2015 27th International Teletraffic Congress

Specialized Heuristics for the Controller Placement Problem in Large Scale SDN Networks

Stanislav Lange, Steffen Gebert, Joachim Spoerhase, Piotr Rygielski, Thomas Zinner, Samuel Kounev, and Phuoc Tran-Gia

University of Würzburg, Institute of Computer Science, Würzburg, Germany
{stanislav.lange, steffen.gebert, joachim.spoerhase, piotr.rygielski, zinner, samuel.kounev, trangia}@uni-wuerzburg.de

***Abstract*—The Software Defined Networking (SDN) concept introduces a paradigm shift in the networking world towards an externalized control plane which is logically centralized. When designing an SDN-based WAN architecture, it is of vital importance to find a feasible solution to the controller placement problem, i.e., to decide where to position a limited amount of resources within the network. In addition to time-independent constraints regarding aspects like scalability, resilience, and control plane communication delays, dynamically changing network conditions like traffic patterns or bandwidth demands need to be considered as well. Consequently, such dynamic environments call for a regular and fast recalculation of placements in order to adapt to the current situation in a timely manner. While an exhaustive evaluation of all possible solutions can be performed within a practically feasible time frame for small and medium-sized networks, such an approach is out of scope for large problem instances which have significantly higher time and memory requirements. Therefore, this work investigates a specialized heuristic, which takes into account a particular set of optimization objectives and returns solutions representing the possible trade-offs between them. Due to its low computation time and acceptable margin of error, this heuristic can be employed by automatic decision systems operating in dynamic environments.**

***Index Terms*—SDN, Controller Placement, Latency, Multiobjective Optimization.**

I. Introduction

Within the domain of communication networks, the Software Defined Networking (SDN) paradigm has caused a shift towards an architecture whose key characteristics are the separation of control and data plane as well as a logically centralized control plane. This is achieved by moving control plane functions from individual network devices to a dedicated controller software running on commodity hardware. Communication between this centralized control plane and the data plane is then performed via the southbound API [1] which is implemented by protocols like OpenFlow [2]. Furthermore, the scalability and resilience of an SDN infrastructure can be enhanced by physically distributing the logically centralized control plane as proposed by concepts like HyperFlow [3] and ONOS [4].

A physically distributed control plane introduces additional challenges which have to be addressed. This includes the number of SDN controllers required for a targeted performance or resilience level, but also the appropriate placement of these controllers based on the relevant objectives for the given use case. These objectives cover aspects like load balancing among controller instances and communication delays between the involved control and data plane instances in heterogeneous network environments. Furthermore, operators often need to cope with dynamically changing network conditions [5] which require a periodic recalculation of viable placements in order to adapt to these changes in an appropriate manner. Therefore, the time consumption of a placement algorithm constitutes one of its key performance indicators.

The controller placement problem for the SDN domain was first introduced in [6], where an optimization regarding the latency from nodes to their assigned controller is performed. This optimization is equivalent to the *facility location problem*, a task which is known to be NP-hard. The authors perform an exhaustive evaluation of all possible placements in order to analyze the characteristics of the optimal solutions.

Our previous work [7] investigates an extended version of the controller placement problem which deals with multiple optimization objectives, e.g., resilience considerations and inter-controller latencies. In realistic environments, such performance objectives are often competing, thus there is usually no definite solution that satisfies all goals optimally. Rather, a trade-off between the competing objectives that fits the particular use case needs to be chosen. While such an exhaustive evaluation can be executed within a practically feasible time frame for small and medium-sized problem instances, such an approach is out of scope for large problems, which have significantly higher time and memory requirements.

Previous research [8] shows that large instances of the controller placement problem can be handled efficiently by employing heuristics from the domain of multiobjective combinatorial optimization. Such heuristics allow trading off between the accuracy and the solving time in order to provide fair solutions in a timely manner. For example, the Pareto simulated annealing heuristic proposed in [8] yields results within tens of seconds for problem instances whose exhaustive evaluation takes tens of minutes while producing an average error below 2%. However, such generic heuristics do not take advantage of the use case specific set of optimization objectives.

This work explores the potential of heuristics, which are designed to leverage a use case specific subset of objectives. In particular, we investigate the applicability of a specialized k-Medoids [9] algorithm for the controller placement problem,

978-1-4673-8422-3/15 $31.00 © 2015 IEEE
DOI 10.1109/ITC.2015.32

and discuss its impact on the optimization accuracy and the required computational effort.

The remainder of this paper is structured as follows. Section II provides the formal problem statement, the specialized heuristic algorithm proposed in this work, and an overview of the evaluation setup. Evaluation results are presented in Section III. After that, Section IV covers related work and Section V concludes the work.

II. Methodology

In this section, we formally define the graph-based controller placement problem and introduce the necessary notation. Then, we describe in detail the proposed Pareto capacitated k-Medoids (PCKM) heuristic algorithm. Finally, we outline the methodology that is used in the evaluation.

