015. He currently serves as an Associate Editor for the *IEEE Transactions on Network and Service Management* and on the Editorial Boards of the *IEEE Communications Surveys and Tutorials* and the *IEEE Networking Letters*. He is a member of ACM and VDE ITG.

Patrick Kalmbach completed his Bachelor of Science in the course Applied Computer Science at the Baden-Wuerttemberg Cooperative State University Stuttgart in September 2014. His bachelor studies were in corporation with the Hewlett-Packard Company. He continued his studies at the Technical University Munich (TUM), and graduated as Master of Science in April 2017. Since then, he works as teaching and research associate at the Chair of Communication Networks at the TUM. He is currently pursuing his PhD as a member of the software-defined networking and network virtualization research group. His research is focused an the application of machine learning for network security, network function virtualization and resource allocation in virtualized environments.



Andreas Blenk received his diploma degree in computer science from the University of Wuerzburg, Germany, in 2012 and the Dr.-Ing. degree (Ph.D.) from TUM in 2018. He joined the Chair of Communication Networks at TUM in June 2012, where he is currently working as a senior researcher and associate lecturer. His research is focused on self-driving, flexible and predictable virtual and software-defined networks, as well as data-driven networking algorithms.

Arsany Basta received the M.Sc. degree in Communication Engineering and the Dr.-Ing. (Ph.D.) degree in Communication Networks from the Technical University of Munich in 2012 and 2017, respectively. Since then, he is since a senior researcher at the Chair of Communication Networks, where his current research activities focus on the application of Software Defined Networking (SDN), Network Virtualization (Slicing), and Network Functions Virtualization (NFV) to mobile networks toward the next generation 5G.

Martin Reisslein (S'96-M'98-SM'03-F'14) is a Professor in the School of Electrical, Computer, and Energy Engineering at Arizona State University (ASU), Tempe. He received the Ph.D. in systems engineering from the University of Pennsylvania, Philadelphia, in 1998. He currently serves as Associate Editor for the *IEEE Transactions on Mobile Computing*, the *IEEE Transactions on Education*, and *IEEE Access* as well as *Computer Networks*. He is Associate Editor-in-Chief of the *IEEE Communications Surveys & Tutorials*, Co-Editor-in-Chief of *Optical Switching and Networking*, and chairs the steering committee of the *IEEE Transactions on Multimedia*.

Stefan Schmid Stefan Schmid is a Professor at University of Vienna, Austria. He received his MSc (2004) and PhD degrees (2008) from ETH Zurich, Switzerland. In 2009, Stefan Schmid was a postdoc at TU Munich and the University of Paderborn, between 2009 and 2015, a senior research scientist at the Telekom Innovations Laboratories (T-Labs) in Berlin, Germany, and afterwards an Associate Professor at Aalborg University. His research interests revolve around the fundamental and algorithmic problems of networked and distributed systems.