Quality and Cost of Deterministic Network Calculus – Design and Evaluation of an Accurate and Fast Analysis

STEFFEN BONDORF, PAUL NIKOLAUS, and JENS B. SCHMITT,

Distributed Computer Systems (DISCO) Lab, University of Kaiserslautern

Networks are integral parts of modern safety-critical systems and certification demands the provision of guarantees for data transmissions. Deterministic Network Calculus (DNC) can compute a worst-case bound on a data flow's end-to-end delay. Accuracy of DNC results has been improved steadily, resulting in two DNC branches: the classical algebraic analysis and the more recent optimization-based analysis. The optimization-based branch provides a theoretical solution for tight bounds. Its computational cost grows, however, (possibly super-)exponentially with the network size. Consequently, a heuristic optimization formulation trading accuracy against computational costs was proposed. In this article, we challenge optimization-based DNC with a new algebraic DNC algorithm.

We show that:

(1) no current optimization formulation scales well with the network size and
(2) algebraic DNC can be considerably improved in both aspects, accuracy and computational cost.

To that end, we contribute a novel DNC algorithm that transfers the optimization's search for best attainable delay bounds to algebraic DNC. It achieves a high degree of accuracy and our novel efficiency improvements reduce the cost of the analysis dramatically. In extensive numerical experiments, we observe that our delay bounds deviate from the optimization-based ones by only 1.142% on average while computation times simultaneously decrease by several orders of magnitude.

1 INTRODUCTION

Accurately bounding timing constraints is a fundamental problem of systems analysis. Applied in the design phase of communication networks, it allows for their certification while preventing over-provisioning of resources. For networks, the main resource is the forwarding capability of links and the crucial metric to consider is the end-to-end delay of data flows. An example of a networked system requiring certification are the Ethernet-based Avionics Full-Duplex (AFDX) data networks embedded in modern Airbus aircraft. These have to be verified against
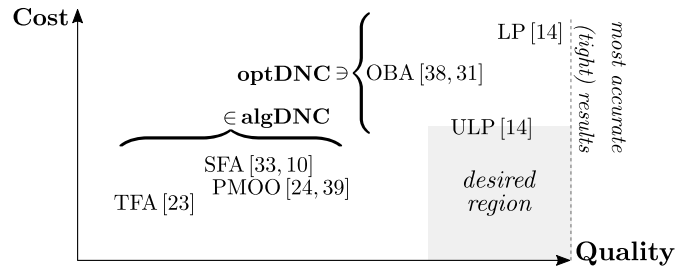
Fig. 1. Assumed quality and cost of Deterministic Network Calculus (DNC) analyses, based on insufficient evaluation data. While all analyses from the algebraic branch (algDNC) are "believed" to be of insufficient accuracy, the optimization's (optDNC's) ULP is currently regarded as the only analysis of good quality as well as just feasible to execute.

strict deadlines in order to attain the necessary certification. Given these demands and a formal worst-case model of the network, Deterministic Network Calculus (DNC) can provide worst-case bounds on the communication delay in general feed-forward networks. Indeed, DNC was used for the certification of the AFDX backbone as found in the Airbus A380 [26, 27]. Other recent example applications for DNC can be found in shared networked storage in order to meet tail latency QoS [45] and in multi-tenant data centers [29].

DNC network analysis already allowed to derive end-to-end delay bounds at an early stage [23], yet, achieving accurate results has turned out to be a hard problem. DNC's evolution in the pursuit of ever more accurate delay bounds has led to two branches: algebraic DNC (algDNC), as used in the verification of AFDX, followed by optimization-based DNC (optDNC). Both share the same network model, yet, vastly differ in the tools they apply to derive delay bounds, i.e., the actual network analysis. AlgDNC uses its composition rules for operators to derive an equation bounding the delay. OptDNC, on the other hand, derives a linear program formulation whose solution bounds the end-to-end delay.

OptDNC, the latest evolutionary step, is the more accurate branch of DNC. In theory, it can compute most accurate, i.e., tight, bounds, yet the effort to execute the according LP analysis grows (possibly super-)exponentially with the network size. In fact, it becomes computationally infeasible to analyze networks of even rather small size and its authors accompanied their exact theoretical solution with a practical variant, an optDNC heuristic called ULP [14].

Unfortunately, the current plethora of DNC analyses, most of them heuristics, was never evaluated comprehensively. Questions regarding their fundamental tradeoff between quality (accuracy) and cost remain open. Especially w.r.t. computational cost, there are crucial knowledge gaps. Figure 1 sketches the currently "believed" quality and cost relations of DNC analyses. These are vague estimates based on very scarce evaluation data. We set out to fill that gap. This allows us to make the following contributions:

(1) Current optDNC design does not provide a fast heuristic for attaining delay bounds in feed-forward networks.

Both optDNC analyses of [14], LP and ULP, follow the same non-compositional design principle. The given heuristic provides the only known way to trade accuracy against computational efficiency and was believed to be just in the desired region of feasibility and high quality (see Figure 1). In this article, we show that this heuristic is actually very costly and only applicable to small networks – though, it constitutes the maximum reduction of effort by reducing the optimizations to a single (but huge) linear program. The design principle thus inherently limits the efficiency of the analysis.

(2) Identification of current algDNC analyses' problems. Their impact needs to be minimized to improve algDNC accuracy.

(a) Sample network with 20 devices.

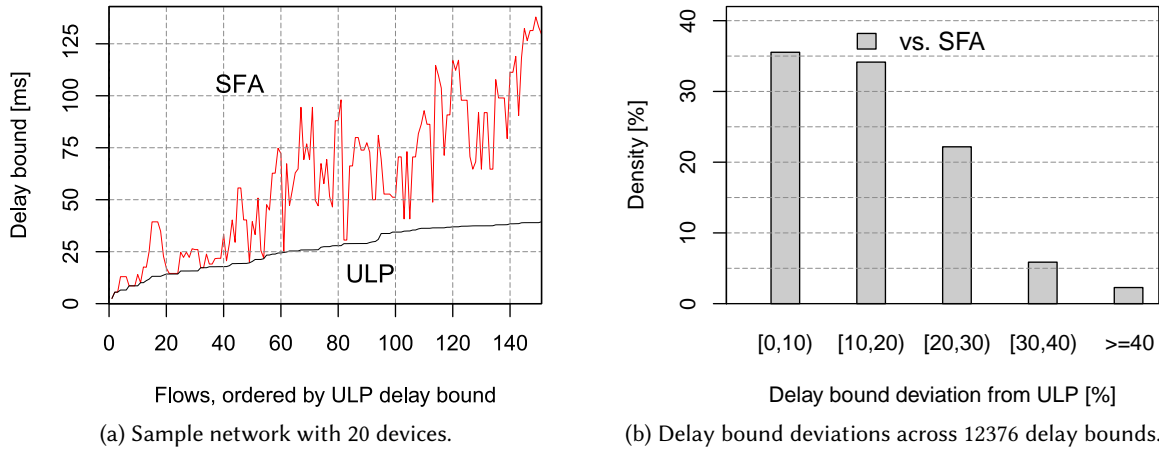(b) Delay bound deviations across 12376 delay bounds.

Fig. 2. Delay bounds, algDNC (SFA) vs. optDNC (ULP).

The DNC branch of choice in the industrial context, especially avionics, is the inaccurate algDNC [18, 19, 25, 26, 36]. Therefore, we derive in-depth knowledge of its problems, the impact of which needs to be minimized. We turn this knowledge into an analysis design that achieves our objective.

The algDNC is computationally attractive due to its compositional approach, yet, its inaccuracy was the very reason for DNC's branching into optimization-based analyses. Regarding quality, algDNC, represented by the recent SFA of [10], cannot compete with optDNC's ULP heuristic. Figure 2a depicts flow delay bounds in a small network of 20 devices. Already in this small network, the algebraically derived SFA bounds oscillate wildly with a large amplitude above the ULP results. This behavior can be observed in larger networks as well. We have taken statistics over 9 different topology sizes, amounting to 12376 individual flow delay bounds (cf. Table 2, left, in Appendix A.2). This first comprehensive comparison of an algDNC network analysis and an optDNC one reveals the gap in accuracy we set to overcome in this article. SFA delay bounds deviate from the respective ULP bounds by an average of $\approx 15.2\%$, the maximum is even as large as 72.65%. Figure 2b shows a more fine-grained distribution of deviations.

Hence, DNC currently offers an accurate, yet, computationally infeasible branch and a reasonably fast, but inaccurate branch of network analyses.

(3) Design of a novel algDNC analysis algorithm, the first DNC heuristic that is accurate and fast at the same time.

Our new algDNC analysis achieves its accuracy by incorporating concepts from optDNC's analysis design. Consequently, it becomes computationally infeasible at first sight. Yet, we provide an algorithm that vastly reduces the computational cost while accuracy of delay bounds is unchanged. Both is shown by extensive benchmarks where our new analysis achieves delay bounds that deviate from the ULP by an average of only 1.142%. Moreover, our algorithm bounds delays several orders of magnitude faster than the ULP.

The remainder of this article is structured as follows: Section 2 describes the modeling background of DNC. In Section 3, we evaluate the cost of optDNC and in Section 4 we derive the problems causing loss of accuracy in algDNC, leading to an analysis procedure that minimizes their impact. In Section 5 we design a novel algorithm for this analysis that is accurate and fast at the same time. Section 6 presents a comprehensive numerical evaluation

of all presented algDNC and optDNC analyses, ranking them regarding their quality and cost. Section 7 relates our contribution to the literature and Section 8 concludes the article.

2 THE COMMON NETWORK CALCULUS MODEL

2.1 Data Arrivals and Forwarding Service

In DNC, flows are characterized by functions that cumulatively count their data [22]. These functions are non-negative, wide-sense increasing and pass through the origin:

$$\mathcal{F}_0 = \left\{ f : \mathbb{R} \to \mathbb{R}_\infty^+ \;\middle|\; f(0) = 0, \; \forall \tau \le t : f(\tau) \le f(t) \right\}, \mathbb{R}_\infty^+ := [0, +\infty) \cup \{+\infty\} .$$

A flow's functions $A(t)$ and $A'(t)$ describe its data put into a server s and its data put out from s, both from the start of arrivals up until time t. Given these functions, the expected delay of a data unit arriving at s can be computed.

Definition 1. (Delay) Assume a flow with input function A traverses a server s and results in the output function A'. The *(virtual)* delay for a data unit arriving at s at time t is

$$D(t) = \inf \left\{ \tau \ge 0 \mid A(t) \le A'(t + \tau) \right\}.$$

Note, that this definition requires the server to preserve the order of the flow's arrivals when forwarding its data.

In a second modeling step, network calculus introduces bounding functions for data arrivals, so-called arrival curves.