A. Problem Statement

In this work, the network under study is represented by a graph $G = (V, E)$ with node set V and edge set E. This network consists of a total of n nodes that correspond to network elements, i.e., switches and controllers, that are connected via links. Furthermore, shortest path latencies between each pair of nodes are stored in a distance matrix D. In particular, entry $d_{i,j}$ contains the latency between nodes i and j. The entries of D are normalized with the graph's diameter.

Given k, the number of controllers to be placed in the network, the goal of the optimization task lies in determining locations for k controllers $\mathcal{P}_k = \left\{\mathcal{P} \in 2^V \mid |\mathcal{P}| = k\right\}$ so that objective functions $f_i\,(i \in \{1, \ldots, J\})$ are minimized. These functions map placements to numeric values corresponding to their performance with respect to the individual objectives like latency or imbalance. As discussed in Section I, there are usually multiple competing objectives for which practically feasible trade-offs need to be found. These trade-offs can be captured by analyzing the set of Pareto optimal solutions. Formally, 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.

In order to quantify the loss of accuracy caused by using the heuristic approach, we adopt a measure from [10] that quantifies the distance between the actual and the estimated Pareto frontier. In the following, R denotes the original Pareto frontier that is used as reference, and M represents the estimate provided by the heuristic approach. Before we define the distance between two Pareto frontiers, a distance metric for placements is introduced. According to Equation 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 the inverse of objective f_j's range and is used for normalization, i.e., $w_j = \left(\max_{x \in R} f_j(x) - \min_{x \in R} f_j(x)\right)^{-1}$ and thus, $c(x, y) \in [0, 1]$. 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 metric δ is shown in Equation 2 and measures the average distance between each element in R and its closest element from M.

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

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

While there are many important criteria that need to be considered when planning an SDN architecture, this work focuses on two particular criteria for which an optimization algorithm is developed. On the one hand, minimizing the average latency between the nodes and their controller reduces the delays appearing in the southbound communication. Given a placement $\mathcal{P} \in 2^V$ and a distance matrix D, the average node to controller latency $\pi^{\text{avg latency}}$ is defined according to Equation 3. This implicitly assumes a node to controller assignment based on shortest path latencies. However, it is also possible to explicitly define an assignment $A\colon V \to V$ that maps each node to a specific controller. In such a case, the calculation of the average latency follows Equation 4. On the other hand, minimizing the load imbalance among controllers produces a more robust network and also assures that each controller behaves in a similar fashion. For each placement $\mathcal{P}$ and controller p, the total number of nodes that are assigned to p 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. Equation 5 provides a formal definition of this imbalance metric.

$$\pi^{\text{avg latency}}(\mathcal{P}) = \frac{1}{|V|} \sum_{v \in V} \left(\min_{p \in \mathcal{P}} d_{v,p} \right) \tag{3}$$

$$\pi^{\text{avg latency}}(\mathcal{P}, A) = \frac{1}{|V|} \sum_{v \in V} d_{v,A(v)} \tag{4}$$

$$\pi^{\text{imbalance}}(\mathcal{P}) = \max_{p \in \mathcal{P}} n_p - \min_{p \in \mathcal{P}} n_p \tag{5}$$

Although the developed optimization algorithm does not optimize any other criteria, an evaluation of its performance with respect to additional objectives illustrates the differences between this specialized approach and more generic heuristics. There are three further objectives under study. Similarly to $\pi^{\text{avg latency}}$, the $\pi^{\text{max latency}}$ measure considers the maximum node to controller latency and thus offers a worst case analysis. Furthermore, another kind of latency is relevant in the context of a distributed control plane, i.e., the latency among controllers who need to communicate in order to maintain a synchronized state. Again, average and maximum values are calculated for this objective.

B. Pareto Capacitated k-Medoids

The heuristic algorithm proposed in this work combines ideas from several graph theoretical algorithms in order to construct an approximation of the Pareto frontier with respect to two objective functions. These functions include the average node to controller latency and the imbalance regarding controller load. When considering only the node to controller latency, a clustering based approach is sufficient as it provides the location of controllers as well as an assignment from nodes to controllers which minimizes the latency between them. However, such algorithms do not take into account the amount of nodes assigned to each controller and thus might return arbitrarily bad results with respect to $\pi^{\text{imbalance}}$. Therefore, the k-Medoids clustering algorithm [9] is enhanced with a capacity bound ρ which restricts the number of nodes that can be assigned to a single controller. By iteratively increasing the bound ρ, the maximum resulting imbalance can be influenced and thus, different trade-offs between the two objectives can be explored and summarized in a Pareto frontier. While the parameter ρ primarily affects the minuend of Equation 5, i.e., $\max_{p \in \mathcal{P}} n_p$, it also has an effect on the resulting imbalance metric. This is caused by the fact that the capacity bound implicitly limits the range of values $\pi^{\text{imbalance}}$ can attain. In contrast to other clustering algorithms such as k-Means [11], the centers returned by the k-Medoids algorithm coincide with the input graph's nodes. Therefore, it is chosen as the foundation for the proposed heuristic.

