Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks

Stanislav Lange, Steffen Gebert, Thomas Zinner, Phuoc Tran-Gia,
David Hock, Michael Jarschel, and Marco Hoffmann

***Abstract*—Software Defined Networking (SDN) marks a paradigm shift towards an externalized and logically centralized network control plane. A particularly important task in SDN architectures is that of controller placement, i.e., the positioning of a limited number of resources within a network to meet various requirements. These requirements range from latency constraints to failure tolerance and load balancing. In most scenarios, at least some of these objectives are competing, thus no single best placement is available and decision makers need to find a balanced trade-off. This work presents POCO, a framework for Pareto-based Optimal COntroller placement that provides operators with Pareto optimal placements with respect to different performance metrics. In its default configuration, POCO performs an exhaustive evaluation of all possible placements. While this is practically feasible for small and medium sized networks, realistic time and resource constraints call for an alternative in the context of large scale networks or dynamic networks whose properties change over time. For these scenarios, the POCO toolset is extended by a heuristic approach that is less accurate, but yields faster computation times. An evaluation of this heuristic is performed on a collection of real world network topologies from the Internet Topology Zoo. Utilizing a measure for quantifying the error introduced by the heuristic approach allows an analysis of the resulting trade-off between time and accuracy. Additionally, the proposed methods can be extended to solve similar virtual functions placement problems which appear in the context of Network Functions Virtualization (NFV).**

***Index Terms*—SDN, NFV, controller placement, POCO, OpenFlow, resilience, failure tolerance, latency, multiobjective optimization, simulated annealing.**

I. Introduction

With the introduction of Software Defined Networking (SDN), a paradigm shift in communication networks has been set in motion. This shift is directed towards a logically centralized architecture which separates control and data plane, and thus allows moving control plane functions from network devices to dedicated controller instances running in software. Currently, OpenFlow [1] constitutes the most popular SDN-enabling communications protocol between control and data plane, thus allowing communication over the southbound API [2]. In the OpenFlow architecture, a logically centralized *controller* manages switches by providing them with rules that dictate their packet handling behavior. To cover aspects like scalability and resilience, concepts like HyperFlow [3] allow partitioning of OpenFlow networks into multiple domains that are each handled by individual controllers.

This work discusses key issues arising in architectures with an externalized control plane and presents mechanisms for overcoming them. The first relevant aspect is the number of SDN controllers required for a reliable and resilient network operation. Second, the position of each controller in the network affects competing objectives like inter-controller latency, switch-controller latency, and resilience. Thus, finding a controller placement featuring an adequate trade-off between the goals that are relevant for a particular use case is crucial for an efficient operation. Furthermore, to cope with large scale networks and dynamically changing network conditions like traffic patterns or bandwidth demands [4], a computationally fast approach for the aforementioned placement problem is required.

In [5], the SDN controller placement problem is addressed for the first time. The basic case of optimizing the latency from nodes to their assigned controller is traced back to to the *facility location problem*, which is known to be NP-hard. Instead of resorting to approximations, the authors argue that an exhaustive evaluation of the entire solution space is practically feasible for realistic networks. Thus, there is a guarantee for finding optima with respect to the latency. These optima are used to derive guidelines for dimensioning of the control plane. For example, most of the investigated topologies require only one controller to comply with realistic latency constraints. In our previous work, the controller placement analysis is extended with various resilience aspects that are particularly important in SDN scenarios [6]. Further investigations in [6] cover additional objectives like load balancing among controllers and inter-controller latencies to gain insights into the trade-offs between different placement choices as the different objectives are usually competing. This functionality is bundled in POCO, a Matlab-based framework capable of computing resilient Pareto-based Optimal COntroller placements. Its efficient CPU and RAM utilization allows for an exhaustive evaluation of the

Manuscript received November 28, 2014; revised February 2, 2015; accepted February 5, 2015. Date of publication February 10, 2015; date of current version March 17, 2015. This work has been performed in the framework of the CELTIC EUREKA project SASER-SIEGFRIED (Project ID CPP2011/2-5), and it is partly funded by the BMBF (Project ID 16BP12308). The associate editor coordinating the review of this paper and approving it for publication was F. De Turck.

S. Lange, S. Gebert, T. Zinner, and P. Tran-Gia are with the Institute of Computer Science, Chair of Communication Networks, University of Würzburg, 97074 Würzburg, Germany (e-mail: stanislav.lange@informatik.uni-wuerzburg.de; steffen.gebert@informatik.uni-wuerzburg.de; zinner@informatik.uni-wuerzburg.de; trangia@informatik.uni-wuerzburg.de).

D. Hock is with Infosim GmbH & Co. KG, 97076 Würzburg, Germany (e-mail: david.hock@infosim.net).

M. Jarschel and M. Hoffmann are with Nokia, 81541 Munich, Germany (e-mail: michael.jarschel@nsn.com; marco.hoffmann@nsn.com).

Digital Object Identifier 10.1109/TNSM.2015.2402432

entire solution space of small and medium sized networks within seconds, even in the presence of resilience considerations. On the one hand, this eliminates the necessity to impose constraints on objective values *a priori* which might result in excluding feasible alternatives or even a problem instance without admissible solutions. On the other hand, the multiobjective approach allows for a clearer illustration of the trade-offs between competing criteria. Additionally, the decision maker's preferences with respect to the different objectives usually depend on available alternatives [7]. This, in turn, decreases the feasibility of transformations to single objective optimization problems, e.g., via a weighted combination of individual objectives.

In the context of large scale WANs, operators face various challenges like dynamically changing network conditions that affect the network's overall performance. These changes not only appear on a regular and predictable basis as in the case of day and night cycles, but also on shorter time scales, e.g., hourly. To adapt to these dynamics and assure a smooth operation, placements which take into account the current network situation need to be calculated. The faster this calculation can be performed, the faster a switch to another placement can be triggered. As demonstrated in [8], POCO is already capable of providing an exhaustive evaluation of placements within seconds for small and medium sized instances. However, an exhaustive evaluation of large scale networks has time requirements in the order of magnitude of tens of minutes, which might not be sufficient to cope with the described dynamics. This work explores heuristic approaches to the multiobjective controller placement problem. While such approaches do not guarantee to find optimal solutions, they are capable of yielding results significantly faster than their exhaustive counterpart.

Building on the insights gained from exhaustive evaluations of medium sized real world network topologies, the POCO framework is extended with a heuristic approach from the domain of multiobjective combinatorial optimization (MOCO), namely Pareto Simulated Annealing (PSA) [9]. With this mechanism, POCO is able to handle huge network instances without sacrificing much accuracy. Several measures of accuracy from the MOCO domain are investigated and the time-accuracy trade-off of the heuristic approach is quantified for numerous realistic network topologies for which the actual optima are available from the exhaustive evaluation. These results are used to infer guidelines regarding the parameter choice in the context of huge problem instances where reference values are not available. Further improvements to the POCO framework include an extended filtering mechanism as well as the possibility to visualize up to four dimensions of the solution space, thus allowing for an analysis of the interrelations between up to four different objectives.

This work is structured as follows. Section II provides an overview of the SDN scenario discussed in this article as well as an outlook regarding possible extensions towards the related NFV functions placement problem. Then, a formal definition of relevant metrics and the resulting optimization problem are presented. The POCO framework and extensions to its functionality are outlined in Section III. Section IV presents the Pareto Simulated Annealing algorithm, its adaptation to the controller placement problem, and details of the evaluation scheme employed in this work. Evaluation results are shown in Section V. Section VI covers relevant related work on the mathematical background of the presented problem as well as work related to the controller placement problem in particular. Finally, Section VII concludes the paper.

II. Scenarios and Problem Statement

This section first introduces the most important aspects of SDN as well as scenarios in which the presented optimization approach is useful. After a brief overview of possible applications in the related field of NFV functions placement, a formal problem definition is given alongside necessary notation.

A. SDN, NFV, and Scenarios

The key principle of SDN consists of the separation of data plane and control plane. The design of the externalized control plane can be performed in various ways. On the one hand, there is a choice between controller architectures, e.g., featuring a single, centralized controller, multiple equally important controllers responsible for partitions of the network [10], or a set of distributed controllers arranged in a hierarchy [11]. On the other hand, purely software-based or hardware solutions for the control plane allow trade-offs between flexibility, performance, and cost aspects. Furthermore, connections involving controllers may be realized inband or outband, i.e., either sharing the same physical links as data plane traffic or utilizing dedicated lines.

Typically, core networks contain pre-configured primary and backup paths for the aggregated traffic between different nodes in the network. Thus, there is no need for communication with the SDN controllers regarding each individual TCP flow but only in case of certain occasions like outages or traffic management actions. In such environments, a single controller can be sufficient for a viable network operation. Nonetheless, growing network sizes and the importance of resilience against failures increase the amount of required controllers. Moreover, network functions that are deployed on the SDN control plane further raise its load, leading to even higher numbers of required controllers. For the remainder of this paper, it is assumed that all sites possess the capability to run a software-based SDN controller. Furthermore, communication with SDN controllers is assumed to happen inband, i.e., via the same physical links as regular traffic.

Similar to the controller placement problem in the SDN domain, technologies like Network Functions Virtualization (NFV) [12] call for appropriate algorithms that can tackle challenges beyond the single criterion placement problems as discussed in [13]. These problems might introduce additional complexity due to possible interdependencies between the network functions as in the case of function chaining and potential new constraints regarding additional aspects like security. While previous work illustrates how POCO can be extended to work in an NFV functions placement setting [14], this work focuses on the SDN controller placement problem and investigates strategies for dealing with huge problem instances.