Definition 2. (Arrival Curve) Let a flow have input function $A \in \mathcal{F}_0$, then $\alpha \in \mathcal{F}_0$ is an arrival curve for A iff it bounds A in any observation window of duration d, i.e.,

$$\forall t \; \forall d, \; 0 \le d \le t : A(t) - A(t - d) \le \alpha(d).$$

Using arrival curves, only the duration of observation is required to obtain an upper bound on the cumulative arrivals of data. I.e., flow arrivals' absolute starting point in time as well as the history up until time t need not be known anymore.

A useful basic shape for an arrival curve is the so-called token bucket. It is enforced by the eponymous traffic regulation algorithm. These curves are from the set $\mathcal{F}_{TB} \subseteq \mathcal{F}_0$,

$$\mathcal{F}_{TB} = \left\{ \gamma_{r,b} \;\middle|\; \gamma_{r,b}(d) = \begin{cases} 0 & \text{if } d = 0 \\ b + r \cdot d & \text{otherwise} \end{cases} , \; r, b \in \mathbb{R}_\infty^+ \right\},$$

where r denotes the maximum arrival rate and b is the maximum burstiness (bucket size).

Scheduling and buffering at a server result in the output function $A'(t)$. Network calculus captures the minimum forwarding capabilities that lead to A' in interval time as well:

Definition 3. (Service Curve) If the service provided by a server s for a given input A results in an output A', then s is said to offer a service curve $\beta \in \mathcal{F}_0$ iff

$$\forall t : A'(t) \ge \inf_{0 \le d \le t} \{A(t - d) + \beta(d)\}.$$

A number of servers fulfill a stricter definition of service curves by considering their state in addition to input A.

Definition 4. (Backlogged Period) A server s with input A and output A' is backlogged during period $(\underline{t}, \bar{t})$, if $\forall t \in (\underline{t}, \bar{t}) : A(t) > A'(t)$.

Servers offering strict service guarantees have a higher output guarantee during any backlogged period.

Definition 5. (Strict Service Curve) If, during any backlogged period of duration d, a server s with input A and output A' guarantees an output of at least $\beta(d)$, it offers a *strict* service curve $\beta \in \mathcal{F}_0$.

A basic shape for service curves is the rate latency function defined by the set $\mathcal{F}_{RL} \subseteq \mathcal{F}_0$,

$$\mathcal{F}_{RL} = \left\{ \beta_{R,T} \mid \beta_{R,T}(d) = \max\{0, R \cdot (d - T)\}, \ R, T \in \mathbb{R}_\infty^+ \right\},$$

where R denotes the minimum service rate and T is the maximum latency. Networks where curves are exclusively from the sets $\mathcal{F}_{TB}$ and $\mathcal{F}_{RL}$ have been in the focus of recent advances in DNC [6, 15].

Given these bounding curves, the maximum delay experienced by a flow when crossing a server is simply computed by the horizontal deviation between arrival curve α and service curve β:

$$\sup_{d \geq 0} \left\{ \inf \left\{ \tau \geq 0 \ : \ \alpha(d) - \beta(d + \tau) \leq 0 \right\} \right\}.$$

2.2 The Network Model

2.2.1 Arbitrary Multiplexing. Consistent with our worst-case perspective, we make no assumption on the ordering between flows when they are multiplexed at servers. That is we employ the so-called arbitrary multiplexing when computing left-over service for a flow of interest at a shared server. Certainly, if we could, for instance, assume FIFO multiplexing, then the performance bounds could be improved. However, in applications requiring worst-case guarantees such as, e.g., for the certification in avionics, a FIFO assumption may already be considered too optimistic. In fact, many network switches use highly optimized internal switching fabrics that can lead to reordering in order to avoid head-of-line blocking [31]. Overall, by using arbitrary multiplexing we play safe with respect to the worst-case.

2.2.2 The Feed-forward Property. A second assumption of current DNC analyses for networks is the absence of cyclic dependencies between flows. Work departing from this assumption can be found in [30, 39] but is not covered by this article. Therefore, we focus on networks that guarantee the feed-forward property by design. I.e., the analyzed network does not allow for cycles. In Appendix A, we present a means to generate feed-forward networks for evaluation of DNC analyses.

3 OPTIMIZATION-BASED DNC

In this section, we perform a first cost evaluation of optDNC in larger feed-forward networks. Bounding worst-case delays tightly is a NP-hard problem [14]. Therefore, we examine optDNC's LP analysis that derives these tight bounds and provide deeper insights into the computational cost of optDNC's less accurate heuristic, the ULP. We compare our results to the literature's algebraic analysis SFA, as used in recent work of [10, 14]. The comparison reveals that that DNC currently requires to choose between a computationally barely feasible, yet accurate optimization analysis and a feasible, but inaccurate algebraic analysis.

3.1 The Tight LP Analysis

OptDNC's LP analysis [14] takes the following steps to derive a given flow of interest's (foi's) tight delay bound:

(1) Starting at the foi's sink server, *backtrack* all flows in order to derive the dependencies between starts of backlogged periods of servers. For simple tandems of servers, this step results in a total order. However, in more general feed-forward networks, it derives a partial order.
(2) The partial order is extended to the set of all *compatible total orders*. This step enumerates all relations of servers' backlogged period beginnings. It is subject to restrictions caused by flows that rejoin again after demultiplexing.

(3) Each total order is converted to one linear program that also includes the network description's constraints such as arrival curves and service curves (Section 2.1). The *maximum of all their solutions* constitutes the tight delay.

The third step shows that the LP is an all-or-nothing analysis where we cannot judge validity of a delay bound before computing all of them. Unfortunately, step 2 is prone to a combinatorial explosion that constitutes the underlying reason for this DNC analysis' (possibly super-)exponential growth in effort. To illustrate this problem, let us briefly discuss on the number of linear programs (LPs) that have to be solved in a sink-tree network: In the best case, we have a pure tandem network of n servers and then a single LP results; in the worst case, we have a so-called fat tree with one root node and $n - 1$ leaf nodes directly connected to it, resulting in $(n - 1)!$ LPs. In a full binary tree, the number of LPs is lower bounded by $\Omega\left(\left(\frac{n}{2}\right)!\right)$ [38]. In general, calculating the number of total orders being compatible with a given partial order is itself not a simple problem. One solution is the Varol-Rotem algorithm [44]; we implemented this algorithm to provide some numbers for the case of full k-ary trees (Table 1). It is obvious that the computational cost to solve such large numbers of linear programs becomes prohibitive quickly, even for moderately sized networks.

Table 1. Number of LPs to solve for full k-ary trees of moderate size.

		Height h			
		0	1	2	3
	1	1	1	1	1
Outdegree k	2	1	2	80	$21,964,800$
	3	1	6	$7,484,400$	$3.54 \cdot 10^{37}$
	4	1	24	$3.89 \cdot 10^{15}$	$1.12 \cdot 10^{110}$

3.2 The Accurate, but Costly ULP Heuristic

Based on the LP's optimization formulation and the analysis design it was accompanied with, a heuristic called Unique LP (ULP) was proposed [14]. It circumvents the combinatorial explosion by skipping step 2 from above. I.e., it always derives a single linear program that is directly based on the partial order from step 1. Thus, the ULP does not attain tight delays but it was shown to stay very close to these in a (small) sample network. Its computational cost in terms of analysis run time [28, 33] in larger, more general feed-forward networks had not been investigated yet.

3.2.1 Inaccuracy due to Paying Segregation more than Once. The ULP models multiplexing of flows with joint backlogged periods and has access to global knowledge in the optimization step. Yet, recent work [9] shows how skipping step 2 reduces the accuracy of bounds attained by the ULP. In the following situation, these two features of optDNC are only utilized by the LP analysis: Assume flows multiplex at a server and demultiplex after it, then they take different paths and multiplex later at a different server again. In such a network, the analysis should strive for the Pay Segregation Only Once (PSOO) principle – separated paths should not result in segregated flow analysis and overly pessimistic service allocation to flows. Yet, the ULP cannot capture the (global) interdependency between the former and the latter server where the flows jointly cause a backlogged period. Instead, it indeed derives an overly-pessimistic worst-case result by violating the PSOO principle. This PSOO violation, however, occurs in all known DNC heuristics, optimization and algebraic analysis.

3.2.2 Computational Cost Evaluation of the ULP. We implemented the ULP to measure its analysis execution time – deriving the partial order, the linear program and solving it – for our set of feed-forward networks (Appendix A.2). For the optimization, we employ two different solvers: the open-source LpSolve 5.5.2.0 and

(a) Scaling of network analysis time.
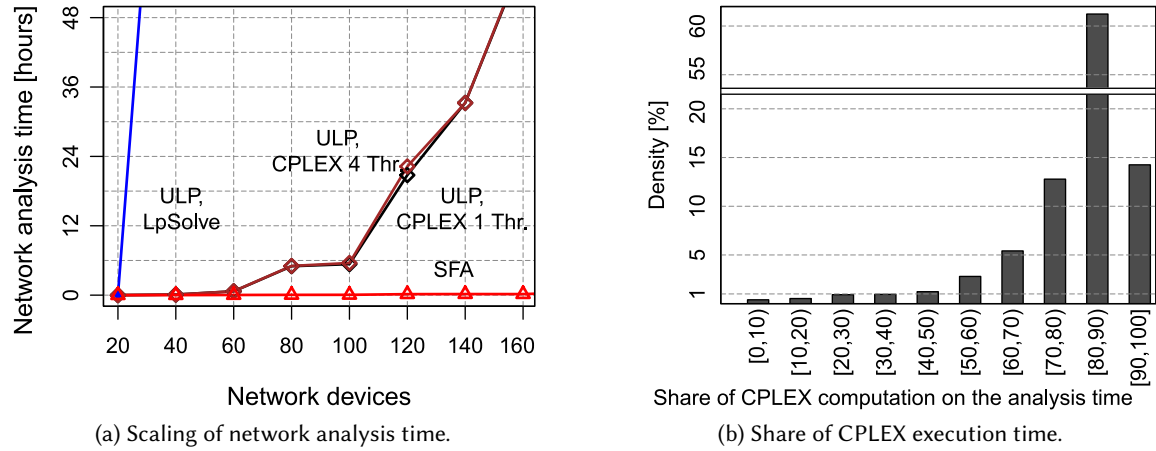
(b) Share of CPLEX execution time.

Fig. 3. Analysis cost (run time of a network analysis).

IBM CPLEX 12.6.2. All computations were executed on a physical machine equipped with two Intel Xeon E5420 server CPUs (4 physical cores each) running with a clock speed of 2.5GHz and a total of 12GB RAM.

Figure 3a shows our results and compares them to algDNC's SFA (see Section 4.1.2, [10]), the contender used in [14]. These results defeat the hope in the ULP. The choice of tooling becomes crucial quickly as LpSolve already struggles with 40 devices, with regard to the computation time as well as derivations of results. IBM CPLEX performs considerably better, yet, the ULP analysis time still increases very fast with the network size. This trend is decisively negative as, among the three steps, optimization with CPLEX takes on average > 80% of the analysis time (cf. Figure 3b). For instance, the ULP analysis already requires 13 days at less than 200 devices. Moreover, applying it in a design space exploration for a network with as little as 80 devices does not seem feasible either as the analysis time of a single alternative exceeds 5 hours (cf. Figure 3a). The SFA, on the other hand, finishes analyzing these networks in just a small fraction of these times.