Algorithm 1 illustrates the capacitated k-Medoids approach. In addition to the distance matrix D, the network size n, and the number of controllers k, the algorithm receives the capacity bound ρ. First, the unmodified k-Medoids algorithm is applied to the problem instance (line 2). However, instead of assigning each node to its closest controller, a latency minimal balanced assignment of nodes to controllers is determined. Finding such an assignment corresponds to finding a cost minimal perfect matching in a bipartite graph which is constructed in lines 3 and 4 of the algorithm. The first partition, N, consists of n vertices representing the network's nodes, while controllers are placed in the second partition, which is referred to as F. In order to enforce the desired capacity limit of controllers, each controller is replicated $\left\lceil \frac{n}{k} \right\rceil + \rho$ times. Thus, a value of $\rho = 0$ corresponds to the tightest bound, i.e., the number of instances per controller prohibits configurations in which one controller manages significantly more nodes than another controller.

Next, a complete bipartite graph is created by adding an edge between each pair of nodes from the two partitions. Edge weights in this graph correspond to the entries in the distance matrix D. Consequently, a cost minimal perfect matching in the resulting bipartite graph yields an assignment of nodes to the controllers provided by the k-Medoids algorithm. As this assignment does not necessarily match the one intended by the k-Medoids algorithm, a shift of centers inside the partitions defined by the assignment might improve the latency in each partition without affecting imbalance. Hence, in each cluster, each node is considered as the new center. If this relocation improves the sum of latencies inside the cluster, it is accepted. Afterwards, the last two steps (i.e., calculating assignments given centers and calculating centers given clusters) are repeated until convergence with respect to the latency is reached. This process corresponds to the while loop in lines 7 to 12 of Algorithm 1.

The final output consists of C, the cluster centers and A, the assignment of each node to its center. Due to the fact that the exhaustive evaluation assumes an assignment that is based on the minimization of shortest path latencies, the capacitated k-Medoids algorithm can produce placements that are not analyzed by the exhaustive approach. However, this does not have a negative impact on the results of the performance evaluation as the distance measure defined in Equation 1 defaults to zero when the estimate performs better than the reference solution.

Algorithm 1 Capacitated k-Medoids

1: **input**: D, n, k, ρ
2: $C = \text{kMedoids}(D, n, k)$ $(= \{c_1, \ldots, c_k\})$
3: $N = \{1, \ldots, n\}$
4: $F = \left\{ c_1^1, c_1^2, \ldots, c_1^{\left\lceil \frac{n}{k} \right\rceil + \rho}, c_2^1, \ldots, c_k^{\left\lceil \frac{n}{k} \right\rceil + \rho} \right\}$
5: $(A, costs) = \text{match}(N, F, D)$
6: $(C', costs') = \text{recalculateCenters}(A)$
7: **while** $costs' < costs$ **do**
8: $\quad C = C'$
9: $\quad F = \left\{ c_1^1, \ldots, c_k^{\left\lceil \frac{n}{k} \right\rceil + \rho} \right\}$
10: $\quad (A, costs) = \text{match}(N, F, D)$
11: $\quad (C', costs') = \text{recalculateCenters}(A)$
12: **end while**
13: **return** (C, A)

For a single value of ρ, Algorithm 1 calculates a placement which minimizes the average node to controller latency while respecting the imbalance constraint introduced by ρ. However, the goal of this work is to develop an algorithm capable of providing insights into the available alternatives and their associated trade-offs with respect to different objectives. Therefore, Algorithm 2 combines the results of multiple runs of the capacitated k-Medoids algorithm with different values of ρ into a Pareto frontier of possible solutions. Additionally, parts of the k-Medoids heuristic rely on random numbers. Thus, the quality of results can be further improved by running the algorithm multiple times for each configuration. In total, there are two parameters that control the runtime and performance of the Pareto capacitated k-Medoids (PCKM) algorithm. First, P, the set of values for the capacity bound ρ, and second, n_r, the number of times the capacitated k-Medoids routine is called for each $\rho \in \text{P}$. The placements that are obtained during these $|\text{P}| \cdot n_r$ iterations are stored in the set S which is constructed in line 5 of the algorithm. Finally, the Pareto optimal placements are determined by evaluating the elements in S with respect to the two metrics under consideration and deriving their Pareto frontier.

Algorithm 2 Pareto Capacitated k-Medoids

input: D, n, k, P, n_r
$S = \emptyset$
for all $i \in \{1, \ldots, n_r\}$ **do**
  for all $\rho \in \mathrm{P}$ **do**
    $S = S \cup \mathrm{capacitatedKMedoids}(D, n, k, \rho)$
  end for
end for
$S = \{s \in S \mid s \in \mathrm{paretoFrontier}(\mathrm{evaluate}(S))\}$
return S

C. Evaluation Methods

The performance evaluation of the developed Pareto capacitated k-Medoids (PCKM) algorithm focuses on two main aspects. First, an analysis of the algorithm's runtime for different sets of parameters and the distance between the resulting and the actual Pareto frontier provides the data required for quantifying the achieved trade-off between time and accuracy. Second, the specialized heuristic proposed in this work is compared to the generic Pareto simulated annealing discussed in Section I as well as to a baseline algorithm that is based on randomly guessing placements. This comparison also explores the consequences of specialization by calculating Pareto frontier distances with respect to the subset of objectives optimized by the Pareto capacitated k-Medoids approach as well as with respect to a larger set of objectives. Both parts of the evaluation are carried out on a set of real world network topologies from the Internet Topology Zoo [12]. In order to provide reference values for the time and accuracy assessment, an exhaustive evaluation of placements is performed for each of the selected networks and desired number of controllers beforehand.