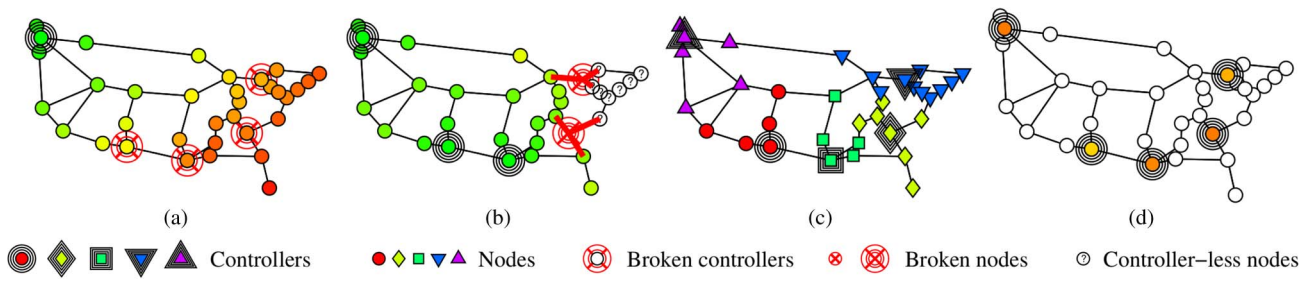


Fig. 1. Assessing the quality of controller placements with different objective measures. (a) Latencies when controllers fail; (b) isolated nodes; (c) controller load imbalance; (d) inter-controller latency.

B. (Resilient) Controller Placement Problem

While minimizing latencies between each node and its assigned controller constitutes a crucial aspect of the controller placement problem, there are numerous other, possibly competing, objectives that require consideration. In the following, objectives that are covered in this work are presented along with examples that motivate their necessity in different use cases. The Internet2 OS3E network is used as example topology and the best placement with respect to maximum node to controller latency for $k=5$ controllers as shown in the work of Heller *et al.* [5] acts as reference. Fig. 1 displays this placement's performance when evaluated with respect to different objective measures and different conditions, e.g., latencies and load balance in the presence of node or controller failures. In contrast to the work of Zhang *et al.* [15], this work assumes that the node to controller assignment can change when failures occur.

Fig. 1(a) illustrates the latency between each node and its assigned controller when multiple controllers stop working. Each node's color indicates its latency to the closest functioning controller. The color is normalized with the graph's diameter with green representing a latency of zero, yellow representing a latency that corresponds to 50% of the graph's diameter, and red indicating a latency of 100%. While the distributed controller structure asserts low latencies in the failure-free case [5], the illustrated failure scenario highlights the fact that in the presence of failures, the position of each controller matters. The only controller that is not affected by the failure is located at the edge of the network and thus, control traffic of many nodes needs to traverse almost the entire network. In order to better cope with scenarios of this kind, the optimization mechanism for resilient controller placements should consider failure scenarios. Depending on the specific use case, average and maximum latencies might be useful measures.

In addition to controller failures caused by software issues, physical network elements may also suffer from hardware problems and stop working. This type of failure has more severe consequences as it induces changes in the topology, which in turn results in changes of shortest paths. Thus, node to controller assignments tend to change and in extreme cases, the network graph can even become disconnected. While the nodes in each connected component are still fully operational, not being able to connect to any controller in the connected component prohibits any functionality beyond forwarding according to previously installed rules. Fig. 1(b) displays the case of two network nodes failing at the same time, which results in both of the aforementioned phenomena. First, the graph is decomposed into two disjoint components. Second, the right part of the graph does not contain a controller. Thus, nodes in this part of the graph lose access to any functionality realized by the controller. These nodes are represented by question mark icons ⑦.

Assuming that nodes connect to their nearest controller, certain placements tend to result in imbalanced assignments, i.e., some controllers provide instructions for significantly more nodes than others. Consequently, environments with a high intensity or frequency of control plane communication can run into problems, like increased delays, due to queueing at controller instances. Fig. 1(c) illustrates the imbalance aspect of the latency-optimal placement. Each node is colored and shaped according to the controller it is assigned to. While the blue controller is responsible for ten network elements, the green and red controllers need to manage just four nodes, i.e., less than half as many. Additionally, our previous investigations show [16] that for some controller implementations, the number and order of connected switches might cause unfairness with respect to aspects like the switches' flow setup times. Thus, load balancing should be a part of the decision criteria when choosing a controller placement for various types of SDN and NFV setups. Additionally, the *link assignment* task presented in [10] corresponds to minimizing the imbalance, further supporting the relevance of this aspect.

Previously discussed scenarios, especially those involving failure tolerance, indicate the need for a distributed control plane. However, such an architecture also requires various forms of state synchronization between the individual controllers. This ensures proper functionality in case of outages and allows making decisions that are not limited to a local view on a part of the network. Therefore, another goal of the controller placement task is to maintain a small inter-controller latency to minimize synchronization times. A visualization of this measure is provided in Fig. 1(d), where each controller is colored according to the distance to the controller that is farthest away from it. Like in Fig. 1(a), the distances are normalized with respect to the graph's diameter. For most controllers, the shown placement results in high maximum latencies to other controllers which might be not acceptable for certain use cases. Therefore, inter-controller latency is part of POCO's set of analyzed objectives. Additionally, the pair of inter-controller latency and node to controller latency constitutes a set of competing objectives. While a tight cluster of controllers results in low inter-controller latencies and high node to controller latencies, a spatially widespread distribution of controllers leads to the opposite. Such relationships between

objectives are the motivation for the analysis of Pareto optimal placements in POCO, which allows decision makers to express their preferences after inspecting possible placements.

Furthermore, the set of objectives taken into account is not restricted to those presented in this section and can be extended according to use case specific requirements. These also include management aspects which could be introduced as separate constraints. For example, metrics like the expected service quality provided by a network in the context of a given placement could be added into the evaluation and decision process.

C. Notation

Formally speaking, the controller placement problem is a multiobjective combinatorial optimization (MOCO) problem. The network is represented as a graph $G = (V, E)$ with node set V containing n nodes which are connected by edges from the set E. Additionally, shortest path latencies between each pair of nodes are stored in a distance matrix D, where $d_{i,j}$ denotes the latency from node i to node j. Latencies in D are normalized with the graph's diameter, i.e., $d_{i,j} \in [0, 1]$. Given the desired number of controllers k, there is a finite set of $\binom{n}{k}$ possible placements, hence the term *combinatorial optimization*. The goal of the MOCO task is to find controller placements from the set of size k placements $\mathcal{P}_k = \{\mathcal{P} \in 2^V \mid |\mathcal{P}| = k\}$ that are Pareto optimal with respect to various objective functions $f_i (i \in \{1, \ldots, J\})$. These were discussed informally in the previous section and are presented in detail in the following. A placement x is considered Pareto optimal, if and only if there is no placement y such that $\forall i \ f_i(y) \leq f_i(x)$ and $f_i(y) < f_i(x)$ for at least one i. The set of all Pareto optimal solutions is referred to as Pareto frontier.

While POCO performs an exhaustive evaluation of all $\binom{n}{k}$ possible placements for small and medium sized networks, a heuristic approach is suggested in case of instances that are too huge to be fully evaluated in a practical time frame. While an exhaustive evaluation of the former is performed in a matter of seconds using POCO, the size of the search space grows very quickly. Thus, when increasing the desired number of controllers beyond 7, an exhaustive evaluation of a network with 50 nodes can take between several minutes and hours on a machine with an Intel Core i7 4770 CPU at 3.40 GHz and 16 GB of RAM. Although such proportions may not be realistic in the context of SDN-based networks, NFV scenarios might require higher numbers of functions that need to be distributed in the network. To quantify the loss of accuracy caused by switching to the heuristic approach, different measures for the difference between the actual and the estimated Pareto frontier have been adopted from [9]. In the following, R denotes the original Pareto frontier which is used as reference, and M represents the estimate provided by the heuristic approach. Before the distance between Pareto frontiers is defined, a distance metric for two placements is introduced. According to (1), $c(x, y)$ defines the distance between two placements as the maximum weighted difference between individual objective values achieved by the placements. The weight w_j corresponds to objective f_j's range and is used for normalization, i.e., $c(x, y) \in [0, 1]$. Adding zero to the argument of the maximum asserts that no negative distance is returned. With this distance metric, it is possible to define measures for the distance between two Pareto frontiers, of which one is known to be better than the other and is therefore used as reference. The first metric, δ_1, is shown in (2) and measures the average distance between each element in R and its closest element from M. While δ_1 measures the average difference and may hide outliers, δ_2 considers the maximum distance between each element from R and its closest element in M. Its formal definition is provided in (3) and allows for a worst case analysis of the estimate M.

$$c(x, y) = \max_{j=1,\ldots,j} \{0, w_j \left(f_j(x) - f_j(y)\right)\} \tag{1}$$

$$\delta_1(R, M) = \frac{1}{|R|} \sum_{y \in R} \left\{ \min_{x \in M} \{c(x, y)\} \right\} \tag{2}$$

$$\delta_2(R, M) = \max_{y \in R} \left\{ \min_{x \in M} \{c(x, y)\} \right\} \tag{3}$$

As motivated in Section II-B, numerous competing objective functions need to be considered when evaluating a given controller placement. This section provides formal definitions of the objective functions that are available in the POCO framework. First, the node to controller latency provides information on the connectivity between each node and its assigned controller. Similar to δ_1 and δ_2, latency measures can be analyzed either by calculating the average across all latencies in the examined placement or by taking the maximum value for a worst case analysis. For a placement $\mathcal{P} \in 2^V$ and distance matrix D, the maximum node to controller latency $\pi^{\text{max latency}}$ can be defined according to (4). In an analogous fashion, the average node to controller latency $\pi^{\text{avg latency}}$ is determined as per (5).