4 QUALITY OF ALGEBRAIC DNC: PROBLEMS AND PROSPECTS

Algebraic DNC (algDNC) derives a (min-plus algebraic) equation that computes the delay bound for the flow of interest. We show that the algDNC analyses currently do not derive the best equation possible and we provide the theoretical foundation to solve this issue. In consequence, the best algDNC delay bound can now be obtained.

4.1 (min,+)-Algebraic Calculus

First, we provide the basic results we rely on in algDNC and thus in our accurate and fast analysis design.

4.1.1 Basic Operations. AlgDNC's equations consist of (min,+)-operations [21, 34] that are computationally attractive. E.g., for the above curves in $\mathcal{F}_{\mathrm{TB}}$ and $\mathcal{F}_{\mathrm{RL}}$, they can be implemented in $O(1)$ [17].

Definition 6 ((min, +)-*Operations*). *The* (min, +)-*algebraic operations for two network calculus curves* $f, g \in \mathcal{F}_0$ are:

- *Aggregation:* $(f + g)(t) = f(t) + g(t)$
- *Convolution:* $(f \otimes g)(t) = \inf_{0 \leq u \leq t} \{f(t - u) + g(u)\}$
- *Deconvolution:* $(f \oslash g)(d) = \sup_{u \geq 0} \{f(d + u) - g(u)\}$

The service curve definition then translates to $A' \geq A \otimes \beta$, the arrival curve definition to $A \otimes \alpha \geq A$, and performance bounds can be derived using the deconvolution operation:

Theorem 7 (Performance Bounds). *Consider a server s that offers a service curve β. Assume flow f with arrival curve α traverses s. We get the following bounds for f:*

- *Delay: $\forall t \in \mathbb{R}^+ : \ D(t) \leq \inf \{d \geq 0 \,|\, (\alpha \oslash \beta)(-d) \leq 0\}$*
- *Output: $\forall d \in \mathbb{R}^+ : \ \alpha'(d) = (\alpha \oslash \beta)(d)$, where α' is an arrival curve[1] for A'.*

4.1.2 Tandem Analysis. AlgDNC focuses the delay analysis on a specific flow of interest (foi), end-to-end on its path (a tandem of servers).

Theorem 8 (Concatenation of Servers). *Let a flow f cross a tandem of servers $\mathcal{T} = \langle s_1, \ldots, s_n \rangle$ and assume that each s_i, $i \in \{1, \ldots, n\}$, offers a service curve β_{s_i}. The overall service curve offered to f is their concatenation*

$$\beta_{\mathcal{T}} \ = \ \bigotimes_{i=1}^{n} \beta_{s_i}.$$

Theorem 9 (Left-Over Service Curve). *Consider a server s that offers a strict service curve β_s. Let s be crossed by two flow aggregates $\mathbb{F}_0$ and $\mathbb{F}_1$ with arrival curves $\alpha^{\mathbb{F}_0}$ and $\alpha^{\mathbb{F}_1}$, respectively. Then $\mathbb{F}_1$'s worst-case resource share under arbitrary multiplexing at s, i.e., its left-over service curve, is*

$$\beta_s^{\mathrm{l.o.}\mathbb{F}_1} = \beta_s \ominus \alpha^{\mathbb{F}_0}$$

where $(\beta \ominus \alpha)(d) = \sup_{0 \leq u \leq d} (\beta - \alpha)(u)$ denotes the non-decreasing upper closure of $(\beta - \alpha)(d)$.

In this article, we use an open-source implementation of these operations [5]. An algDNC analysis of the flow's tandem then defines rules for their composition such that the model's worst case is retained.

Separate Flow Analysis (SFA) [34]. The first tandem analysis of algDNC, Separate Flow Analysis (SFA), constitutes a straight-forward application of Theorem 9 and Theorem 8: In order to derive the foi's service curve on its path $\mathcal{T} = \langle \ldots \rangle$, cross-traffic is subtracted server by server and the per-server $\beta^{\mathrm{l.o.}}$ are (min-plus) convolved. The resulting $\beta_{\mathcal{T}}^{\mathrm{l.o.}}$ is used to bound the foi's delay. This procedure has a disadvantage: If cross-flows share longer (sub-)paths with the foi, then multiplexing is considered more than once in the equation bounding the foi's delay.

Pay Multiplexing Only Once (PMOO) Analysis [24, 41]. PMOO is an analysis principle that tackles the SFA's problem. It suggests to convolve the analyzed sub-tandems of servers as much as possible before subtracting cross-traffic. We denote this computation by

$$\beta_{\mathcal{T}}^{\mathrm{l.o.}} = \beta_{\mathcal{T}} \ominus \left(\alpha^{\mathbb{F}_i}, \ldots, \alpha^{\mathbb{F}_{i+j}} \right)$$

where $\mathcal{T}$ is the analyzed tandem, $\beta_{\mathcal{T}}$ is its convolved service curve and $\left(\mathbb{F}_i, \ldots, \mathbb{F}_{i+j} \right)$ denotes the cross-flow aggregates to subtract. For single-server tandems $\langle s \rangle$, the derivation equals Theorem 9, i.e., $\beta_{\langle s \rangle}^{\mathrm{l.o.}} = \beta_{\langle s \rangle} \ominus \left(\alpha^{\mathbb{F}_i}, \ldots, \alpha^{\mathbb{F}_{i+j}} \right) = \beta_s \ominus \sum_{k=0}^{j} \alpha^{\mathbb{F}_{i+k}}$. For larger tandems and more involved interference patterns, [41] provides the first theoretical implementation of this principle (code is in [5]). We use this PMOO analysis in our article. It results in more accurate bounds than SFA in many scenarios, with some noticeable exceptions which we detail later [40].

---

[1]Note, that arrival curves need to pass through the origin. In a slight abuse of notation, we use the symbol $\oslash$ for both, deconvolution and output bounding.

Validity of PMOO Results. The PMOO's underlying principle to convolve service curves before subtracting the impact of cross-traffic arrivals is, in fact, not covered by the basic rules to compose algDNC operations. The left-over service curve operation can only be applied to *strict* service curves as only these guarantee a bound on the backlogged period. In the worst case, this bound needs to be reached in order to assume left-over service. Yet, the convolution of two strict service curves does not necessarily result in a strict service curve. Correctness of the PMOO result is guaranteed nonetheless. The algDNC literature offers multiple justifications: On the one hand, it was proved that the algebraic manipulations are inherently pessimistic, such that the optimistically assumed bound on the backlogged period of simple service curves is set off. This is shown by proving that (valid) results of an optimization analysis are always better than PMOO results [32, 40]. Hence, the PMOO result constitutes a valid bound as well. On the other hand, more direct proofs arguing over input/output relations of flows crossing a tandem of servers can be found in the literature, too [11, 13].

4.2 Compositional Feed-forward Analysis: From Tandem Analyses to Network Analyses

The algDNC tandem analyses are used in a compositional fashion to derive delay bounds in feed-forward networks (compFFA). Conceptually, the procedure can be split into two parts [6]:

i) The analysis abstracts from the feed-forward network to the foi's path. In this part, arrivals of cross-traffic are bounded at the locations of interference with the foi.

ii) A tandem analysis derives the foi's delay bound.

Whereas the tandem to analyze in the second part is known from the start, the tandems in the first part need to be derived. I.e., the network must to be decomposed into a sequence of tandems to analyze (derive $\beta^{\text{l.o.}}$s). This decomposition depends on the applied analysis. Tandems are then interfaced with the output bound operation $\oslash$. We illustrate the decomposition in Figure 4a, a minimal network that requires both parts of the procedure. Given the global view in this sample network, we see that SFA decomposes each flow's paths into smallest possible tandems (i.e., individual servers). In contrast, the PMOO analysis "decomposes" into longest possible tandems to apply its left-over service curve derivation to.

These decompositions define orders of algebraic operators in order to attain worst-case results. However, current algDNC analyses do not exploit all degrees of freedom to decompose a feed-forward network into tandems. This can cause pessimism and reduce the accuracy of results. In the following, we provide detailed insight into these aspects. Last, note that the presented optDNC analyses are not compositional. They directly derive the foi's delay bound.

Causes for Overly-pessimistic Results of compFFA Heuristics. We provide an analysis of problems encountered in compFFA. There are several situations that algebraic DNC's compositional feed-forward analysis can only handle by overly-pessimistic worst-case approximations. Detailed knowledge about them enables us to derive mitigation strategies that reduce occurrences to a minimum. We present newly found phenomena as well as known ones that lack comprehensive evaluation. All are considered in the design of our novel algebraic analysis algorithm. Our mitigation strategies exploit unused degrees of freedom in composing algDNC equations. Thus, the derived worst-case bounds are less pessimistic than the known ones but still more pessimistic than optDNC's ULP results. In Section 6, we evaluate the respective accuracy differences.

4.2.1 Pay Segregation Only Once Violations in compFFA. Compositional feed-forward analysis compFFA causes the same PSOO violations as presented in Section 3.2.1. When backtracking flows, the left-over service curve operation computes a worst-case service share for this flow. I.e., the result is computed under local information only; interdependencies between flows that multiplex at a server s_n, demultiplex after it, take different paths and multiplex later at a different server s_m again is not considered. If two flows f_i and f_j are each backtracked from s_m to s_n, they compute their respective left-over service curves $\beta_{s_n} \ominus f_i$ and $\beta_{s_n} \ominus f_j$ at server s_n. Both

(a) Minimal network with both compFFA parts.

(b) Construction of PMOO's (min,+)-equation.

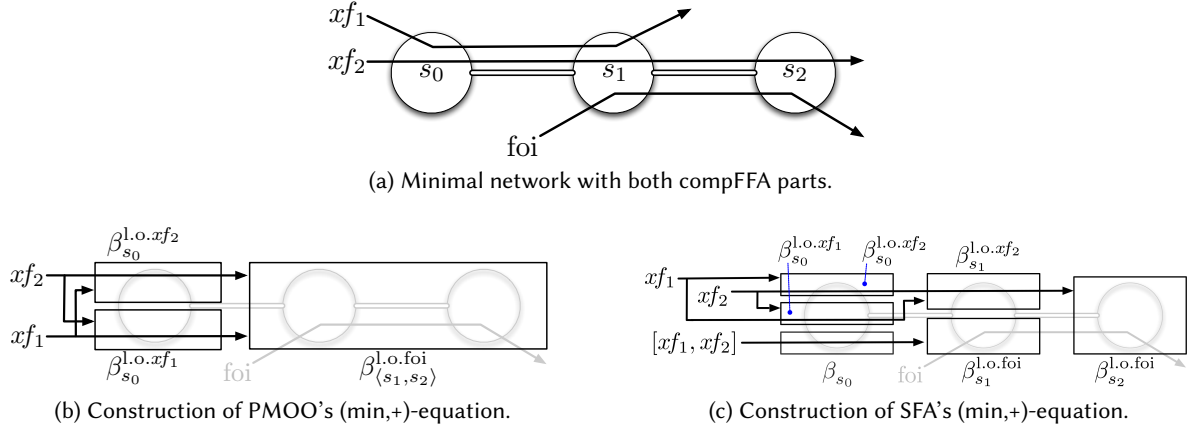(c) Construction of SFA's (min,+)-equation.