Investigated problem instances are chosen based on two factors. First, an exhaustive evaluation of the search space corresponding to the problem instance should not exceed the available memory of the used machine. We motivate this by the fact that problem sizes beyond this threshold cause phenomena like page thrashing which in turn make the comparison unfair. Second, the computation times for an exhaustive evaluation of the chosen instances tend to be in the order of magnitude of several to tens of minutes. Such runtimes exceed the constraints that often appear in practice. Hence, these scenarios correspond to use cases which can be made tractable by employing heuristics. Overall, more than 60 graphs from the Internet Topology Zoo are evaluated. Their sizes range from 25 to 50 nodes and the experiments cover numbers of controllers between 5 and 15, resulting in state spaces that contain between one and 100 million distinct possible placements. All algorithms are implemented in Matlab and run on a server equipped with an Intel Xeon CPU at 2.10 GHz and 128 GB of memory running the 64-bit version of Ubuntu 13.10 and Matlab version R2014a.

In addition to the absolute runtimes of all investigated algorithms and parameter sets which are recorded via Matlab's `timeit`[1] function, the relative time consumption of the heuristics are computed. While absolute runtimes provide insights into the time scales which can be achieved by using heuristics, statements about the relative time consumption allow for a hardware independent comparison of algorithms. The relative runtime of a heuristic approach is defined as the ratio of the heuristic's computation time and the time required for an exhaustive evaluation.

As discussed in Section II-B, the performance of the Pareto capacitated k-Medoids algorithm with respect to runtime and accuracy can be controlled with two input parameters. On the one hand, n_r, the number of algorithm runs per configuration, controls the number of placements investigated and can be used to gather more reliable results. In this paper, we use $n_r = \{2, 4, \ldots, 10\}$. Increasing n_r beyond 10 did not show significant improvements regarding the algorithm's accuracy. It is worth noting that n_r is not the number of experiment repetitions used in the performance evaluation, but rather an input parameter to the PCKM algorithm that controls its runtime. On the other hand, the parameter P controls the range of capacity bounds that are evaluated in order to construct the two dimensional Pareto frontier. Two options regarding the choice of P are analyzed. First, $\mathrm{P} = \{0, 1, \ldots, 9\}$, which covers 10 consecutive values for ρ and aims at thoroughly analyzing the trade-off between latency and imbalance. Second, $\mathrm{P} = \{0, 2, \ldots, 18\}$, which also contains 10 distinct values for ρ but targets covering a wider range of trade-offs.

In order to provide a context for PCKM's performance, it is compared with two additional algorithms. These include the Pareto simulated annealing (PSA) heuristic [8], [10] as well as an algorithm that evaluates a given number of randomly generated placements and calculates the Pareto frontier of the resulting set of solutions. Due to the fact that the performance and the runtime of the different mechanisms depends on their input parameters, the following methodology is used in order to achieve a fair comparison. For a given set of input parameters of the PCKM approach, the average runtime is determined. Afterwards, the input parameters of the alternative algorithms are tweaked in such a fashion that their runtime equals that of PCKM. Hence, the comparison allows statements about the different algorithms' performance when equipped with a particular time budget. In order to obtain statistically significant results, 10 repetitions are performed and evaluated for each combination of algorithm, network graph, and set of input parameters.

III. Results

This section presents results of the performance evaluation setup outlined in Section II-C. On the one hand, the influence of the parameter choice on the performance of the Pareto capacitated k-Medoids algorithm is investigated. On the other hand, a comparison of the Pareto capacitated k-Medoids (PCKM) mechanism with the Pareto simulated

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

annealing heuristic as well as a baseline approach based on random guessing is presented.

Figure 1 illustrates the results obtained when using the proposed evaluation scheme. It shows the cumulative distribution of the Pareto frontier distance δ with respect to the two objectives optimized by the algorithm, i.e., the average node to controller latency and the controller imbalance. The x-axis shows increasing values of δ and the y-axis represents the fraction of scenarios in which the PCKM approach achieves a distance of at most δ. While the capacity range P is represented by the line style, with the fine grained capacity range $\{0, 1, \ldots, 9\}$ as solid lines and the coarse grained capacity range $\{0, 2, \ldots, 18\}$ as dashed lines, the number of repetitions n_r is represented by the lines' color and takes on values 2, 6, and 10.