$$\pi^{\text{avg latency}}(\mathcal{P}) = \max_{v \in V} \min_{p \in \mathcal{P}} d_{v,p} \tag{4}$$

$$\pi^{\text{avg latency}}(\mathcal{P}) = \frac{1}{|V|} \sum_{v \in V} \left(\min_{p \in \mathcal{P}} d_{v,p} \right) \tag{5}$$

When resilience with respect to controller outages is part of the analysis, additional calculations are necessary. Let $\mathcal{C} = 2^{\mathcal{P}} \setminus \{\emptyset\}$ denote all alternative placements that result from the failure of up to $k - 1$ controllers. Then, $\pi_{\mathcal{C}}^{\text{max latency}}$ denotes the maximum node to controller latency which, in addition to the failure free case, also accounts for any failure scenario that spares at least one controller. Equation (6) shows the formal definition of $\pi_{\mathcal{C}}^{\text{max latency}}$ as well as the average-based $\pi_{\mathcal{C}}^{\text{avg latency}}$.

$$\begin{aligned} \pi_{\mathcal{C}}^{\text{max latency}}(\mathcal{P}) &= \max_{P \in \mathcal{C}} \max_{v \in V} \min_{p \in P} d_{v,p} \\ \pi_{\mathcal{C}}^{\text{avg latency}}(\mathcal{P}) &= \frac{1}{|\mathcal{C}|} \sum_{P \in \mathcal{C}} \left(\frac{1}{|V|} \sum_{v \in V} \left(\min_{p \in P} d_{v,p} \right) \right) \end{aligned} \tag{6}$$

With the above definition of the node to controller latency, the definition of inter-controller latency follows by means of analogy. For a given placement $\mathcal{P}$ and any pair of controllers p_1, p_2 in this placement, the inter-controller latency $\pi^{\text{controller-latency}}(\mathcal{P})$ can be defined either with respect to the

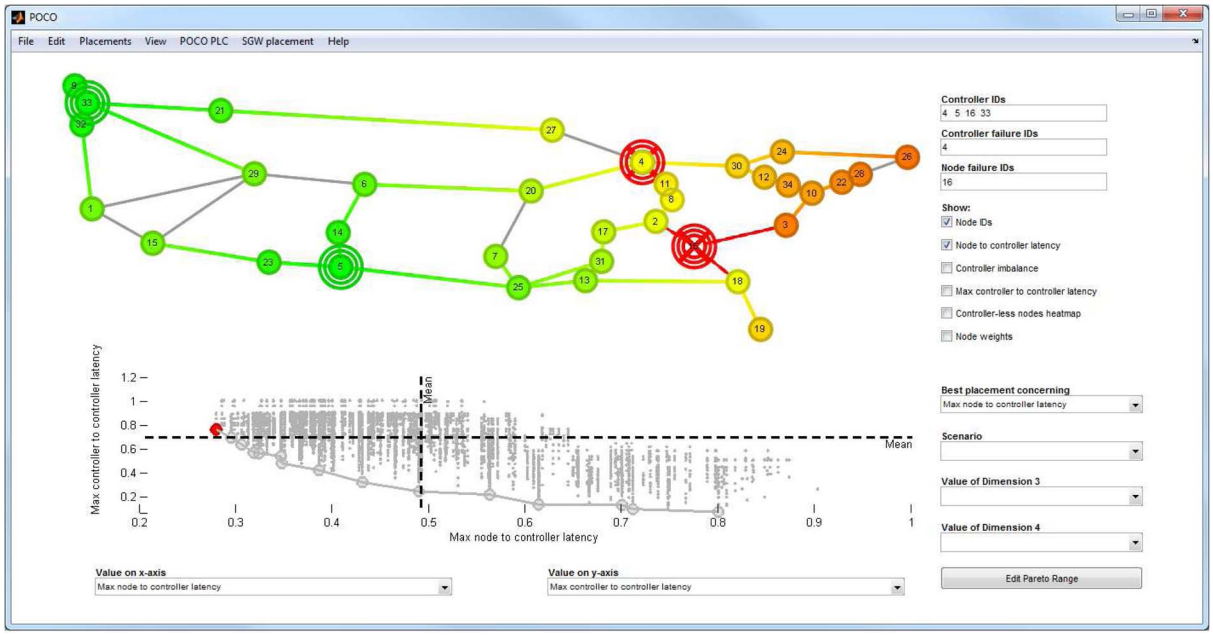


Fig. 2. POCO GUI displaying the current placement with the latency to the controller at the top and the whole solution space at the bottom.

maximum or with respect to the average latency between p_1 and p_2. A formal representation of these relationships is presented in (7).

$$\pi^{\text{max controller-latency}}(\mathcal{P}) = \max_{p_1,p_2\in\mathcal{P}} d_{p_1,p_2}$$
$$\pi^{\text{avg controller-latency}}(\mathcal{P}) = \frac{1}{\binom{|\mathcal{P}|}{2}} \sum_{p_1,p_2\in\mathcal{P}} d_{p_1,p_2} \tag{7}$$

While metrics regarding latency strive for short communication paths, controller load balance considerations also need to be taken into account when a reliable network operation is desired. To comply with the problem definition and previous metrics, the imbalance metric is introduced rather than a balance metric so that the goal is to minimize the metric's value. For each placement $\mathcal{P}$ and controller p, the total number of nodes that are assigned to p when each node connects to its closest controller is defined as n_p. The imbalance metric $\pi^{\text{imbalance}}$ captures the difference in n_p for the two controllers with the lowest and highest amount of assigned nodes, respectively. Additionally, imbalance in the presence of failures can be quantified by analyzing imbalance in assignments that result from different failure scenarios. In these cases, n_p^s indicates the number of nodes assigned to controller p when failure scenario s occurs. Equation (8) defines both imbalance metrics. The indices $\emptyset$ and $\mathcal{X}$ denote the failure free case and the set of considered failure scenarios, respectively. Furthermore, the imbalance metrics can be normalized by division with $|V|$ as this is the maximum amount of nodes that can be assigned to a single controller in the worst case.

$$\pi_{\emptyset}^{\text{imbalance}}(\mathcal{P}) = \max_{p\in\mathcal{P}} n_p^{\emptyset} - \min_{p\in\mathcal{P}} n_p^{\emptyset}$$
$$\pi_{\mathcal{X}}^{\text{imbalance}}(\mathcal{P}) = \max_{s\in\mathcal{X}} \left(\max_{p\in\mathcal{P}} n_p^s - \min_{p\in\mathcal{P}} n_p^s \right) \tag{8}$$

Finally, node and link failures can lead to a decomposition of the network graph, thus isolating nodes from all controllers. As discussed in Section II-B, nodes that are not connected to a controller have very limited functionality and are therefore undesired. Hence, $\pi_{\mathcal{X}}^{\text{controller-less}}(\mathcal{P})$ computes the maximum amount of such nodes for any failure scenario specified in $\mathcal{X}$. The calculation of this metric is performed using a connectivity matrix E^s whose entries $e_{i,j}^s$ are equal to 0, if and only if node i can reach node j in failure scenario s and 1 otherwise. As controller outages that spare at least one controller do not affect the amount of controller-less nodes, only node and link failures are considered for $\pi^{\text{controller-less}}$. These scenarios are summarized in the set $\mathcal{N}$, finally yielding (9).

$$\pi_{\mathcal{N}}^{\text{controller-less}}(\mathcal{P}) = \max_{s\in\mathcal{N}} \sum_{v\in V} \min_{p\in\mathcal{P}} e_{v,p}^s \tag{9}$$

III. POCO's User Interface

POCO does not only evaluate the possible placements for a given topology and number of controllers but is also capable of providing a visualization of the corresponding solution space in a graphical user interface. The motivation behind is to give also novice users an easy access to the implemented mechanisms for calculating and evaluating controller placements.

A. Introduction to the POCO GUI

The POCO GUI is, as POCO itself, implemented in Matlab and available as open source software [17], [18]. Users can choose between metrics for investigation and are presented with a plot displaying the performance of all possible placements with regard to the chosen metrics. Each placement is represented as a point in the plot whose x and y-axes denote its metric value. Additionally, points located on the Pareto frontier with respect to these metrics are highlighted and connected

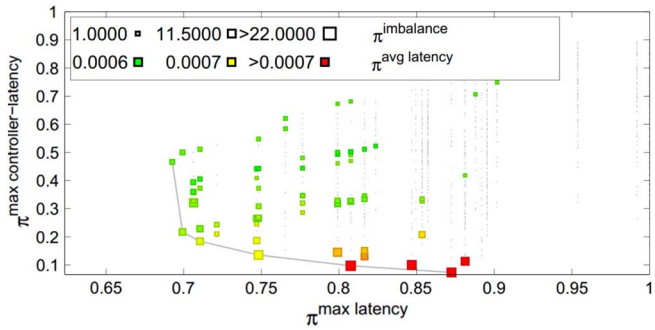


Fig. 3. Visualization of the 4-dimensional Pareto frontier in POCO.

with line segments. Fig. 2 shows an example session of the POCO GUI featuring the Internet2 OS3E topology with $k = 4$ controllers. The top half shows the topology including the currently selected placement by highlighting controllers with double circles. The color of each node, based on a traffic light color scheme, denotes the latency to the node's controller, which is selected based on the lowest distance. Additionally, the implications of a failed node (#16) and a failed controller (#4) for this particular placement are illustrated. Further information can be added to the representation of the placement. Check boxes on the right hand side allow enhancing the graphic with information regarding the number of controller-less nodes or different latency measures as well as applying a vertex coloring to visualize node to controller assignments for imbalance analyses. In the bottom part of the interface, the Pareto frontier of all possible placements with respect to the maximum node to controller (x-axis) and controller to controller latencies (y-axis) is displayed. By clicking on any point in the Pareto plot, the detailed visualization at the top is updated according to the selected placement corresponding to the clicked point. This allows the decision maker to interactively explore the search space according to her/his preferences and current use case.