Fig. 4. Network with derivations of (min,+)-equations that illustrate the restricting design of compFFA (Section 4.2.2).

Interpretation: Boxes depict tandems for $\beta^{\text{l.o.}}$ computation and arrows depict flows. Flows pointing at a box are subtracted from a β, flows crossing a box use the respective $\beta^{\text{l.o.}}$ to bound their output from it.

flows simultaneously assume that the opposite flow has a higher priority and is served first. This composition of algebraic operations is permissible, meaning it represents a worst case, leading to valid results. However, the total service assumed to be available to both flows is less than the server's minimum guarantee β_{s_n}. It is shown that PMOO and ULP suffer from this PSOO violation equally, i.e., compositional PMOO analysis cannot outperform the ULP in this basic scenario [9].

However, in compFFA, there are additional causes for flow segregation. These can eventually result in PSOO violations and thus reduce accuracy of algebraically derived bounds. In pursuit of the PSOO principle, i.e., the least pessimism attainable while preserving correctness of worst-case bounds, we attempt to mitigate these. Aggregate bounding of cross-flows was shown to be such a mitigation strategy [7]. Yet, the PMOO's $\beta_{\mathcal{T}}^{\text{l.o.}}$-derivation enforces a segregation of cross-flows that interfere with the foi on different subpaths. This is the case for xf_1 and xf_2 in Figure 4b. We obtain

$$\beta_{\langle s_1, s_2 \rangle}^{\text{l.o.foi}} = \beta_{\langle s_1, s_2 \rangle} \ominus \left(\alpha_{s_1}^{xf_1}, \alpha_{s_1}^{xf_2} \right) = \beta_{\langle s_1, s_2 \rangle} \ominus \left(\alpha^{xf_1} \oslash \beta_{s_0}^{\text{l.o.}xf_1}, \alpha^{xf_2} \oslash \beta_{s_0}^{\text{l.o.}xf_2} \right).$$

The foi sees overly pessimistic cross-traffic arrival bounds compared to a hypothetical version that aggregates cross-traffic during arrival bounding. AlgDNC does not provide a tandem $\beta_{\mathcal{T}}^{\text{l.o.}}$ achieving $\beta_{\langle s_1, s_2 \rangle} \ominus \left(\alpha_{s_1}^{[xf_1, xf_2]} \right) = \beta_{\langle s_1, s_2 \rangle} \ominus \left((\alpha^{xf_1} + \alpha^{xf_2}) \oslash \beta_{s_0} \right)$ in Fig. 4a. Other DNC tandem analyses implementing the PMOO principle for arbitrary multiplexing servers, namely (min,+) multi-dimensional convolution [11, 12] and OBA [32, 40], also require segregate bounding of cross-traffic and thus cause the same PSOO violation.

The SFA looses awareness of the foi's path in its single-server analysis. Therefore, it may even segregate cross-flows on this tandem. On the other hand, it can aggregate all cross-flows sharing a single hop with the foi. Both situations are illustrated in Figure 4c: Deriving $\beta_{s_1}^{\text{l.o.foi}}$ can use the cross-traffic aggregate $[xf_1, xf_2]$. $\beta_{s_2}^{\text{l.o.foi}}$'s computation enforces the a PSOO violation, indicated by the presence of $\beta_{s_0}^{\text{l.o.}xf_1}$ and $\beta_{s_0}^{\text{l.o.}xf_2}$, like in the PMOO equation:

$$\beta_{\langle s_1, s_2 \rangle}^{\text{l.o.foi}} = \beta_{s_1}^{\text{l.o.foi}} \otimes \beta_{s_2}^{\text{l.o.foi}} = \left(\beta_{s_1} \ominus \left(\alpha^{[xf_1, xf_2]} \oslash \beta_{s_0} \right) \right) \otimes \left(\beta_{s_2} \ominus \left(\alpha^{xf_2} \oslash \left(\beta_{s_0}^{\text{l.o.}xf_2} \otimes \left(\beta_{s_1} \ominus \left(\alpha^{xf_1} \oslash \beta_{s_0}^{\text{l.o.}xf_1} \right) \right) \right) \right) \right)$$

4.2.2 Restricting Design of compFFA. The above compFFA's strict separation of its two parts is also reflected in the algDNC equations. Both, SFA of Figure 4c and PMOO of Figure 4b, assume that tandems cannot reach over the foi's path on the left. A global view on the network reveals that another decomposition is permissible without violating worst-case assumptions. I.e., we can match tandems onto the network in a different way:
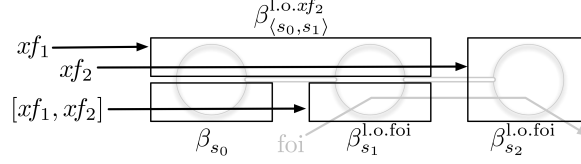

Fig. 5. Third alternative to match tandems onto Figure 4a.

The according (min,+)-equation shows a reduction of enforced segregation in an SFA-like derivation:

$$\beta^{\text{l.o.foi}}_{\langle s_1, s_2 \rangle} = \beta^{\text{l.o.foi}}_{s_1} \otimes \beta^{\text{l.o.foi}}_{s_2} = \left(\beta_{s_1} \ominus \left(\alpha^{[xf_1, xf_2]} \oslash \beta_{s_0} \right) \right) \otimes \left(\beta_{s_2} \ominus \left(\alpha^{xf_1} \oslash \underbrace{\left(\beta_{\langle s_0, s_1 \rangle} \ominus \left(\alpha^{xf_2} \right) \right)}_{\beta^{\text{l.o.}xf_2}_{\langle s_0, s_1 \rangle}} \right) \right)$$

where $\beta^{\text{l.o.}xf_2}_{\langle s_0, s_1 \rangle}$ stretches over the boundaries of both compFFA parts. Thus, with global knowledge about the network, we can identify more degrees of freedom when decomposing the network into a sequence of tandem analyses. In the above example, this can result in arbitrarily better delay bounds (see Appendix B), similar to the results presented in [8, 9]. Yet, in Section 6 we evaluate our finding in a broader scope than a selected extreme case. Moreover, we add a cost evaluation as our finding demands additional analysis effort to be exploited.

4.2.3 Restricting Design of Tandem Analyses. Currently, decomposition of the network is defined by the tandem analysis to be used; it is independent of the compFFA procedure. We illustrate this on another small sample network (Figure 6a). As already mentioned, PMOO decomposes into longest possible tandems (see Figure 6b) and SFA decomposes into smallest possible tandems (see Figure 6c). Note, that there are no segregation effects in these equations for tandem analysis. In [40], it was shown that the SFA can outperform the PMOO if the foi's left-over service strongly differs between servers. The authors designed the first optDNC optimization

(a) Minimal network with decomposition effects.

(b) Construction of PMOO's (min,+)-equation.

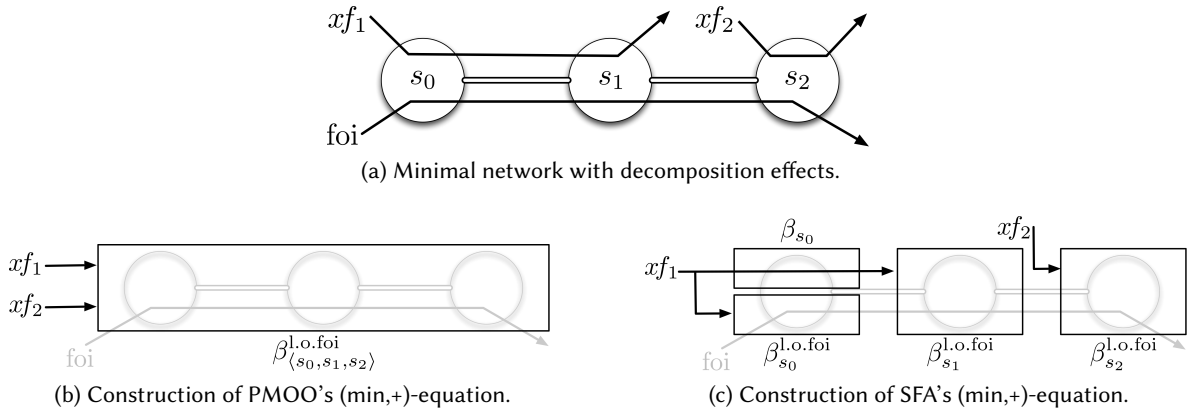(c) Construction of SFA's (min,+)-equation.

Fig. 6. Network and derivations of (min,+)-equations illustrating the restricting design of tandem analyses (Section 4.2.3).

formulation to overcome this problem – a tandem analysis that integrates with the compFFA – but its cost renders an application in feed-forward networks infeasible [32]. Therefore, we propose to stay with algDNC but decouple the decomposition of a network into tandems from their analysis. We allow for the following superior derivation that can exploit the PMOO principle at the start of the foi's path as well as a fast left-over service curve $\beta_{s_2}^{\text{l.o.foi}}$ at the end:
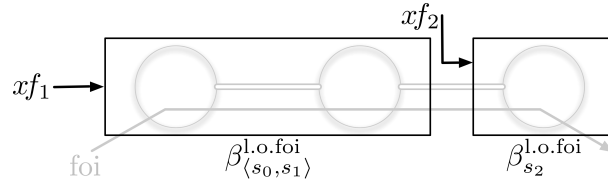
4.2.4 Decomposition on Insufficient Information.
We presented the previous two insights to already reveal countermeasures to the problems of algDNC. However, it is not possible to derive the best tandem decomposition from the network model in a static fashion. Neither of the alternatives is strictly superior to the others, i.e., all alternatives need to be assessed based on their actual cross-traffic arrival bounds and left-over service curves. Depending on the analyzed tandem's location in the feed-forward network, the required preceding computations can already be very costly. Starting with a single decomposition (e.g., SFA or PMOO) and eventually recognizing that the analysis can be improved, requires additional feed-forward analyses for the decomposition alternatives.

4.2.5 Overly-pessimistic Cross-traffic Burstiness [8].
A recent improvement to algDNC was proposed in [8]. This work identifies a problem that occurs when the burstiness of a flow is bounded after crossing a server, i.e., when applying Theorem 7, Output, to compute the output bound's relevant part $\alpha'(0) = \left(\alpha \oslash \beta^{\text{l.o.}}\right)(0)$. The computation relies on the deconvolution of the analyzed flow's arrival curve α with the crossed server's left-over service curve for it, $\beta^{\text{l.o.}}$. This deconvolution-based computation is ignorant about the server's queue. It was shown that this can lead to overly-pessimistic bounds on the worst-case burstiness. During compFFA's step i), this inaccuracy propagates through the cross-traffic arrival bounding and eventually leads to overly-pessimistic flow delay bounds. In [8], it is propose to derive the worst-case queue length at a server and use this information to cap the output burstiness $\alpha'(0)$. This finding suggests that neither solution to compute bounds on output burstiness is tight. However, it was only theoretically evaluated in an artificial network designed to provoke the burst cap mechanism. An evaluation in realistic networks, benchmarks against optDNC and an evaluation of the additional effort are lacking.