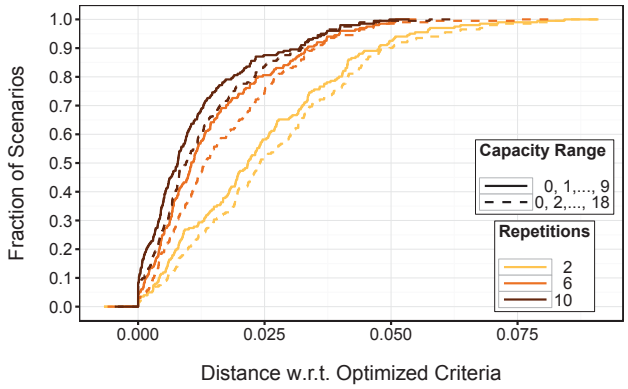


Fig. 1: CDF of the algorithm's error with respect to average latency and imbalance for different numbers of repetitions n_r and capacity ranges P

There are three main observations. First, an increase in n_r leads to an increase in accuracy. Increasing n_r not only affects the total number of placements analyzed by the algorithm, but also adds diversity to the solution as the k-Medoids subroutine starts its optimization from a different set of centers in each iteration. Second, the accuracy gains between consecutive n_r values decrease for higher n_r. For example, the 90% quantiles of the distance in case of $\mathrm{P} = \{0, 1, \ldots, 9\}$ take on values of roughly 5%, 3.4%, and 3.2% for $n_r = 2, 6, 10$, respectively. This phenomenon hints at a converging behavior, i.e., the accuracy doesn't improve significantly beyond a particular value of n_r. Third, the fine grained capacity range yields a higher accuracy than its coarse grained counterpart for all the scenarios covered in this work. This behavior stems from the fact that the reference Pareto frontier usually contains many distinct imbalance values in the lower range while the coverage of higher values is rather sparse. Hence, $\mathrm{P} = \{0, 1, \ldots, 9\}$ is used for the remainder of this work.

In order to determine input parameters for the alternative algorithms that are used in the performance comparison, the absolute runtimes of the PCKM algorithm are measured for the different configurations. Figure 2 presents the distributions of these runtimes. Differently colored curves represent different values of the number of repetitions n_r. When the number of repetitions n_r is increased from 2 to 10 in steps of 2, the median runtime increases from roughly 0.5 seconds to 2.7 seconds in almost equidistant steps. This behavior is in line with the fact that each repetition is performed independently of the other, and thus n_r affects the total runtime in a linear fashion. However, the interquantile range also increases with n_r in the analyzed scenarios. This can be explained by the varying runtimes of consecutive repetitions of the k-Medoids algorithm. For each repetition, the number of iterations spent inside the k-Medoids routine can differ. Increasing n_r implies a wider interval of possible values for the sum of these iterations and thus a higher variance is observed.

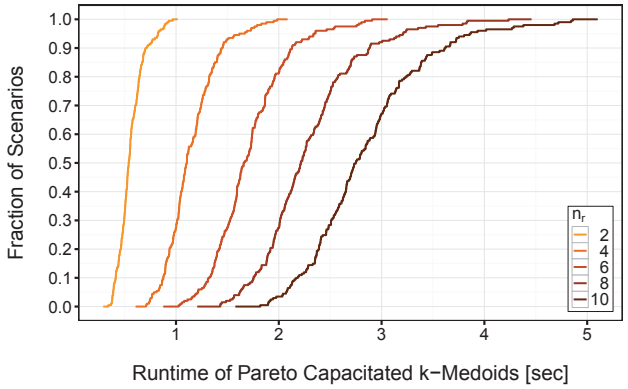


Fig. 2: CDF of the absolute time consumption of the Pareto Capacitated k-Medoids algorithm for different numbers of repetitions n_r

The contributed algorithm is compared with three reference algorithms by means of Pareto frontier distances. The results are depicted in Figures 3 and 4. In addition to PCKM, a two dimensional version of the Pareto simulated annealing algorithm (PSA2D) and an algorithm based on random guessing (RND) are analyzed.

Before aggregated results are presented in Figure 4, Figure 3 illustrates the different algorithms' behavior by displaying the Pareto frontiers returned during a single run of each algorithm and comparing them to the reference Pareto frontier obtained with the brute force approach. For this example, the Sinet topology is chosen. In this network of 47 nodes, 5 controllers are to be placed within a time budget of one second. The Pareto frontiers are determined with respect to the two objectives that are being optimized, i.e., the average node to controller latency and the controller load imbalance. X- and y-positions of individual points show the values of the objective functions achieved by the corresponding placements. Additionally, the imbalance metric is normalized with the number of nodes in the topology. As a visual aid, each set of Pareto optimal points is connected with line segments which are not part of the Pareto frontier. Different algorithms are represented with different colors and marker shapes.