B. Improvements to the POCO GUI

In order to allow an even more in-depth analysis of the set of placements, POCO's GUI [18] is extended with options to display up to four dimensions of the solution space. This is achieved by applying different transformations to the size and color of the points presented at the bottom of Fig. 2. First, the set of Pareto optimal placements with regard to the four chosen metrics are calculated. Then, minimum and maximum values for the two additional metrics are computed. Using the traffic light color scheme as in Figs. 1 and 2, the points' colors and sizes are adapted so that they reflect the placements' performance with regard to the additional metrics. Fig. 3 illustrates this new, four dimensional visualization of the scenario discussed in the previous figure. While the maximum node to controller latency $\pi^{\text{max latency}}$ and maximum inter-controller latency $\pi^{\text{max controller-latency}}$ remain on the x and y-axes, the imbalance of number of assigned nodes per controller $\pi^{\text{imbalance}}$ and the average node to controller latency $\pi^{\text{avg latency}}$ are included via the size and color of the points. This perspective shows further trade-offs and interdependencies between metrics, and thus allows the decision maker to make a well-founded choice reflecting her/his preferences for a particular use case.

Through the new functionality "Edit Pareto Range", a multi-step filtering of the results can be achieved. After investigating the Pareto frontier for a certain combination of metrics and the performance of all placements, the user can filter out placements that exceed a certain threshold. These placements are then removed from the stored result set of placements so that switching to a view showing a different combination of metrics will not display these placements again. In this next step, the user can further reduce the number of Pareto optimal results, in order to finally set a trade-off and choose between only a little number of Pareto optimal results that fits best for the objectives important for the particular use case.

Using this improved mechanism, it is possible for a network engineer to set thresholds for metrics after investigating the whole solution space instead of having to define upper bounds before starting the computation.

IV. Methodology

Besides improvements to the user interface of POCO, including the four dimensional visualization of the Pareto results, the main contribution of this work is the analysis of the trade-off between accuracy and cost with respect to time and memory resources when employing heuristic methods or performing an exhaustive evaluation to solve the controller placement problem. In order to incorporate the heuristic mechanisms into the POCO framework, guidelines for deciding whether to use an exhaustive evaluation or switch to a heuristic approach are derived. Furthermore, the influence of different parameters of heuristic algorithms is investigated in order to infer viable parameter values for different use cases and requirements. The heuristic algorithm proposed in this work is based on Pareto Simulated Annealing [9] and analyzes are performed on numerous realistic network topologies from the Internet Topology Zoo [19].

After an overview of the Pareto Simulated Annealing algorithm, the evaluation procedure used in this work is described.

A. Pareto Simulated Annealing

In its default configuration, POCO performs an exhaustive evaluation of all possible placements for a given network topology and desired number of controllers in order to find the Pareto optimal placements with respect to the metrics defined in the previous section. Although this guarantees finding all Pareto optima, the time and memory requirements of an exhaustive evaluation increase with the size of the search space which is proportional to $\binom{n}{k}$, the number of possible placements of size k in a network consisting of n nodes. Even for relatively small n, this number rises drastically when the number of controllers, k, approaches $\frac{n}{2}$, e.g., $\binom{34}{4} = 46.376$ while $\binom{34}{12} = 548.354.040$. When performing just a single network planning task before deployment, an exhaustive evaluation also for bigger instances is justified even if it requires a high computational effort and a large amount of time. However, in the context of a dynamic and flexible network that needs to adapt to changes in the

environment and usage patterns, time is a limiting factor. Scenarios of up to 50% of the nodes running an SDN controller are certainly not realistic. However, in the field of NFV, use cases where a function is running at half of the nodes are likely to appear. Such computations are still in scope of the presented algorithms as well as the POCO software.

In order to provide viable solutions while meeting given time constraints, a heuristic approach is integrated into POCO. Such mechanisms explore only a subset of the search space and return the Pareto frontier of this subset. The exploration techniques as well as the number of visited solutions are the key influence factors for algorithms from this family.

In the context of finding the global optimum of a function that has a large domain, i.e., the optimization problem has a large search space, simulated annealing [20] is a popular heuristic approach. Simulated annealing is a Monte Carlo method and has two distinctive properties. First, during the exploration of the search space, moves to solutions worse than the current one are permitted in order to avoid getting stuck in a local optimum. This is achieved by incorporating a control parameter that is referred to as temperature, which determines the probability of accepting such moves. Second, the probability of moving to a worse solution gradually decreases with the number of iterations. Additionally, the acceptance probability depends on the difference between the objective values of the current and the proposed solution. The rationale behind this behavior is that accepting rather bad solutions at the beginning allows for a broader coverage of the search space while the lower acceptance probability at the end helps with convergence. However, simulated annealing does not support optimization problems with multiple objectives.

While there are many different options in the domain of multiobjective combinatorial optimization (MOCO), POCO was extended with Pareto simulated annealing (PSA), a MOCO algorithm inspired by simulated annealing. This decision is based on multiple criteria. First, PSA lends itself to an efficient implementation in Matlab and provides a Pareto frontier of explored solutions as its output. Second, PSA incorporates mechanisms that assert that the resulting output has a high degree of dispersion, i.e., that Pareto optima with respect to different objectives are found. This aspect of PSA is similar to the notion of *recall* in the domain of information retrieval. Recall is used to quantify the ratio between the amount of documents relevant to a given query that are returned by a search algorithm and the total amount of relevant documents. Third, PSA is an anytime algorithm and can thus provide a set of solutions at any time. Due to this property, it is not necessary to find algorithm parameters that fit with particular time constraints. Instead, it is possible to just run the algorithm and stop it when results are needed. Finally, a heuristic approach based on simulated annealing has been successfully applied to a related single objective problem [4], demonstrating its feasibility in the controller placement context.

The PSA procedure used in this work is based on the algorithm presented in [9]. Algorithm 1 outlines the structure of the developed approach. The input consists of two parts. On the one hand, there is problem specific data like the topology graph G and the desired number of controllers k. On the other hand, there are parameters for the PSA mechanism. These include the number of placements to evaluate during each iteration s, the number of iterations per temperature level m, as well as T_0 and ρ which control the *annealing schedule*, i.e., the initial temperature and the rate of temperature decrease. Initially, a set S of s random placements of size k is generated. For each combination of placement and objective, random weights Λ are assigned. Later, these weights will help achieving the dispersion property discussed in the previous paragraph. During the whole procedure, the Pareto frontier of all visited placements M, as well as the corresponding placements are updated.

Algorithm 1 Pareto Simulated Annealing

input: $G = (V, E)$, k, s, m, T_0, ρ
$n = |V|$
$S = generateRandomPlacements(n, s, k)$
$\Lambda = generateRandomWeights(S)$
$M = paretoFrontier(evaluatePlacements(S))$
$T = T_0$
while $T > 1$ **do**
  $Y = drawNeighbors\left(S, n, \left\lceil \frac{kT}{2T_0} \right\rceil\right)$
  $updateParetoFrontier(M, Y)$
  $\Lambda = updateWeights(S)$
  $S :=$ accept $y \in Y$ with probability $P(S, Y, T, \Lambda)$
  if m iterations were performed at T
    $T = T\rho$
  end if
end while
return M and corresponding placements

Starting with temperature T_0, the algorithm decreases the current temperature T by a factor of ρ after each m iterations until T falls below 1. Thus, following (10), a total of $\left\lceil -\frac{\log T_0}{\log \rho} \right\rceil$ temperature levels are traversed. In each iteration, alternative placements that are "close" to those in S are generated. As usual in the Monte Carlo context, these placements are referred to as the *neighbors* of S and are stored in the variable Y. It is possible to define different neighbor relations between placements. One common way to define neighborhood is to allow replacement of at most one element, i.e., two placements are considered neighbors if they share all but one controller location. However, experiments with different definitions of neighborhood suggest using another method in the presented use case. Depending on the current temperature, placements are allowed to differ in up to $\left\lceil \frac{k}{2} \right\rceil$ elements to be considered neighbors. This ensures further dispersion at the beginning of the procedure and thus results in a broader coverage of the search space. At higher temperatures, the number of replaced controller locations decreases in order to favor convergence. After generation, the proposed neighboring placements are integrated into M immediately. Then, the weight matrix Λ is recalculated according to [9]. The iteration ends with an update of S. In this step, an element of S is replaced with its corresponding neighbor from Y with probability $P(S, Y, T, \Lambda)$. While this probability equals 1 for placements that constitute an improvement over their predecessor, it decreases for placements