4.3 Derivation of all Permissible (min,+)-Equations

Given the new insights about compositional algDNC analysis we derived in this section, we propose to exhaustively derive all permissible (min,+)-equations. An equation is permissible if the result constitutes a valid worst-case delay bound for the foi. In theory, we execute the following three steps that somewhat resemble the optDNC feed-forward analysis:

(1) *Backtracking*: Starting at the foi's sink server, this step derives the dependencies of flows on each other. The backtracking progresses along the paths taken by flow aggregates in order to mitigate PSOO violations as much as possible.

(2) *(min,+)-equations*: The previous step's backtracking considers flow entanglements but still leaves the degrees of freedom presented in Section 4.2. We enumerate all alternatives to match tandems onto the intermediate network representation from step 1. Tandems are analyzed with the PMOO and interfaced with the burst-capped output bound. Thus, we derive all permissible (min,+)-equations that bound the foi's delay in the given feed-forward network.

(3) *Finding the best solution*: All equations are permissible and derive a valid delay bound. The minimum of all solutions is the most accurate valid delay bound.

Note, that the SFA and the PMOO analysis are algDNC heuristics that reduce the computational cost of steps 2 and 3 by reducing the search space to a single tandem matching / (min,+)-equation – just like the ULP only derives a single linear program. Therefore, our exhaustive decomposition scheme is guaranteed to be at least as good as the SFA and the PMOO analysis.

Whereas optDNC's LP analysis searched for the worst delay of any linear program, we search for the best delay bound of any algebraic (min,+)-equation. Our search space, the set of permissible equations bounding the foi's delay, is constructed subject to the result of the backtracking step and the composition rules for algDNC operators. Therefore, our search follows the objective to find the permissible algDNC (min,+)-equation that

- maximizes aggregation of cross-flows [7],
- minimizes the impact of enforced segregation,
- maximizes benefits of long tandems,
- maximizes gain from fast local left-over service curves, and
- caps overly pessimistic cross-traffic burstiness [8].

As our exhaustive procedure's design somewhat resembles the optDNC's LP, an obvious concern is its computational cost. As a matter of fact, the amount of decompositions and the involved operations increase with the network size, reaching 10^9 fast. Figure 7 illustrates the combinatorial explosion on a small network with just 32 devices. Routing 100 flows over this network, the worst-case amount of permissible (min,+)-equations of a single flow reaches close to 10^{20}, and for 300 flow, more than 10^{100} equations exist for some flows. Next, we design an efficient algorithm that allows to attain the same delay bound quality as this total enumeration approach, yet, it imposes only a tiny fraction of the cost.

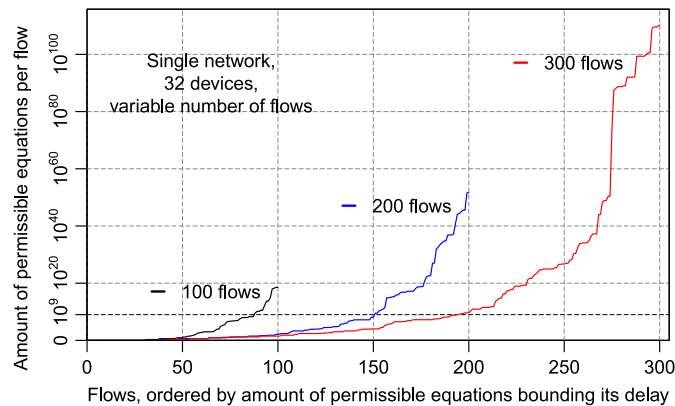Fig. 7. Combinatorial explosion in exhaustive enumeration of all permissible (min,+)-equations with the compFFA procedure.

5 EXHAUSTIVENESS & EFFICIENCY – A NOVEL ALGORITHM DESIGN FOR DNC

In this section, we provide an algorithm that efficiently computes the best delay bound attainable with the exhaustive set of permissible (min,+)-equations. In Section 6, we will benchmark it against optDNC.

5.1 Integrated Design

We depart from separately deriving global knowledge and the permissible (min,+)-equations. Instead, we unify steps 1 and 2 of Section 4.3's compFFA into a single one Step 1, the backtracking, was designed to derive information of the entanglement of flow aggregates and step 2 used this information derive a permissible (min,+)-equation based on the composition rules for (min,+)-operations. In this Section, we show that it is possible to unify both steps, yet, still consider all permissible equations and derive the best valid result.

We aim to derive all decompositions of a feed-forward network in a single step. The exhaustiveness we propose is defined by a multitude of alternatives that were previously considered in step 2. In order to allow our algorithm to attain the same set of permissible equations, we need to extend the new backtracking to consider alternatives as well. The solution is simple yet powerful. The analysis starts with a single tandem (the foi's path) and recursively backtracks cross-traffic flow aggregates over the tandems they jointly traverse until reaching the tandem currently analyzed. Given any tandem in this procedure, the integrated analysis decomposes it into all disjoint sequences of sub-tandems by deciding whether to decompose ("cut") the tandem at a link or not. On sub-tandems, the left-over service curve is computed and all sub-tandem ones are convolved into the tandem one for a specific decomposition alternative. In order to derive left-over service curves, the backtracking needs to be started each of sub-tandem, exhaustively working on decompositions again. The exhaustive derivation all sub-tandem decompositions mitigates the causes for compFFA's pessimism:

Section 4.2.1 Compositions that align with sub-path sharing of cross-flows are tested. They trade algDNC's need for segregated bounding against composite left-over derivation.

Section 4.2.2 Small sub-tandems on the foi's path are combined with large (sub-)tandems for cross-traffic arrival bounding.

Section 4.2.3 Neglected decompositions for the foi's are included.

Section 4.2.4 Sufficiently many decompositions are tested to find the best one for any tandem in the analysis.

Section 4.2.5 The output of each tandem can be easily capped with the backlog bound of its last server.

In fact, as the backtracking simultaneously considers the entanglement of flows (backtracking of flow aggregates) and the composition rules of (min,+)-operations, it derives the same set of permissible equations as presented in Section 4.3.

5.2 Efficiency Concerns and Improvements

The exhaustiveness of our proposed procedure naturally raises concerns about its computational costs. A tandem of length n can be decomposed into 2^{n-1} distinct sub-tandem sequences. All sub-tandem sequence of two adjacent levels of the backtracking recursion need to be combined in order to find the best attainable algDNC result. This consideration reveals a combinatorial explosion resulting in the large amount of permissible equations presented in Figure 7. In detail, a large search tree (Figure 8) is constructed where branching corresponds to the sub-tandem decomposition that, in turn, requires bounding of cross-traffic arriving on a tandem to cut. Each path through this search tree derives a single permissible (min,+)-equation. With our integrated design, we aim for an efficient solution to fully cover the search tree and achieve the objective to find the permissible equation for most accurate results. In algDNC, we can compute intermediate results by solving already derived parts of permissible (min,+)-equations. Their results can be used to cut down the search tree and thus improve efficiency.

5.2.1 Caching of Arrival Bounds. Disjoint sub-tandem decompositions only need to differ by a single link that was cut, so they often share many sub-tandems which all require the same derivations of traffic arrivals. These arrival bounds are solely defined by the sub-tandem, the flows to bound, and the foi – i.e., the path through the search tree that leads to a certain arrival bound (black dot in Figure 8) does not matter. Caching and reusing arrival bounds therefore reduces the computational effort by allowing the analysis to terminate a search before reaching any of the search tree's leaves.
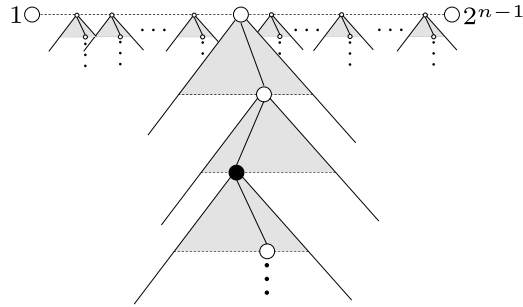
Fig. 8. Search tree [1] for permissible (min,+)-equations. The black dot marks a computation of cross-traffic arrivals.

5.2.2 Convolution of Alternative Arrival Bounds. In the search tree, effort spreads over adjacent levels: Each decomposition results in one left-over service curve that computes one arrival bound, all of which are combined with those of the next higher level. In an algDNC analysis, we compute these arrival bounds as internal, intermediate results and will use them to counteract the combinatorial explosion. All of these bounds are valid arrival curves and therefore their convolution is a valid arrival curve as well [34]. This countermeasure reduces the amount of arrival bounds per level in the recursion to a single one. Hence, it prevents the combinatorial explosion when combining any two adjacent recursion levels' results.

5.3 The Exhaustive yet Efficient Decomposition Algorithm

Our efficient algorithm depicted in Algorithm 1 achieves very accurate compositional algDNC delay bounds in feed-forward networks. It computes a flow of interest's alternative end-to-end left-over service curves using the following steps: First, $\mathrm{getB}_{\mathcal{T}}^{\mathrm{l.o.}\mathbb{F}}$ is invoked with the foi's path $\mathcal{P}$, i.e., $\mathcal{T} := \mathcal{P}$, and the foi itself. $\mathcal{P}$ constitutes the first tandem to decompose into 2^{n-1} disjoint sub-tandem decompositions $\{\mathbb{T}_1, \ldots, \mathbb{T}_{2^{n-1}}\}$, where n is the number of servers on $\mathcal{P}$. For each sub-tandem T of such a decomposition, we next backtrack (forcibly segregated, Section 4.2.1) cross-traffic to bound its arrivals. Our efficiency improvements can be found in this part of the algorithm: retrieval of a cached bound (line 20), convolution of alternative bounds (line 35) and caching of a bound (line 42). After the arrival bounding recursion has terminated, its results are used to derive the foi's left-over service curve for composition alternative $\mathbb{T}$, $\beta_{\mathbb{T}}^{\mathrm{l.o.}\mathbb{F}}$. The efficiency of this step could be increased by caching a sub-tandem $\beta_T^{\mathrm{l.o.}}$ which we leave for future work. In a final step (not shown in Algorithm 1) the delay bounds for all decomposition alternatives are computed. The minimum among them is the best algDNC delay bound attainable with our exhaustive decomposition design. In Appendix C we provide more details about our algDNC algorithm's complexity and compare it to the SFA from the literature.