There are four main observations. First, the Pareto frontier returned by the PCKM algorithm has the highest cardinality of all discussed approaches. This corresponds to a thorough coverage of the different possible trade-offs between the optimized objectives and is achieved by PCKM's iterative approach with respect to the capacity limits. By imposing different imbalance constraints, the resulting latency is varied and the search space is explored in a systematic fashion. Second, PCKM discovers a solution that is not captured by the reference Pareto frontier. In contrast to the brute force approach, PCKM is not restricted to assigning nodes to controllers based on latency and thus explores a larger search space than the other algorithms. Third, PSA2D is also characterized by the diversity of solutions, i.e., the trade-offs between objectives are reflected in the resulting Pareto set. However, the available time budget is not sufficient to achieve convergence, so that some regions contain only few solutions or feature outliers. These phenomena can be observed in the sparse coverage of the 0.2 to 0.3 range of the imbalance metric, where PSA2D yields only two solutions, as well as the rightmost outlier with respect to the latency objective. Finally, the placements found by the RND approach are scattered throughout the objective space. While the mechanism finds one Pareto optimal solution by chance, its result set also features an extreme outlier. This demonstrates the high variance and thus low reliability of the RND algorithm.

While the preceding discussion is focused on one individual run, Figure 4 presents the algorithms' performance in an aggregated fashion. Furthermore, not only the Pareto frontier distance regarding the two objectives optimized by PCKM and PSA2D is presented, but also the distance from a five dimensional Pareto frontier is taken into account. The extended set of criteria contains the maximum node to controller latency as well as the average and maximum latency among controllers. Such an analysis provides insights into the strategies that are employed by the different algorithms in order to explore the search space and find feasible solutions.

While the different algorithms are represented by differently colored curves, the line style indicates the kind of distance measure under investigation. All three subfigures of Figure 4 display cumulative distributions of these distances, each resulting from different algorithm parameters obtained according to Section II-C.

A comparison between the algorithms' performance with respect to the Pareto frontier distance regarding the average node to controller latency and imbalance shows that the PCKM algorithm consistently outperforms the PSA2D heuristic for all three time constraints that were used in Figures 4a, 4b, and 4c. This demonstrates the gain achieved by utilizing a specialized heuristic over a generic method given the same time budget on identical hardware. Moreover, the distance in case of the RND algorithm is significantly higher than those of PCKM and PSA2D as the RND approach does not systematically explore the solution space but rather evaluates random placements. Although the small absolute differences between achieved Pareto frontier distances for PCKM and

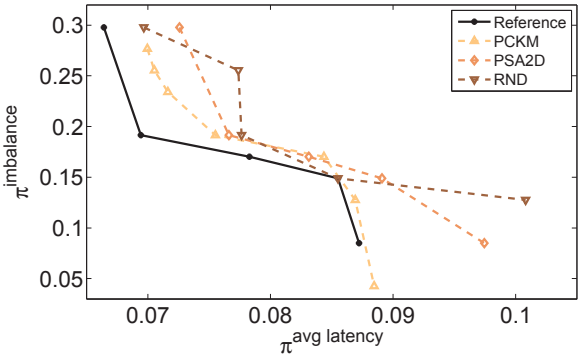


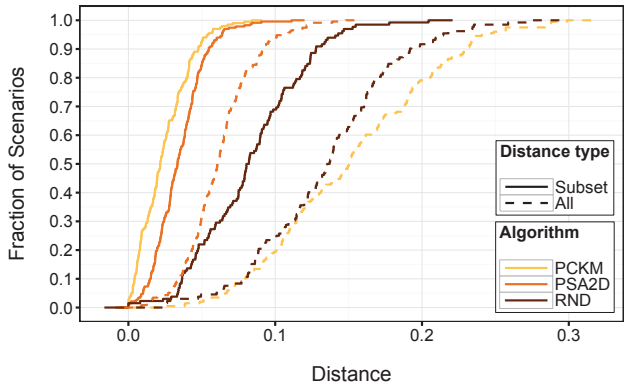
Fig. 3: Exemplary Pareto frontiers obtained with the algorithms discussed in this work. Settings: Sinet topology (47 nodes), 5 controllers, and a time budget of 1 second.

PSA2D might suggest that utilizing PCKM over PSA2D provides only a slight improvement, the relative increase is significant. For example, when inspecting the 90% quantiles of the distributions in Figure 4b, PCKM achieves a distance of 3% with respect to the subset of optimized criteria while PSA2D produces an error of 5%. Hence, choosing PCKM corresponds to a relative gain of 40% in terms of accuracy.

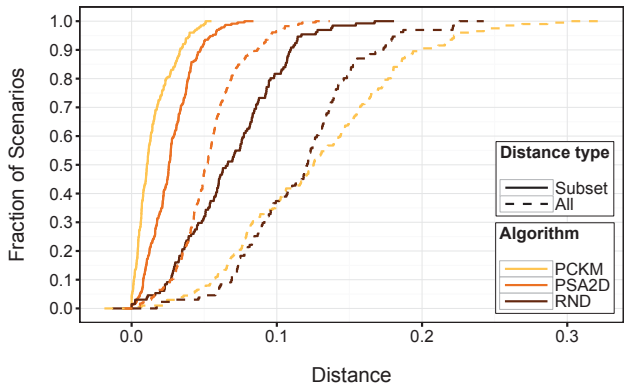
When extending the distance measure to take into account additional objectives representing the maximum and average inter-controller latency and maximum node to controller latency, two phenomena are observed. First, the absolute distance values increase for all algorithms. Such a behavior is expected as none of the algorithms explicitly optimizes for the additional objectives and the available time budget is not increased to accommodate for the increased complexity, either. Furthermore, the relative order of algorithms with respect to the achieved distance changes. In the context of all three time settings, the PSA2D mechanism provides the highest accuracy, i.e., the lowest distance values. The reasons for PSA2D's advantage in this domain are twofold. On the one hand, it generally explores a larger number of placements than PCKM which results in a higher chance of coming across solutions which are viable with respect to the newly added optimization goals. On the other hand, the PSA2D approach follows a more systematic path through the solution space than RND, which lowers the chance of visiting the same placement multiple times. Due to having the lowest amount of evaluated placements, PCKM falls short of RND when the extended distance measure is of interest.