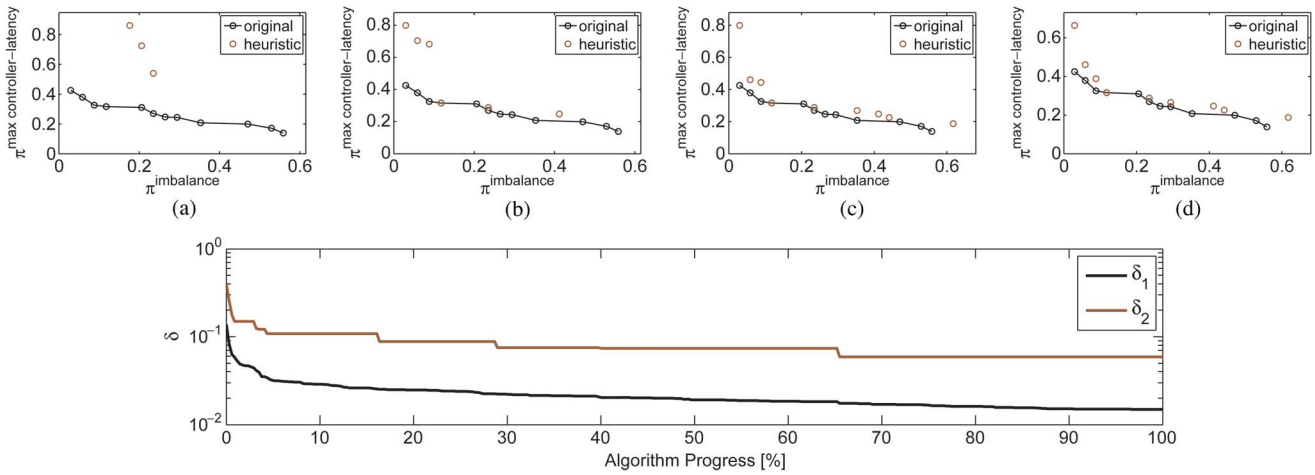


Fig. 4. Development of the Pareto frontier estimate and corresponding δ_1 and δ_2 values during a single PSA run. (a) First iteration; (b) 10%; (c) 50%; (d) 100%.

that are worse. The decrease in probability depends on the amount of deterioration as well as on the current temperature T and the weight matrix Λ. At last, the Pareto frontier M and the corresponding placements are returned.

$$\begin{aligned} T_0\rho^i &\leq 1 \\ \rho^i &\leq \frac{1}{T_0} \\ i\log\rho &\leq -\log T_0 \\ i &\leq -\frac{\log T_0}{\log\rho} \end{aligned} \tag{10}$$

Fig. 4 illustrates the described mechanisms by providing different views on a single PSA run. The particular scenario consists of finding a good placement of size $k = 6$ for the Internet2 OS3E network topology which contains $n = 34$ nodes. The input parameters for the PSA algorithm are $m = 90$, $s = 10$, $T_0 = 50$, $\rho = 0.9$, resulting in $\left\lceil -\frac{\log 50}{\log 0.9} \right\rceil = 38$ temperature levels in which up to $90 \cdot 10 = 900$ distinct placements are evaluated. Thus, PSA explores only $\frac{38 \cdot 900}{\binom{34}{6}} = 2.5\%$ of the search space. Before launching the PSA algorithm, POCO performs an exhaustive evaluation in order to provide reference data. During the run, the current Pareto frontier of the PSA routine is periodically compared to the reference by means of the distance measures δ_1 and δ_2. Additionally, the development of the approximated Pareto frontier is visualized by taking snapshots of the two-dimensional Pareto frontier with respect to a subset of two metrics. These snapshots are presented at the top of Fig. 4. Each of the four plots displays the original Pareto frontier with respect to $\pi^{\text{imbalance}}$ on the x-axis and $\pi^{\text{max controller-latency}}$ on the y-axis alongside the set of Pareto optimal points explored by the PSA algorithm. To provide a better distinction, the actual Pareto frontier is connected with line segments. Plot captions provide the relative progress of the PSA algorithm. While at the beginning, the heuristic solutions are clustered at a $\pi^{\text{imbalance}}$ value of around 0.25, the dispersion mechanism manages to explore a wide range of different solutions quickly. Thus, already at 10% of the algorithm's runtime, many different combinations of metric values are available. With increasing progress, the margin between the two frontiers narrows and finer trade-offs become visible as more solutions become available. In some instances, the exact Pareto optimal placements are identified. In contrast to the limited view on just two metrics displayed at the top, the bottom part of the figure shows how the values of δ_1 for the average distance and δ_2 for the maximum distance between the elements of the Pareto frontiers with respect to all considered metrics develop. The x-axis shows the percental algorithm progress while the logarithmically scaled y-axis denotes the δ values. Two phases can be identified in this plot. First, the segment from 0 to 5%, in which a rapid decrease of both metrics can be observed. As already mentioned, this is due to the quick exploration of the solution space when the temperature setting in the PSA routine is still high. The second phase is characterized by a slow but steady decrease of δ_1 and a stepwise decrease of δ_2. The jagged shape of the curve displaying the development of δ_2 can be explained by the fact that δ_2 considers only the maximum distance between any point of the reference set and its closest counterpart in the approximation. Thus, not every improvement in the approximation is immediately reflected by δ_2. As soon as the solution that causes the maximum value is replaced with a better alternative, δ_2 represents the next worst placement. In contrast to this behavior, every improvement is covered by the average distance δ_1, leading to a smooth curve. At the end of the run, values of δ_1 and δ_2 reach 1.5% and 5.5%, respectively.

B. Evaluation Methods

In order to analyze the performance of the implemented PSA approach, various evaluation schemes are carried out. Special focus lies on quantifying the trade-off between the time saved when using the heuristic approach and the loss in terms of accuracy that is entailed. For this purpose, numerous network topologies from the Internet Topology Zoo are first evaluated in an exhaustive fashion using the POCO framework. Results from these evaluations serve as reference for the accuracy assessment of the heuristic. By varying the input parameters of the PSA algorithm as well as by investigating its behavior in the context of different topologies and numbers of controllers, practical guidelines are derived. These allow operators to express their

specific needs in terms of accuracy requirements and time constraints.

The evaluation works as follows. Initially, combinations of network sizes and numbers of controllers are determined that can be handled by POCO's routine for exhaustive evaluation without exceeding the RAM of the used machine. The motivation for choosing these scenarios is twofold. First, the described combinations pose the highest time and memory requirements to POCO while avoiding distortive performance degradation due to issues like swapping. Second, resulting computation times that are beyond multiple minutes exceed many practical constraints, especially when aiming for dynamic controller placement. Thus, these scenarios represent use cases in which decision makers need to resort to heuristic approaches in order to comply with time and resource constraints.

Afterwards, parameters for the PSA algorithm are determined for different target specifications regarding accuracy and reliability. These specifications are represented by triples consisting of the reference metric $\delta \in \{\delta_1, \delta_2\}$ as defined in Section II-C, a threshold τ for δ, and $f_{\min}$, the desired fraction of instances in which δ is below τ. Given such a specification, parameters for the PSA procedure are calculated as follows. Starting with the lowest amount of iterations for the PSA routine, i.e., setting $m = s = 1$, PSA is applied to the problem instance 40 times. For each repetition, the difference between the Pareto frontier returned by the heuristic and the actual Pareto frontier obtained via exhaustive evaluation is computed with respect to the metric δ. If the percentage of instances in which δ does not exceed τ is beyond $f_{\min}$, the current parameters of the PSA algorithm are returned. Otherwise, the search for parameters continues in a fashion similar to binary search, i.e., increasing m and s until the constraints are met and consequently decreasing them in order to obtain the smallest viable values.

In addition to finding the minimal parameter values for a given specification, performance statistics are recorded. On the one hand, the time consumption of the exhaustive evaluation is compared with that of PSA. For this, the Matlab function `timeit`[1] is applied to both computation procedures, yielding times t^{POCO} and t^{PSA}, respectively. However, absolute times are specific to the used hardware and are thus difficult to interpret. This issue is tackled by combining both values into a ratio $t^{\mathrm{rel}} = \frac{t^{\mathrm{PSA}}}{t^{\mathrm{POCO}}}$ which denotes the speed achieved by PSA relative to the exhaustive approach. On the other hand, the fraction of the search space that is explored by the PSA algorithm is recorded in order to investigate possible relationships between this fraction and the achieved performance. Especially when deciding upon parameters for network configurations for which no reference values are available, this can provide reasonable settings for PSA. As described in the previous section, PSA goes through $\left\lceil -\frac{\log T_0}{\log \rho} \right\rceil$ temperature levels. With m iterations per temperature level and s proposed neighbors in each iteration, up to $m \cdot s \cdot \left\lceil -\frac{\log T_0}{\log \rho} \right\rceil$ distinct elements of the search space are visited. This number is denoted as b^{PSA} and is referred to as *budget*. In an analogous fashion to the runtime analysis, $b^{\mathrm{POCO}} = \binom{n}{k}$ refers to the budget requirement of POCO and $b^{\mathrm{rel}} = \frac{b^{\mathrm{PSA}}}{b^{\mathrm{POCO}}}$ describes the relative budget.

For all considered scenarios, the parameter k which indicates the desired number of controllers to be placed is assumed to be known beforehand. This assumption simplifies the problem by reducing the size of the search space from $\sum_{k=1}^{n} \binom{n}{k}$, where all possible numbers of controllers k are considered, to $\binom{n}{k}$. However, the presented evaluation scheme focuses on the relationship between the size of the search space and the resulting performance. In the future, algorithms for estimating feasible values of k could be analyzed and incorporated into the framework. The field of machine learning offers approaches like the x-means algorithm [21] for the related problem of determining the number of clusters which is a required input for the k-means clustering algorithm.

All evaluations are carried out with the same set of objectives, namely node to controller latency, inter-controller latency, and controller load imbalance. This setup results in a total of five objectives as average and maximum values are optimized for the latency measures. Due to the fact that each considered placement is evaluated with respect to each of the chosen objectives, the number of objectives as well as their individual complexity affect the total runtime of the exhaustive and the heuristic method. However, the dependency on the number of objectives is relative because both methods use the same subroutines for evaluating placements and are thus equally affected by changes in the number and complexity of objectives.

V. Results

Based on the evaluation technique introduced in Section IV-B, computations with different specifications are performed. On the one hand, the general performance of the PSA heuristic is analyzed. On the other hand, trade-offs between the invested computational effort and the resulting accuracy are investigated. Beyond that, guidelines regarding the choice of algorithms and input parameters for the PSA algorithm are derived from the analysis of real world datasets. Finally, a comparison of the absolute time consumption of the exhaustive and the heuristic approach is presented. This demonstrates the feasibility of the heuristic approach in the presence of large problem instances and dynamic environments where recalculations of controller locations need to be performed on a regular basis.