6 EVALUATION

In this numerical evaluation, we benchmark our new algDNC analysis algorithm against the optDNC's ULP as well as the SFA. It is known that the ULP will derive the most accurate delay bounds among the alternatives and that SFA will derive the least accurate ones. We show that our novel algDNC analysis considerably outperforms the SFA. It even derives delay bounds close to the ULP, yet, in a fraction of the ULP's computation time.

6.1 AFDX Topology Case Study

First, we investigate delay bound accuracy in a network topology as found in the industrial avionics context. The topology we exemplarily analyze is dimensioned similarly to the backbone network in the Airbus A380. It has a

---

ALGORITHM 1: Exhaustive Decomposition Algorithm: Efficient Derivation of all Left-over Service Curves of $\mathcal{T}$.

1 $\text{get}\mathbb{B}_{\mathcal{T}}^{\text{l.o.}\mathbb{F}}$(Tandem $\mathcal{T}$, Flow aggregate of interest $\mathbb{F}$)

2      $\{\mathbb{T}_1, \ldots, \mathbb{T}_{2^{n-1}}\} = \text{getDecompositions}(\mathcal{T});$ /* Disjoint subtandem decompositions of $\mathcal{T}$ */

3   **foreach** *Decomposition* $\mathbb{T} \in \{\mathbb{T}_1, \ldots, \mathbb{T}_{2^{n-1}}\}$ **do**

4   **foreach** *Subtandem* $T \in \mathbb{T}$ **do**

5              $\left(\mathbb{F}_i, \ldots, \mathbb{F}_{i+j}\right) = \text{xtxSegregation}\,(T, \mathbb{F});$ /* Enforced segregations (Section 4.2.1) */

6              $(\alpha_T^{\mathbb{F}_i}, \ldots, \alpha_T^{\mathbb{F}_{i+j}}) = \text{AB}\,(T, \left(\mathbb{F}_i, \ldots, \mathbb{F}_{i+j}\right));$ /* Arrival bounding and left-over service*/

7              $\beta_T^{\text{l.o.}\mathbb{F}} = \beta_T \ominus (\alpha_T^{\mathbb{F}_i}, \ldots, \alpha_T^{\mathbb{F}_{i+j}});$

8              $\beta_{\mathbb{T}}^{\text{l.o.}\mathbb{F}} \otimes= \beta_T^{\text{l.o.}\mathbb{F}};$ /* End-to-end service of decomposition $\mathbb{T}$ */

9   **end**

10          $\mathbb{B}_{\mathcal{T}}^{\text{l.o.}\mathbb{F}}.\text{put}\,(\beta_{\mathbb{T}}^{\text{l.o.}\mathbb{F}});$

11   **end**

12 **return** $\mathbb{B}_{\mathcal{T}}^{\text{l.o.}\mathbb{F}}$

13 $\text{AB}\,$(Tandem T, Flow aggregates to bound $\left(\mathbb{F}_i, \ldots, \mathbb{F}_{i+j}\right)$)

14   **foreach** *Flow aggregate* $\mathbb{F} \in \left(\mathbb{F}_i, \ldots, \mathbb{F}_{i+j}\right)$ **do**

15   **try**{

16              $\alpha_T^{\mathbb{F}} = \text{getCacheEntry}\,(T, \mathbb{F});$ /* Efficiency: Check for cached $\alpha_T^{\mathbb{F}}$ */

17              $\mathbb{A}_T^{\mathbb{F}}.\text{put}\left(\alpha_T^{\mathbb{F}}\right);$

18   **continue;**

19   }

20   /* Start to recursively backtrack flows in the aggregate $\mathbb{F}$: */

21          $\mathcal{T}_{\text{shared}} = \text{backtrackTandem}\,(\mathbb{F}, T);$ /* Get $\mathbb{F}$'s tandem before reaching T */

22          $\mathbb{B}_{\mathcal{T}_{\text{shared}}}^{\text{l.o.}\mathbb{F}} = \text{get}\mathbb{B}_{\mathcal{T}}^{\text{l.o.}\mathbb{F}}\,(\mathcal{T}_{\text{shared}}, \mathbb{F});$ /* Compute and store left-over service curves */

23   /* Get arrivals of flows in $\mathbb{F}$ at $\mathcal{T}_{\text{shared}}$'s source */

24          $src = \mathcal{T}_{\text{shared}}.\text{getSrc}();$ /* The common source src has l inlinks */

25          $(\mathbb{F}_1, \ldots, \mathbb{F}_l) = \text{splitAsPerInlink}(src, \mathbb{F});$

26          $\alpha_{\mathcal{T}_{\text{shared}}}^{\mathbb{F}} = \sum \text{AB}\,(\langle src \rangle, (\mathbb{F}_1, \ldots, \mathbb{F}_l));$ /* Sum up the arrivals of flows in $\mathbb{F}$ at src */

27   /* Arrivals at T are the input at src, $\alpha_{\mathcal{T}_{\text{shared}}}^{\mathbb{F}}$, put out from $\mathcal{T}_{\text{shared}}$ with left-over service curves $\mathbb{B}_{\mathcal{T}_{\text{shared}}}^{\text{l.o.}\mathbb{F}}$ */

28   **foreach** $\beta_{\mathcal{T}_{\text{shared}}}^{\text{l.o.}\mathbb{F}} \in \mathbb{B}_{\mathcal{T}_{\text{shared}}}^{\text{l.o.}\mathbb{F}}$ **do**

29              $\alpha_T^{\mathbb{F}} \otimes= (\alpha_{\mathcal{T}_{\text{shared}}}^{\mathbb{F}} \oslash \beta_{\mathcal{T}_{\text{shared}}}^{\text{l.o.}\mathbb{F}});$ /* Efficiency: Convolve alternative $\alpha_T^{\mathbb{F}}$s */

30   **end**

31   /* Burst cap [8] */

32          $B^{\max} = \text{getBurstCap}\,(\mathcal{T}_{\text{shared}}.\text{getSink}\,());$

33          $\alpha_T^{\mathbb{F}}.\text{cap}\,(B^{\max});$

34          $\mathbb{A}_T^{\mathbb{F}}.\text{put}\,(\alpha_T^{\mathbb{F}});$

35          $\text{addCacheEntry}\left(\alpha_T^{\mathbb{F}}\right);$ /* Efficiency: Add a cache entry $\alpha_T^{\mathbb{F}}$ */

36   **end**

37 **return** $\mathbb{A}_T^{\mathbb{F}} = (\alpha_T^{\mathbb{F}_i}, \ldots, \alpha_T^{\mathbb{F}_{i+j}})$

(a) AFDX Case Study: Delay Bounds.

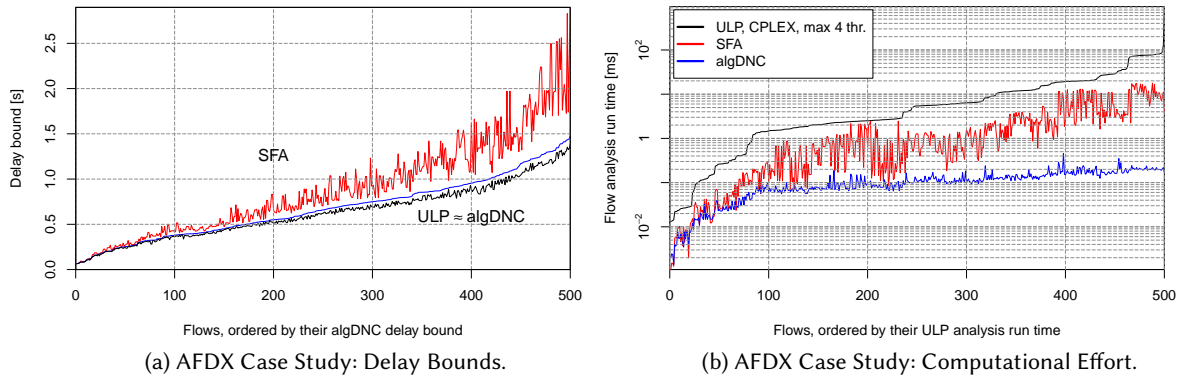(b) AFDX Case Study: Computational Effort.

Fig. 9. Delay bounds and computational effort in the AFDX topology with randomly routed unit size flows.

dense core of 16 switches that connect a total of 125 end-systems located in the network periphery. Each server has a service curve resembling a 100Mbps Ethernet link. We created a representative AFDX topology according to the algorithm presented in [19]. This topology generation scheme has some random factors in it, i.e., from an industrial point of view, the network we analyze may correspond to a single alternative in the pre-deployment design space exploration.

According to the current AFDX specification, flows are routed within so-called virtual links (VLs). Each VL connects a single source end-system to multiple sink end-systems (in the device graph) with fixed resource reservation on the path between these systems. In the view of the network calculus, VLs correspond to multicast flows that reserve large resource shares. An examination of the problems due to VLs' coarse granularity can be found in [37]. The analysis of multicast flows with algDNC was presented in [3], yet, a multicast ULP has not been provided. For an expressive comparison, we thus focus on the AFDX network topology itself and 500 randomly routed unicast flows to the network. They are shaped to unit size token buckets wit rate 1Mbps and bucket size 1Mb.

Figure 9a depicts the 500 individual flow delay bounds. The SFA's delay bounds show a gap to algDNC and ULP bounds that tends to grow on average. Additionally, the SFA delay bounds keep oscillating compared to the ULP and algDNC bounds, such that this analysis is not suitable to confidently rank AFDX design alternatives regarding their performance.

Figure 9b shows the time to analyze each flow in the network. This per-flow effort can differ more than three orders of magnitude within the SFA and the ULP analysis. This is surprising as the AFDX has a small network diameter and therefore the recursive backtracking is not very deep. For the very same reason, algDNC's effort stays within a much smaller range. In absolute terms, the algDNC outperforms ULP as well as SFA run times by multiple orders of magnitude – a decisive advantage in design space explorations.

6.2 Scalability of DNC Analyses

Next, we turn to the scalability. To do so, we created a set of larger test networks (see Appendix A.2) to evaluate quality and cost.