IV. Related Work

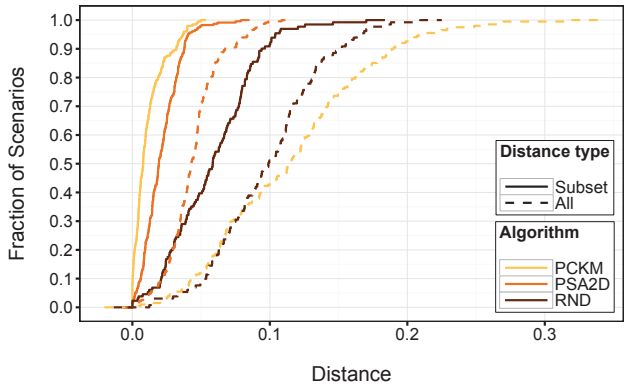
This section gives an overview on related work. First, related work on the underlying mathematical problem is provided, followed by related work on the controller placement in SDN networks. Finally, work related to the different optimization algorithms discussed in this paper is given.



(a) One second, corresponding to $n_r = 2$



(b) Two seconds, corresponding to $n_r = 6$



(c) Four seconds, corresponding to $n_r = 10$

Fig. 4: Comparison of algorithms' performance with respect to different distance types given different time constraints

A. Facility Location Problem

As already mentioned and indicated by Heller et al. [6], 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* [13] 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 [6]. Overviews on different aspects of the facility location problem and on different methodological approaches are also given in [14] in general and in [15] 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. [16] 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., [17] 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. [18] and Chaudhuri et al. [19], look at variants called *fault tolerant* or *p-neighbor k-center problems*. 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 [6], more and more authors have addressed facility location in the context of controller placement in SDN networks. Bari et al. [5] 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. [20] 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. [21], [22] 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. [23] 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 [20]–[22] 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. Use Cases for the Facility Location Problem in the Context of SDN and NFV

The incentive for runtime optimization of controller placement in SDNs and migration of functions in Network Functions Virtualization (NFV) comes from the varying usage patterns of modern data centers. The network workloads and communication patterns dynamically change due to constant virtual machine placement and scaling of the resources [24]. In this section, we review the most common SND- and NFV-related use cases for the facility location problem.

As the authors of [25], [26] observe, proper placement of network functions increases the network performance by minimizing optical-electrical-optical conversions in a "network function chaining" scenario. In that scenario, it is assumed that intra-data center networks use optical technologies in the core layer and the incoming traffic flows are steered through a number of virtual network functions in a predefined order. It is beneficial to locate the chained network functions within a single electrical domain to minimize the amount of conversions. This use case fits well in the "network stretching" use case [27] where location-agnostic networks are considered. In this use case, network workloads span multiple data centers connected by optical medium and enterprise services can be freely migrated from one data center to another. The location agnosticism implies that also the virtualized functions need to migrate as the environment adapts. Additional incentives for the necessity of placement optimization of SDN controller or network function are presented in [6], [28].

D. Multi-criteria Optimization Algorithms

For a given combination of objectives, there are various approaches for multi criteria facility location in literature, e.g., [29]–[34] 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 Pareto Simulated Annealing (PSA) [10] 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. Furthermore, techniques from the domain of evolutionary algorithms [35], [36] or genetic algorithms [37] in particular are also capable of performing multiobjective optimization.

V. Conclusion

Designing the control plane of an SDN-based architecture poses several challenges to network operators. Even when the required number of entities in the control plane is known beforehand, their locations have a significant impact on numerous performance aspects of the network. This results in the multiobjective optimization task that is known as the controller placement problem. Its solution contains sets of controller locations that represent possible trade-offs between different objectives like control plane communication delays or the balanced load distribution among controller instances. While a brute-force approach to this placement problem is practically feasible for small and medium-sized problem instances, the time and resource demands of large problem instances call for alternative mechanisms. Such mechanisms usually involve heuristics that sacrifice accuracy or optimality guarantees for significantly faster runtimes.

This article works towards quantifying the trade-off that results from employing various heuristics as well as providing guidelines with respect to algorithm choice for different use cases. In particular, a specialized heuristic that optimizes a particular set of objectives is compared to a generic heuristic capable of optimizing arbitrary sets of criteria. The results of an evaluation featuring over 60 real world network topologies demonstrate that the effort for developing a specialized heuristic pays off as its optimization accuracy has a lead over the generic approach. In addition to the gains in terms of accuracy, specialized algorithms can provide guarantees with respect to certain objectives' values, a property that might be important in a practical context.