When facing an instance of the controller placement problem whose search space is too huge to analyze in an exhaustive fashion while meeting time and resource constraints, the heuristic approach presented in this work can be applied. While the input parameters of the PSA algorithm allow calculating an upper bound for the amount of analyzed placements, the absence of reference data in such cases prohibits making a statement about the resulting accuracy. In order to provide practical guidelines for the parameter choice, the relationship between performance constraints in terms of accuracy and the required relative budget b^{rel} is investigated. Therefor, an evaluation involving more than 60 graphs from the Internet Topology Zoo is performed. For each graph, four different numbers of controllers are tested and range from 5 to 15, depending on the graph's size. With

[1] http://www.mathworks.com/help/matlab/ref/timeit.html

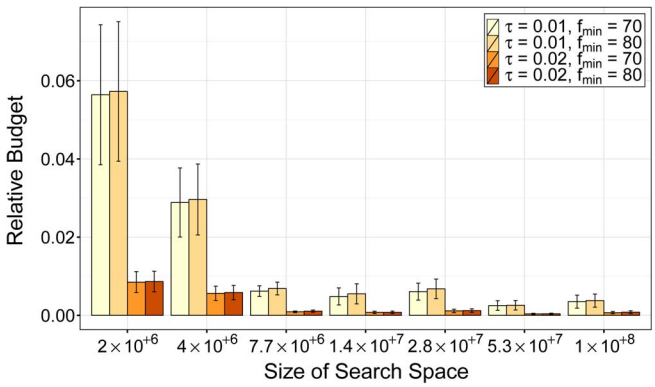


Fig. 5. Size of search space and corresponding relative budget required in order to achieve various performance levels.

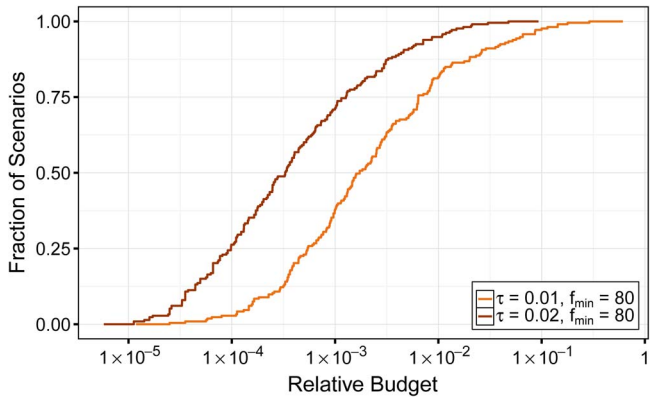


Fig. 6. Distribution of relative budget required in order to achieve various performance levels.

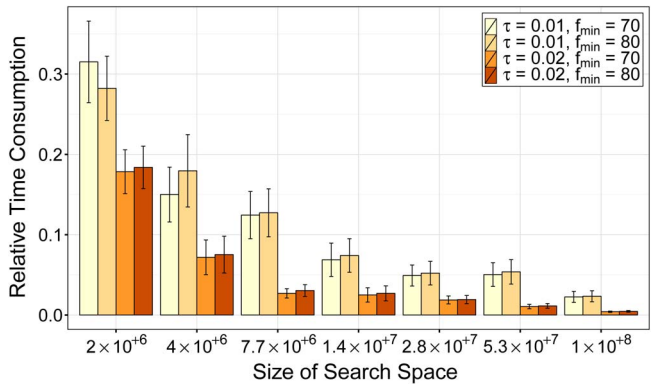


Fig. 7. Size of search space and corresponding relative time required in order to achieve various performance levels.

graph sizes ranging from 25 to 50 nodes, these scenarios feature search spaces containing between one and 100 million different placements.

Fig. 5 illustrates results from this evaluation. The data is grouped into logarithmically spaced bins according to the size of the search space in each scenario. Labels on the x-axis indicate the bins' thresholds, e.g., the first bin contains all scenarios for which $\binom{n}{k} < 2 \times 10^6$ holds. The y-axis displays b^{rel}, the relative budget required for achieving the performance goals set by the specification. While the bars' height denotes the mean value of b^{rel} in the performed evaluations, whiskers represent 95% confidence intervals and bar colors show different accuracy specifications. The investigated specifications use δ_1 as performance measure and feature combinations of accuracy thresholds $\tau \in \{0.01, 0.02\}$ and fractions $f_{\min} \in \{70, 80\}$. There are three main observations. First, in the context of increasingly large search spaces, the required relative budget b^{rel} decreases. This behavior demonstrates the efficiency of the PSA mechanism. While the size of the search space, $\binom{n}{k}$, grows extremely fast with n and k, the algorithm's search strategy finds feasible solutions early on. Second, when facing rather small search spaces containing around 4 million or less options, the required b^{rel} values in order to obtain good results express a high degree of variation as indicated by the increased width of confidence intervals in this range. For such sizes of the search space, POCO's exhaustive evaluation is usually sufficiently fast, and thus is a viable alternative to the heuristic approach. Third, for each bin, two groups of bars can be identified. The two groups correspond to the two values of τ, with b^{rel} values for $\tau = 0.01$ being higher than those for $\tau = 0.02$. This observation confirms the intuition that increased accuracy demands require a higher search budget. On average, the budget requirements for $\tau = 0.01$ are 5.6 times higher than for $\tau = 0.02$. The used values of $f_{\min}$ on the other hand do not have a significant influence on the relative budget b^{rel}. While the mean values of b^{rel} corresponding to $f_{\min} = 70$ are strictly smaller than those corresponding to $f_{\min} = 80$, their confidence intervals overlap, thus prohibiting a statistically significant statement.

In order to provide a size independent view on the evaluation data, Fig. 6 presents the empirical cumulative distribution function (CDF) of b^{rel} values for different accuracy specifications. The x-axis is logarithmically scaled and shows the relative budget required for meeting specific performance constraints. On the y-axis, the fraction of cases in which the required budget is smaller than or equal to a particular value is displayed. As in the previous figure, different specifications are represented by different line colors. In order to improve readability, the curves for specifications with $f_{\min} = 70$ are omitted as they overlap with those that display values for $f_{\min} = 80$. Again, a gap between the curves corresponding to different values of τ can be identified. Furthermore, the CDF representation allows for more generic recommendations with respect to the choice of parameters for the PSA algorithm. In particular, the graph shows that a relative budget of 1% is sufficient in over 90% of tested cases when the threshold for δ_1 equals 0.02. Increasing the accuracy demand by setting $\tau = 0.01$ raises the 90% quantile to a budget of 10%. However, even in the second case, a budget of 1% suffices in 80% of instances.

As described in Section IV-B, not only the budget of the PSA algorithm is measured but also its relative time demand in comparison to the exhaustive evaluation. Using the scenarios described at the beginning of this section, Fig. 7 presents resulting t^{rel} values for different sizes of the search space. The size thresholds of logarithmically scaled bins are displayed on the x-axis, while the y-axis shows corresponding values of $t^{\mathrm{rel}} = \frac{t^{\mathrm{PSA}}}{t^{\mathrm{POCO}}}$, the relative time consumption of the PSA algorithm compared to POCO's exhaustive evaluation. Bars' heights indicate the average value per bin and whiskers mark the respective 95% confidence intervals. The different accuracy specifications

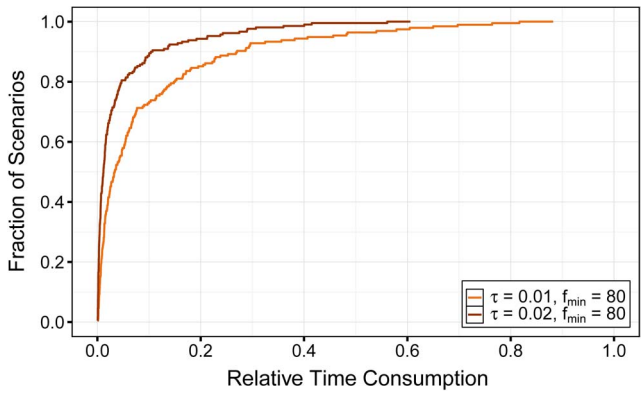


Fig. 8. Distribution of relative time required in order to achieve various performance levels.

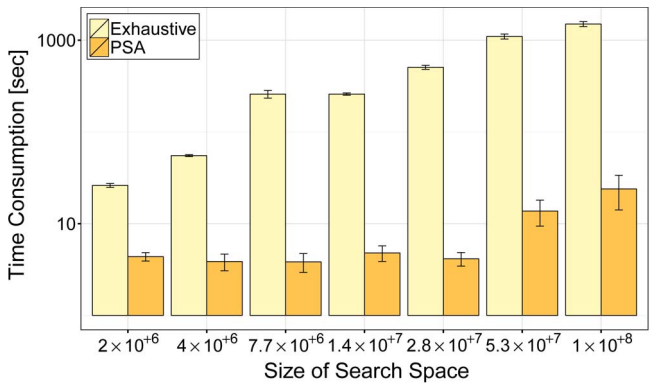


Fig. 9. Size of search space and corresponding absolute runtime requirements for an exhaustive evaluation and the proposed PSA heuristic with $\tau = 0.02$.

for δ_1 are represented by the bars' colors. With increasing size of the search space, the relative time requirement of the heuristic approach decreases steadily, dropping below 10% for sizes beyond 4 million in the case of $\tau = 0.02$ and for sizes beyond 7.7 million in the case of $\tau = 0.01$, respectively. Additionally, the confidence intervals become narrower, indicating an increase in reliability. In contrast to the budget analysis, the t^{rel} values provide a straightforward assessment of the time-accuracy trade-off offered by the PSA heuristic. For example, PSA can deliver a set of placements more than 50 times faster than the default exhaustive evaluation when the scenario's search space contains 14 million placements or more and an average deviation of 2% with regards to accuracy is tolerable.