6.2.1 Quality: Accuracy of Delay Bounds. Figures 10a and 10b show each flow's end-to-end delay bound in two sample networks of very different size and complexity: the smallest in our set of networks (20 devices, 38 servers, 158 flows) and the largest one still feasible to analyze with ULP (180 devices, 646 servers, 2584 flows). For
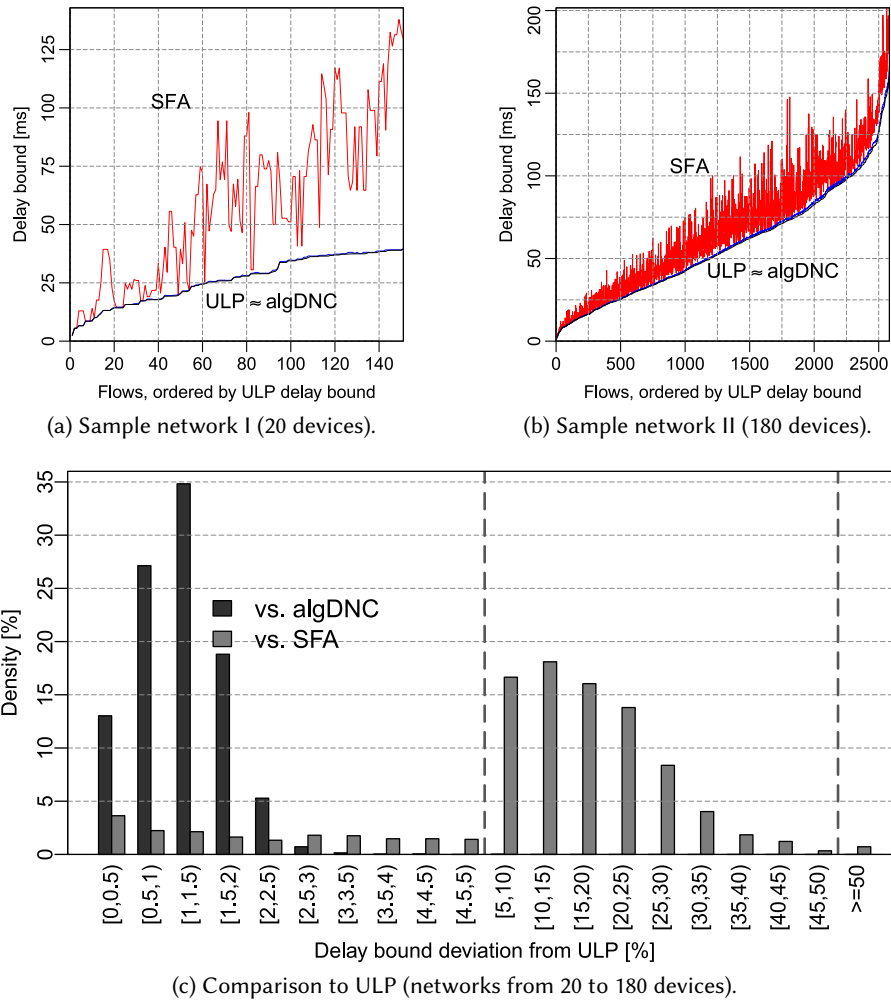
(a) Sample network I (20 devices).

(b) Sample network II (180 devices).

(c) Comparison to ULP (networks from 20 to 180 devices).

Fig. 10. Delay Bound Accuracy. Subfigures 10a and 10b show the delay bounds for two sample networks of different sizes. They depict each flow's end-to-end delay bound computed with SFA, ULP and our new exhaustive algDNC. While the old SFA cannot compete with the ULP, the results of our new algDNC analysis are extremely close to the ULP delay bounds.

Subfigure 10c provides quality statistics over all the networks from 20 to 180 devices (12376 flows, Table 2, left, in Appendix A.2) by depicting the deviation of delay bounds relative to the ULP's results. The accuracy of our new algDNC analysis is stable across all network sizes and, except for a single outlier at 7.57%, all new algDNC bounds deviate from ULP by at most 4.2%.

the small network, we already showed that the SFA results oscillate wildly with a large amplitude above the ULP (Section 3.2). Figure 10a extends this evaluation with the algDNC bounds and reveals that they increase in lockstep with the ULP while staying in close range. The same holds true in the larger sample network of Figure 10b. Both observations are also confirmed by the overall results of our experimental investigation. Figure 10c depicts the deviation of the SFA and the algDNC delay bounds from the ULP as observed in all network sizes from 20 to 180

with a total of 12376 observation points (Table 2, flows on the left, in Appendix A.2). The numerical evaluation confirms that optDNC's ULP barely outperforms the algDNC:

- Our new exhaustive decomposition analysis deviates from the ULP by only 1.142% on average in our evaluations.
- The 99th percentile is as low as 2.48% deviation.
- Results do not deviate by more than 4.2%, except for a single outlier at 7.57%. Yet, this outlier is still more accurate than 73.2% of the SFA-derived delay bounds.
- These accuracy characteristics are stable across different network sizes.

These results confirm that the problems of algDNC found in Section 4.2 were indeed crucial for its previous lack in accuracy. Moreover, our approach to find the (min,+)-algebraic equation with the minimal combined impact of all algDNC problems turned out to allow for delay bounds whose quality is competitive with optDNC's feasible heuristic ULP.

6.2.2 Cost: Network Analysis Times. Last, we evaluate the cost of attaining delay bounds in feed-forward networks. We show that our efficiency improvements have a crucial impact on the network analysis times of the algDNC analysis. Figure 11 shows that both outperform the ULP considerably and scale better with increasing network size. Our novel algDNC analysis performs best, a discernible increase in effort can only be observed in network sizes that are impractical to analyze with any of the other analyses. Although the experimental results are subject to fluctuations due to the high degree of randomness in the creation of our set of sample networks (see Appendix A.2), we can draw clear conclusions about fundamental trends in DNC network analysis:

- The ULP becomes computationally infeasible fast; the 180 devices network requires ∼ 13 days to be analyzed.
- The SFA scales better than the ULP. However, absolute effort increases to levels unsuitable for a design space exploration. Increasing the network size by only 20 devices can cause a huge difference of analysis effort. In our sample, analyzing the 260 devices network is relatively fast while the 280 devices network is already impractical to analyze. The SFA's effort seems barely predictable.
- Our novel algDNC scales better than the other analyses. It is also more resilient to the randomness of our network creation. Provoking a considerable increase of network analysis time required to vastly increase the network size to 1000 devices (3626 servers, 14504 flows).

Figure 11 also depicts results for different levels of deterministically parallelized optimization with CPLEX[2]. A maximum amount of 4 threads yields larger network analysis times than single-threaded execution due to the overhead of thread synchronization. However, for networks > 120 devices, an optimization with up to 8 parallel threads becomes faster. In Section 3.2, Figure 3b, we saw that CPLEX optimization consumes the vast majority of analysis time. In an attempt to improve optDNC analyses run time, we also investigated the potential benefit of further parallelization of this optimization step with CPLEX. Yet, run times were not affected significantly as the results in Appendix D show. Deriving accurate delay bounds in large feed-forward networks is solely possible with the novel algDNC analysis. OptDNC's trends w.r.t. delay bounds of individual flows are retained such that algDNC possesses a similar power in ranking of design alternatives.

7 RELATED WORK

In this work, we rely on the PMOO left-over service curve under arbitrary multiplexing of flows. DNC also offers an algebraically derived left-over service curve for FIFO-multiplexing servers $\beta_\theta^{\text{l.o.}}$ [34]. Similar to the arbitrary-multiplexing one, it is only applicable to a single server. Thus, it allows for an analysis akin to the SFA.

---

[2]Note, that among the tools employed in this evaluation, only CPLEX offers parallelization.
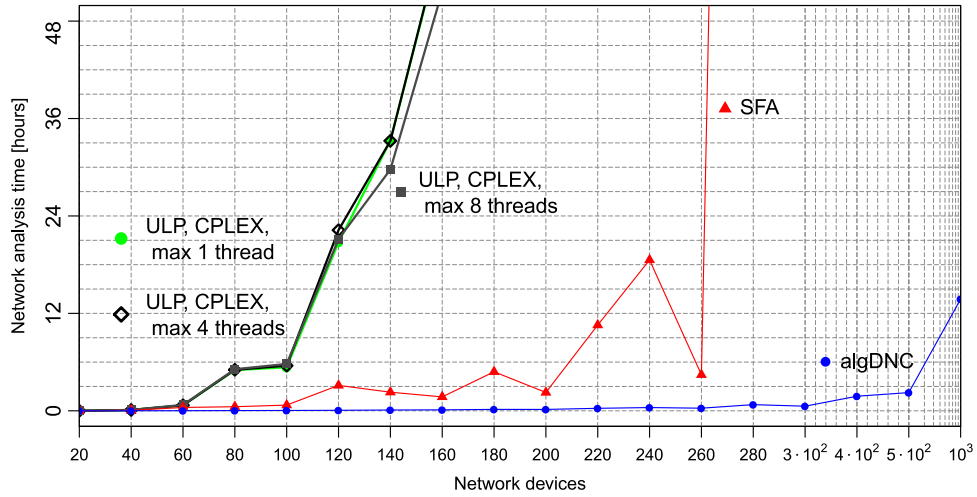
Net	ULP	SFA	algDNC
20	0:00:12	0:00:07	0:00:13
40	0:05:49	0:00:30	0:00:16
60	0:39:44	0:01:49	0:00:48
80	5:00:33	0:03:01	0:01:34
100	5:22:02	0:03:39	0:02:31
120	20:45:44	0:10:19	0:03:41
140	33:15:36	0:11:18	0:05:58
160	58:06:08	0:10:55	0:07:05
180	~13 days	0:29:52	0:10:27
200	–	0:12:15	0:10:13
220	–	0:51:16	0:18:23
	⋯		
400	–	128:27:16	1:47:39
	⋯		
10^3	–	–	13:45:52

Network analysis time (hh:mm:ss)
of single threaded analysis runs.

Fig. 11. Computational Effort. The ULP becomes computationally infeasible at moderate network sizes, its analysis time increases to ~13 days at 180 devices. The old SFA (without improvements) scales better, yet, its cost becomes unpredictable when increasing the network size. Moreover, it reaches >5 days at 400 devices. Our new algDNC scales better and is more resilient to the randomness of our network creation. Its underlying improvements can benefit the inaccurate SFA similarly.

Again, this causes the different problems, both on tandems where multiplexing with cross-flows is paid for more than once (Section 4.1) and in the compositional feed-forward network analysis (Section 4.2).

Effort to achieve the PMOO principle in the analysis of FIFO-multiplexing servers resulted in the Least Upper Delay Bound (LUDB) analysis [2, 35]. If paths of cross-flows do not overlap (i.e., the flows are nested into each other), the LUDB suggests to convolve servers before removing cross-flows, subject to the nesting. The latter step

is done by computing the above FIFO left-over service curve. E.g., in Figure 4a, xf_1's arrival at server s_1 ($\alpha_{s_1}^{xf_1}$) is removed from s_1, then the left-over service curve $\beta_{\theta,s_1}^{l.o.}$ is convolved with β_{s_2}, and finally $\alpha_{s_2}^{xf_1}$ is removed from this curve. Note, that this approach for tandem analysis enforces a PSOO violation during cross-traffic arrival bounding like any other algDNC tandem analysis. If paths of cross-flows are not nested, this approach cannot be applied, for instance, if we analyzed xf_2 in Figure 4a. In this case, [2] suggests to decompose into sub-tandems with nested interference patterns before applying the LUDB procedure. The end-to-end left-over service curve for delay bounding is then derived by convolution of the sub-tandem ones. Oftentimes, there are multiple alternatives to decompose to nested interference patterns (Figure 4a: both links are potential demarcations of a decomposition). LUDB suggests exhaustive enumeration of alternatives, computation of all delay bounds and returning the least one among them – the compFFA procedure depicted in Section 4.2 is strictly executed. Obvious drawbacks of this procedure are a lack of the PMOO principle implementation and a combinatorial explosion as shown in Section 4.3.