Acknowledgments

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 authors thank Fabian Helmschrott for his programming efforts and David Hock for fruitful discussions and helpful comments.

REFERENCES

[1] M. Jarschel, T. Zinner, T. Hoßfeld, P. Tran-Gia, and W. Kellerer, "Interfaces, Attributes, and Use Cases: A Compass for SDN," *IEEE Communications Magazine*, 2014.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM CCR*, 2008.

[3] A. Tootoonchian and Y. Ganjali, "HyperFlow: a Distributed Control Plane for OpenFlow," in *INM/WREN'10*, Berkeley, CA, USA, 2010.

[4] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "ONOS: towards an open, distributed SDN OS," in *Proceedings of the third workshop on Hot topics in software defined networking*, 2014.

[5] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, "Dynamic Controller Provisioning in Software Defined Networks," in *International Conference on Network and Services Management (CNSM)*, Zürich, Switzerland, 2013.

[6] B. Heller, R. Sherwood, and N. McKeown, "The Controller Placement Problem," in *HotSDN '12*, New York, NY, USA, 2012.

[7] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks," in *25th International Teletraffic Congress (ITC)*, 2013.

[8] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks," *IEEE Transactions on Network and Service Management*, 2015.

[9] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.

[10] P. Czyżak and A. Jaszkiewicz, "Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization," *Journal of Multi-Criteria Decision Analysis*, 1998.

[11] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, 1982.

[12] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE JSAC*, vol. 29, no. 9, 2011.

[13] *CPLEX*, ILOG, Inc., http://www.cplex.com/.

[14] Z. Drezner, *Facility Location: A Survey of Applications and Methods*. Springer Verlag, 1995.

[15] S. H. Owen and M. S. Daskin, "Strategic Facility Location: A Review," *European Journal of Operational Research*, vol. 111, no. 3, pp. 423 – 447, 1998.

[16] A. Archer and S. Krishnan, "Importance Sampling via Load-Balanced Facility Location," in *IPCO'08*, Bertinoro, Italy, 2008.

[17] F. J. F. Silva and D. S. de la Figuera, "A Capacitated Facility Location Problem with Constrained Backlogging Probabilities," *IJPR*, vol. 45, no. 21, 2007.

[18] S. Khuller, R. Pless, and Y. Sussmann, "Fault Tolerant K-center Problems," *Theoretical Computer Science*, vol. 1203, 1997.

[19] S. Chaudhuri, N. Garg, and R. Ravi, "The p-Neighbor k-Center Problem," *IPL*, vol. 65, no. 3, 1998.

[20] Y. Zhang, N. Beheshti, and M. Tatipamula, "On Resilience of Split-Architecture Networks," in *GLOBECOM 2011*, 2011.

[21] Y. nan Hu, W. dong Wang, X. yang Gong, X. rong Que, and S. duan Cheng, "On the Placement of Controllers in Software-Defined Networks," *JCUPT*, vol. 19 Supplement 2, 2012.

[22] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-aware Controller Placement for Software-Defined Networks," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ghent, Belgium, 2013.

[23] F. J. Ros and P. M. Ruiz, "Five Nines of Southbound Reliability in Software-Defined Networks," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, 2014.

[24] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management*, pp. 1–53, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10922-014-9307-7

[25] M. Xia, M. shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for nfv chaining in packet/optical data centers," in *Optical Communication (ECOC), 2014 European Conference on*, Sept 2014, pp. 1–3.

[26] R. Guerzoni, R. Trivisonno, I. Vaishnavi, Z. Despotovic, A. Hecker, S. Beker, and D. Soldani, "A novel approach to virtual networks embedding for sdn management and orchestration," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, May 2014, pp. 1–7.

[27] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The dynamic placement of virtual network functions," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, May 2014, pp. 1–9.

[28] 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 *Proceedings of the 4th workshop on All things cellular*, 2014.

[29] 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.

[30] M. Ehrgott, *Multicriteria Optimization*. Springer, 2005.

[31] I. Harris, C. Mumford, and M. Naim, "The Multi-Objective Uncapacitated Facility Location Problem for Green Logistics," in *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, 2009, pp. 2732–2739.

[32] 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*. Springer, 2013, pp. 422–433.

[33] T. Xifeng, Z. Ji, and X. Peng, "A Multi-Objective Optimization Model for Sustainable Logistics Facility Location," *Transportation Research Part D: Transport and Environment*, vol. 22, pp. 45–48, 2013.

[34] 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," *Applied Soft Computing*, 2013.

[35] J. Branke, K. Deb, K. Miettinen, and R. Slowinski, *Multiobjective optimization: Interactive and evolutionary approaches*. Springer, 2008.

[36] A. Abraham and L. Jain, *Evolutionary multiobjective optimization*. Springer, 2005.

[37] L. Davis *et al.*, *Handbook of genetic algorithms*. Van Nostrand Reinhold New York, 1991.