Consolidating the different size levels and calculating probabilities of observed t^{rel} values yields Fig. 8, showing CDF curves for different specifications. The latter are denoted by different colors, while x and y-axes provide t^{rel} values and corresponding cumulative probabilities, respectively. While the t^{rel} differences between consecutive 10% quantiles are rather small until 80%, they increase significantly beyond that threshold. This can be explained by the fact that the CDF also takes into account the evaluation data obtained from instances whose search space is rather small while PSA exerts its speed advantage in the context of huge search spaces. Nonetheless, the 80% quantiles for $\tau = 0.01$ and $\tau = 0.02$ with t^{rel} values of roughly 15% and 5% indicate a potential speedup by a factor larger than 6 and 20 when incorporating PSA rather than an exhaustive evaluation.

While considering the relative time consumption of the heuristic approach provides a hardware independent comparison, absolute times allow reasoning about the practical feasibility of the mechanism in the context of a particular hardware configuration and use case. Hence, Fig. 9 presents the absolute time consumption of the exhaustive and the heuristic approach when applied to the problem instances discussed in this section. In the case of the PSA algorithm, an accuracy demand of $\tau = 0.02$ is set. All measurements were performed on an Intel Core i7 4770 CPU at 3.40 GHz and 16 GB of RAM running Windows 7 and Matlab version R2014a. Again, the x-axis provides the thresholds of logarithmically scaled bins indicating the size of the investigated search space. The y-axis displays the absolute time consumption of the two mechanisms and is logarithmically scaled as well. While differently colored bars correspond to different algorithms, bars' whiskers and heights indicate 95% confidence intervals and mean values, respectively.

In accordance with previous observations, an exhaustive evaluation can be completed nearly as fast as the heuristic approach in the context of small problem instances. On average, the exhaustive approach finished in less than a minute for problem instances with a search space size of up to four million elements. However, its runtime increases dramatically for scenarios that feature a larger search space. For the largest instances, the exhaustive evaluation takes nearly half an hour while PSA delivers a solution within less than an average of 30 seconds. In general, using PSA is one to two orders of magnitude faster than performing an exhaustive evaluation.

Additionally, the growth of the time consumption in the context of PSA is slower. While it remains relatively constant at around 5 seconds for the first five sizes, it rises to 15 and 25 seconds for the last two bins, respectively. Thus, even large instances can be handled on a scale of seconds. This behavior highlights the practical feasibility of the PSA approach for dynamic environments where frequent changes require fast reaction times in order to adapt to the new situation quickly. Even though it is not the main focus of this work, the fast computation times allow for an automated approach to the dynamic controller placement problem. This could be achieved by summarizing an operator's preferences with respect to the objectives into a single score beforehand and choosing the highest scored placement from the Pareto frontier returned by the PSA routine.

VI. Related Work

This section gives an overview on related work. First, related work on the underlying mathematical problem is provided, followed by related work on the controller placement in SDN networks particularly regarding resilience and fault-tolerance. Finally, work related to the different optimization algorithms addressed in this paper is given.

A. Facility Location Problem

As already mentioned and indicated by Heller *et al.* [5], the topic of general controller placement is well explored.

In particular, the very basic version of controller placement according to the latency of nodes to their controller is also well discussed in the context of choosing the best location for plants, warehouses, or any other facilities in a given network topology. The problem is therefore also known as *plant*, *facility*, or *warehouse location problem* and it is a typical example for a Mixed Integer Linear Program provided, e.g., with the *IBM ILOG CPLEX* [22] software. If the objective is to minimize $\pi^{\text{max latency}}$, the problem is called *k-centers problem*, if the objective is $\pi^{\text{avg latency}}$, it is called *k-median* or *k-mean problem*. Further references to this general problem are provided in Heller's work [5]. Overviews on different aspects of the facility location problem and on different methodological approaches are also given in [23] in general and in [24] with the focus on "uncertainty" regarding, e.g., uncertain traffic demands or latencies. These works however have a rather general and theoretical focus. They do not address the particular issues of controller placement in SDN networks with respect to multiple criteria and a focus on resilience. The following overview on related work focuses on variants of the controller placement problem which are closely related to the problems discussed in this paper.

A variant of the problem similar to the node to controller balancing discussed here has been introduced by Archer *et al.* [25] as *load-balanced facility problem*. The objective is similar to $\pi^{\text{imbalance}}$. However, the authors address this problem in a different context concerning particular questions arising in the area of computer graphics. Furthermore, they provide only approximations to the problem regarding their particular optimization goals. In the context of load balancing, also the term *capacitated* and *uncapacitated facility problem* can be found, see, e.g., [26] and contained references. The capacitated version assumes that the maximum number of nodes that can be assigned to a single controller is limited.

Different authors, among others Khuller *et al.* [27] and Chaudhuri *et al.* [28], look at variants called *fault tolerant* or *p-neighbor k-center problems*. These variants are similar to what is called "controller failure resilient placements" here. The works focus only on the theoretical methodology of the problem and provide approximation algorithms.

B. Controller Placement in SDN Networks

Recently, apart of Heller *et al.* mentioned before [5], more and more authors have addressed facility location in the context of controller placement in SDN networks. Bari *et al.* [4] address dynamic controller provisioning, i.e., controller placements changing over time depending on the current number of flows in the network. They propose an Integer Linear Program formulation of their "Dynamic Controller Provisioning Problem" as well as two different heuristic algorithms to solve it for larger problem instances. The authors focus their metrics on flow setup time and minimal communication overhead regarding state synchronization. Controller or network failure issues or a combination of multiple criteria such as, e.g., $\pi^{\text{imbalance}}$ or $\pi^{\text{max latency}}$ are not addressed by their work. Zhang *et al.* [15] address a resilient optimization of the controller placement problem considering the outage of nodes, links, or connections between nodes and controllers. They do not reassign nodes to new controllers if the connection to the original controller fails, but assume these nodes are controller-less and thus not able to communicate with other nodes anymore. They propose a placement heuristic and simulation with the objective of minimizing the amount of lost node to node routes due to link and node failures and controller-less nodes.

The works of Hu *et al.* [29], [30] go in a similar direction. They introduce and compare different heuristic approaches to increase the resilience of software defined networks against connection failures between nodes and controllers. Ros *et al.* [31] again consider something similar and aim at maximizing the reliability of the controller placement. They heuristically search for the minimum number of controllers assigned to each node and the controllers' placement to reach a certain reliability threshold as, e.g., "five nines". All these works [15], [29], [30] focus only on resilience against network failures and do not consider any additional metrics such as $\pi^{\text{imbalance}}$ or $\pi^{\text{max latency}}$. In particular, the trade-off between their metrics and other objectives, such as $\pi^{\text{max latency}}$, is not addressed. Furthermore, compared to the evaluation of the entire solution space, no guarantee for the optimality of the presented results can be given.

C. Multi-Criteria Optimization Algorithms

For a given combination of objectives, there are various approaches for multi criteria facility location in literature, e.g., [32]–[37] and references within. However, most of these works investigate optimization approaches for specific predefined sets of objectives rather than providing generic heuristics. Algorithms dealing with the aforementioned capacitated facility location problem, for example, consider the equivalent of the $\pi^{\text{avg latency}}$ and $\pi^{\text{imbalance}}$ metrics used in this work. Metaheuristics like the presented Pareto Simulated Annealing (PSA) [9] mechanism, on the other hand allow adding arbitrary objectives into the evaluation and are not limited with respect to the number of objectives that are taken into account during optimization. The only requirement is a function that maps elements of the search space to their performance regarding a particular objective. While techniques from the domain of evolutionary algorithms [7], [38] or genetic algorithms [39] in particular are also capable of performing multiobjective optimization, these algorithms often are at risk of getting stuck in local Pareto optima. PSA reduces this risk by accepting some worse solutions but still achieves convergence by employing a time dependent acceptance probability.

VII. Conclusion

Designing the control plane of an SDN-based network poses several challenges to decision makers. Even when the required number of elements in the externalized control plane is known, their locations influence many crucial performance aspects of the resulting system. This work addresses the controller placement problem with respect to various important metrics. These include latencies, both from nodes to controllers as well as among controllers, resilience against node and link failures,

and load balancing in the control plane. However, several of these metrics compete with each other, thus confronting the decision maker with trade-offs between them. Algorithms for exact and heuristic analyses of the resulting solution space are developed and implemented in the Matlab based POCO framework for Pareto-based Optimal COntroller placement. This article shows that the benefits of incorporating heuristic approaches are twofold. First, they allow analyzing problem instances that are too large to evaluate in an exhaustive fashion. Second, in the presence of time constraints in highly dynamic environments, heuristic algorithms allow for a trade-off between time and accuracy. The trade-off is analyzed in detail via an evaluation featuring numerous real world topologies. Depending on the user's requirements with regards to accuracy, speedups beyond a factor of 20 are possible. Furthermore, large problem instances that cannot be computed because of the enormous memory requirements can now be evaluated by applying the heuristics.

Additionally, the POCO tool, which is available as open source software,[2] provides an interactive GUI that displays a visualization of the resulting Pareto frontier with respect to up to four different metrics. Thus, decision makers can explore the solution space, perform what-if analyses, and finally accept the trade-off that suits their current preferences the most.

Finally, the scope of the presented mechanisms is not limited to the problem of SDN controller placement. By defining appropriate objective functions, i.e., mappings from placements to numerical scores, these techniques can be extended in order to tackle similar problems like the functions placement problem which appears in the context of Network Functions Virtualization.

ACKNOWLEDGMENT

The authors alone are responsible for the content of the paper.

REFERENCES

[1] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–44, Apr. 2008.

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

[3] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proc. INM/WREN*, 2010, pp. 1–6.

[4] M. F. Bari *et al.*, "Dynamic controller provisioning in software defined networks," in *Proc. Int. CNSM*, Zürich, Switzerland, 2013, pp. 18–25.

[5] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. HotSDN*, 2012, pp. 7–12.

[6] D. Hock *et al.*, "Pareto-optimal resilient controller placement in SDN-based core networks," in *Proc. 25th ITC*, 2013, pp. 1–9.

[7] J. Branke, K. Deb, K. Miettinen, and R. Slowinski, *Multiobjective Optimization: Interactive and Evolutionary Approaches*. New York, NY, USA: Springer-Verlag, 2008.

[8] D. Hock, M. Hartmann, S. Gebert, T. Zinner, and P. Tran-Gia, "POCO-PLC: Enabling dynamic pareto-optimal resilient controller placement in SDN networks," in *Proc. INFOCOM*, Toronto, ON, Canada, 2014, pp. 115–116.

[9] P. Czyżżak and A. Jaszkiewicz, "Pareto simulated annealing—A metaheuristic technique for multiple-objective combinatorial optimization," *J. Multi-Criteria Decision Anal.*, vol. 7, no. 1, pp. 34–47, Jan. 1998.

[2]POCO is available at http://www3.informatik.uni-wuerzburg.de/poco.

[10] S. Schmid and J. Suomela, "Exploiting locality in distributed SDN control," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 121–126.

[11] S. H. Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, 2012, pp. 19–24.

[12] Network Functions Virtualisation—Introductory White Paper. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[13] 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. 4th Workshop All Things Cellular*, 2014, pp. 33–38.

[14] S. Gebert *et al.*, "Demonstrating the optimal placement of virtualized cellular network functions in case of large crowd events," in *ACM SIGCOMM*, Chicago, IL, USA, 2014, pp. 359–360.

[15] Y. Zhang, N. Beheshti, and M. Tatipamula, "On resilience of split-architecture networks," in *Proc. IEEE GLOBECOM*, 2011, pp. 1–6.

[16] M. Jarschel, F. Lehrieder, Z. Magyari, and R. Pries, "A flexible OpenFlow-controller benchmark," in *Proc. EWSDN*, Darmstadt, Germany, Oct. 2012, pp. 48–53.

[17] POCO: A Framework for the Computation of Pareto-Based Optimal Controller-Placements. [Online]. Available: http://www3.informatik.uni-wuerzburg.de/poco

[18] D. Hock, S. Gebert, M. Hartmann, T. Zinner, and P. Tran-Gia, "POCO-framework for pareto-optimal resilient controller placement in SDN-based core networks," in *Proc. NOMS*, Krakow, Poland, May 2014, pp. 1–2.

[19] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

[20] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simmulated annealing," *Science*, vol. 22, no. 4598, pp. 671–680, May 1983.

[21] D. Pelleg and A. W. Moore, "X-means: Extending K-means with efficient estimation of the number of clusters," in *Proc. ICML*, 2000, pp. 727–734.

[22] ILOG, Inc., CPLEX. [Online]. Available: http://www.cplex.com/

[23] Z. Drezner, *Facility Location: A Survey of Applications and Methods*. New York, NY, USA: Springer-Verlag, 1995.

[24] S. H. Owen and M. S. Daskin, "Strategic facility location: A review," *Eur. J. Oper. Res.*, vol. 111, no. 3, pp. 423–447, Dec. 1998.

[25] A. Archer and S. Krishnan, "Importance sampling via load-balanced facility location," in *Proc. IPCO*, Bertinoro, Italy, 2008, pp. 316–330.

[26] F. J. F. Silva and D. S. de la Figuera, "A capacitated facility location problem with constrained backlogging probabilities," *Int. J. Prod. Res.*, vol. 45, no. 21, pp. 5117–5134, 2007.

[27] S. Khuller, R. Pless, and Y. Sussmann, "Fault tolerant K-center problems," *Theor. Comput. Sci.*, vol. 242, no. 1/2, pp. 237–245, Jul. 2000.

[28] S. Chaudhuri, N. Garg, and R. Ravi, "The p-neighbor k-center problem," *Inf. Process. Lett.*, vol. 65, no. 3, pp. 131–134, Feb. 1998.

[29] Y. N. Hu, W. D. Wang, X. Y. Gong, X. R. Que, and S. D. Cheng, "On the placement of controllers in software-defined networks," *J. China Univ. Posts Telecommun.*, vol. 19, pp. 92–97, Oct. 2012.

[30] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-aware controller placement for software-defined networks," in *Proc. IFIP/IEEE Int. Symp. IM*, Ghent, Belgium, 2013, pp. 672–675.

[31] F. J. Ros and P. M. Ruiz, "Five nines of southbound reliability in software-defined networks," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 31–36.

[32] U. Bhattacharya, J. R. Rao, and R. N. Tiwari, "Fuzzy multi-criteria facility location problem," *Fuzzy Sets Syst.*, vol. 51, no. 3, pp. 277–287, Nov. 1992.

[33] M. Ehrgott, *Multicriteria Optimization*. New York, NY, USA: Springer-Verlag, 2005.

[34] I. Harris, C. Mumford, and M. Naim, "The multi-objective uncapacitated facility location problem for green logistics," in *Proc. IEEE CEC*, 2009, pp. 2732–2739.

[35] A. Lancinskas and J. Zilinskas, "Solution of multi-objective competitive facility location problems using parallel NSGA-II on large scale computing systems," in *Applied Parallel and Scientific Computing*. New York, NY, USA: Springer-Verlag, 2013, pp. 422–433.

[36] T. Xifeng, Z. Ji, and X. Peng, "A multi-objective optimization model for sustainable logistics facility location," *Transp. Res. Part D, Transp. Environ.*, vol. 22, pp. 45–48, Jul. 2013.

[37] S. H. A. Rahmati, V. Hajipour, and S. T. A. Niaki, "A soft-computing pareto-based meta-heuristic algorithm for a multi-objective multi-server facility location problem," *Appl. Soft Comput.*, vol. 13, no. 4, pp. 1728–1740, Apr. 2013.

[38] A. Abraham and L. Jain, *Evolutionary Multiobjective Optimization*. New York, NY, USA: Springer-Verlag, 2005.
[39] L. Davis, *Handbook of Genetic Algorithms*. New York, NY, USA: Van Nostrand Reinhold, 1991.

Stanislav Lange studied computer science at the University of Würzburg, Germany, where he received his M.Sc. degree in 2014. Currently, he is a researcher in the "Next Generation Networks" research group at the Chair of Communication Networks in Würzburg and is pursuing his Ph.D. His research is focused on software defined networking, performance analysis, system modeling, as well as multiobjective optimization.

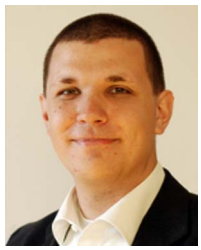
Steffen Gebert studied computer science at the University of Würzburg, Germany, where he received his diploma degree in 2011. Currently, he is a researcher in the "Next Generation Networks" research group at the Chair of Communication Networks in Würzburg and is pursuing his Ph.D. He is interested in advantages and disadvantages brought by software defined networks and virtualized network functions—be it from a performance perspective, as well as from management of orchestration.

Thomas Zinner studied computer science at the University of Würzburg, Germany. He finished his Ph.D. on performance modeling of QoE-aware multipath video transmission in the future Internet in 2012. He is heading now the "Next Generation Networks" research group at the Chair of Communication Networks in Würzburg. His main research interests cover video streaming, QoE-aware networking, SDN and NFV, as well as the performance assessment of these technologies and architetures.

Phuoc Tran-Gia is professor and director of the Chair of Communication Networks, University of Würzburg, Germany. He is also Member of the Advisory Board of Infosim (Germany) specialized in IP network management products and services. Prof. Tran-Gia is also cofounder and board member of Weblabcenter Inc. (Dallas, Texas), specialized in Crowdsourcing technologies. Previously he was at academia in Stuttgart, Siegen (Germany) as well as at industries at Alcatel (SEL) and IBM Zurich Research Laboratory. He is active in several EU framework projects and COST actions.

His research activities focus on performance analysis of the following major topics: Future Internet & Smartphone Applications; QoE Modeling & Resource Management; Software Defined Networking & Cloud Networks; Network Dynamics & Control; Crowdsourcing. He has published more than 100 research papers in major conferences and journals and received the Fred W. Ellersick Prize 2013 (IEEE Communications Society).

David Hock is a senior consultant for research and development at Infosim GmbH & Co. KG and is coordinating the research activities in the area of Software Defined Networking. Before he was working as a research assistant at the Chair of Communication Networks at the Institute of Computer Science in Würzburg where he finished his Dr. rer. nat. degree in 2014. His current main research interests are in the integration of Software Defined Networking and Network Management.



Michael Jarschel is working as a research engineer in the area of Software Defined Networking at Nokia in Munich, Germany. He finished his Ph.D. thesis, titled "An Assessment of Applications and Performance Analysis of Software Defined Networking," at the University of Würzburg in 2014. His main research interests are in the applicability of SDN and NFV concepts to next generation mobile networks.

Marco Hoffmann studied computer science and received the Dr rer. nat. degree from TUM in 2005. In 2004 he joined the Research and Development Department of Siemens. Currently he is Technology Manager for software defined networking and virtualization and Project Manager for international projects in the Research division of NOKIA. He was consortium leader and board member of several national and international research projects and member of company internal and nation-wide Future Internet strategy teams.