The LUDB's sub-tandem decomposition and our novel algDNC's decomposition are similar, yet, they differ in some key aspects. First, our algDNC does not require to result in tandems with nested interference as the arbitrary-multiplexing PMOO analysis fully implements the eponymous principle. Our exhaustive approach thus results in more decompositions per tandem than the LUDB. Therefore, more permissible (min,+)-equations are derived and solved. Secondly, [35] and [2] are concerned with a tandem analysis only. They do not address the problems in feed-forward analyses presented in Section 4.2, although the LUDB enforces PSOO violations. Moreover, they do not provide a technical solution for the potential combinatorial explosion problem; [2] rather presents a heuristic to trade accuracy against computational effort on a single tandem. Our novel analysis design is very generic and thus not restricted to arbitrary multiplexing. The LUDB tandem analysis can be embedded into it and benefit from our efficiency improvements.

An optimization-based DNC approach for tight FIFO-multiplexing feed-forward network analysis exists as well [16]. It transforms the DNC description of the network to a Mixed-Integer Linear Program (MILP) where the integer variables encode the (partially) parallel paths of flows. This circumvents the step of explicitly extending a partial order to the set of all compatible total orders, the root cause the LP analysis' combinatorial explosion. Again, the computational effort to solve the MILP for large networks is not evaluated. Instead, the authors advise to remove constraints such that all integer variables are removed, leaving an ordinary linear program to solve. I.e., tightness is traded for computational effort reduction. This is similar to the ULP heuristic that we showed to be computationally infeasible nonetheless.

8 CONCLUSION

In this article, we contribute the first DNC analysis for high quality and low cost end-to-end delay bounds in large feed-forward networks. We demonstrate this contribution on the novel set of feed-forward networks we created for DNC evaluation as well as the first comprehensive optDNC evaluation in such networks. Figure 12 summarizes the findings of our article: Against previous belief, we showed that optDNC's most efficient heuristic, ULP, is computationally infeasible even for moderately sized networks. For larger networks, we also showed that the algebraic SFA is more costly than expected, becoming barely feasible to execute. The remaining, grayed-out TFA and PMOO were not categorized more precisely – their quality does not change and more detailed knowledge about their cost would still have left the desired area in the intersection of low cost and high quality empty. Therefore, we developed a novel algDNC analysis that combines the strengths of all previously existing analyses, crucially improves their quality and incorporates decisive efficiency improvements. This novel algDNC analysis is currently the only network analysis providing highly accurate delay bounds while being computationally

feasible even in large feed-forward networks. Its algebraically derived delay bounds deviate from the optimization-based ULP analysis by only 1.142% on average in our evaluations while computation times are several orders of magnitude smaller.
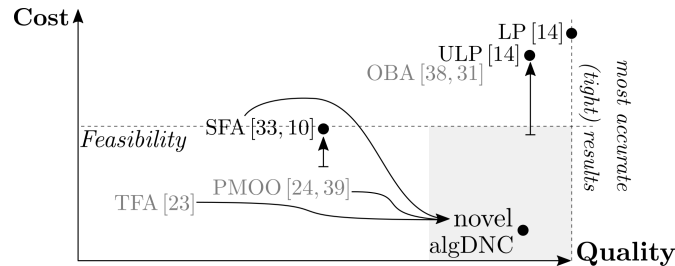

Fig. 12. Categorization of DNC analyses regarding their quality and cost. For analyses with black dots we generated sufficient evaluation data to pinpoint them with high certainty. Our novel algDNC analysis is evidently the only network analysis of good quality that is also feasible to execute.

REFERENCES

[1] Alessandro Biondi, Giorgio Buttazzo, and Stefano Simoncelli. 2015. Feasibility Analysis of Engine Control Tasks under EDF Scheduling. In *Proceedings of the 27th Euromicro Conference on Real-Time Systems (ECRTS)*. 139–148. DOI:https://doi.org/10.1109/ECRTS.2015.20

[2] Luca Bisti, Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. 2012. Numerical analysis of worst-case end-to-end delay bounds in FIFO tandem networks. *Real-Time Systems* 48, 5 (2012), 527–569. DOI:https://doi.org/10.1007/s11241-012-9153-1

[3] Steffen Bondorf and Fabien Geyer. 2016. Generalizing Network Calculus Analysis to Derive Performance Guarantees for Multicast Flows. In *Proceedings of the 10th International Conference on Performance Evaluation Methodologies and Tools (ValueTools)*.

[4] Steffen Bondorf and Jens B. Schmitt. 2010. Statistical Response Time Bounds in Randomly Deployed Wireless Sensor Networks. In *Proceedings of the 35th IEEE Local Computer Network Conference (LCN)*. 340–343. DOI:https://doi.org/10.1109/LCN.2010.5735738

[5] Steffen Bondorf and Jens B. Schmitt. 2014. The DiscoDNC v2 – A Comprehensive Tool for Deterministic Network Calculus. In *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools (ValueTools)*.

[6] Steffen Bondorf and Jens B. Schmitt. 2015. Boosting Sensor Network Calculus by Thoroughly Bounding Cross-Traffic. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*. 235–243. DOI:https://doi.org/10.1109/INFOCOM.2015.7218387

[7] Steffen Bondorf and Jens B. Schmitt. 2015. Calculating Accurate End-to-End Delay Bounds – You Better Know Your Cross-Traffic. In *Proceedings of the 9th International Conference on Performance Evaluation Methodologies and Tools (ValueTools)*.

[8] Steffen Bondorf and Jens B. Schmitt. 2016. Improving Cross-Traffic Bounds in Feed-Forward Networks – There is a Job for Everyone. In *Proceedings of GI/ITG International Conference on Measurement, Modelling and Evaluation of Dependable Computer and Communication Systems (MMB & DFT)*.

[9] Steffen Bondorf and Jens B. Schmitt. 2016. Should Network Calculus Relocate? An Assessment of Current Algebraic and Optimization-based Analyses. In *Proceedings of the 13th International Conference on Quantitative Evaluation of Systems (QEST)*.

[10] Anne Bouillard. 2014. Algorithms and efficiency of Network calculus. Habilitation thesis, École Normale Supérieure. (2014).

[11] Anne Bouillard, Bruno Gaujal, Sébastien Lagrange, and Eric Thierry. 2008. Optimal routing for end-to-end guarantees using Network Calculus. *Elsevier Performance Evaluation* 65, 11–12 (2008), 883–906.

[12] Anne Bouillard, Laurent Jouhet, and Eric Thierry. 2008. Computation of a (min,+) multi-dimensional convolution for end-to-end performance analysis. In *Proceedings of the 3rd International Conference on Performance Evaluation Methodologies and Tools (ValueTools)*. 68:1–68:7. http://dx.doi.org/10.4108/ICST.VALUETOOLS2008.4349

[13] Anne Bouillard, Laurent Jouhet, and Eric Thierry. 2009. *Service curves in Network Calculus: dos and don'ts*. Research Report RR-7094. INRIA. 24 pages.

[14] Anne Bouillard, Laurent Jouhet, and Eric Thierry. 2010. Tight performance bounds in the worst-case analysis of feed-forward networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*. 1–9.

[15] Anne Bouillard and Thomas Nowak. 2015. Fast symbolic computation of the worst-case delay in tandem networks and applications. *Performance Evaluation* 91 (2015), 270 – 285. DOI:https://doi.org/10.1016/j.peva.2015.06.016 Special Issue: Performance 2015.

[16] Anne Bouillard and Giovanni Stea. 2015. Exact Worst-Case Delay in FIFO-Multiplexing Feed-Forward Networks. *IEEE/ACM Transactions on Networking* 23, 5 (October 2015), 1387–1400. DOI:https://doi.org/10.1109/TNET.2014.2332071

[17] Anne Bouillard and Éric Thierry. 2008. An Algorithmic Toolbox for Network Calculus. *Discrete Event Dynamic Systems* 18, 1 (2008), 3–49. DOI:https://doi.org/10.1007/s10626-007-0028-x

[18] Marc Boyer and Christian Fraboul. 2008. Tightening end to end delay upper bound for AFDX network calculus with rate latency FIFO servers using network calculus. In *Proceedings of the IEEE International Workshop on Factory Communication Systems (WFCS)*. 11–20. DOI:https://doi.org/10.1109/WFCS.2008.4638728

[19] Marc Boyer, Nicolas Navet, and Marc Fumey. 2012. Experimental assessment of timing verification techniques for AFDX. In *Proceedings of the 6th European Congress on Embedded Real Time Software and Systems (ERTS)*.

[20] Tian Bu and Don Towsley. 2002. On Distinguishing between Internet Power Law Topology Generators. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*.

[21] Cheng-Shang Chang. 2000. *Performance Guarantees in Communication Networks.* Springer-Verlag, New York, NY.

[22] Rene L. Cruz. 1991. A Calculus for Network Delay, Part I: Network Elements in Isolation. *IEEE Transactions on Information Theory* 37, 1 (1991), 114–131.

[23] Rene L. Cruz. 1991. A Calculus for Network Delay, Part II: Network Analysis. *IEEE Transactions on Information Theory* 37, 1 (1991), 132–141.

[24] Markus Fidler. 2003. Extending the Network Calculus Pay Bursts Only Once Principle to Aggregate Scheduling. In *Proceedings of the 2nd International Workshop on Quality of Service in Multiservice IP Networks (QoS-IP)*. 19–34. http://dl.acm.org/citation.cfm?id=646464.691395

[25] Fabrice Frances, Christian Fraboul, and Jérôme Grieu. 2006. Using Network Calculus to optimize the AFDX network. In *Proceedings of the 6th European Congress on Embedded Real Time Software and Systems (ERTS)*.

[26] Fabien Geyer and Georg Carle. 2016. Network engineering for real-time networks: comparison of automotive and aeronautic industries approaches. *IEEE Communications Magazine* 54, 2 (February 2016), 106–112. DOI:https://doi.org/10.1109/MCOM.2016.7402269

[27] Jérôme Grieu. 2004. *Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques.* Ph.D. Dissertation. Institut National Polytechnique de Toulouse, France.

[28] Nan Guan and Wang Yi. 2013. Finitary Real-Time Calculus: Efficient Performance Analysis of Distributed Embedded Systems. In *Proceedings of the 34th IEEE Real-Time Systems Symposium (RTSS)*. 330–339.

[29] Keon Jang, Justine Sherry, Hitesh Ballani, and Toby Moncaster. 2015. Silo: Predictable Message Latency in the Cloud. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication (SIGCOMM)*. 435–448. DOI:https://doi.org/10.1145/2785956.2787479

[30] Bengt Jonsson, Simon Perathoner, Lothar Thiele, and Wang Yi. 2008. Cyclic dependencies in modular performance analysis. In *Proceedings of the 8th ACM international Conference on embedded software (EMSOFT)*. 179–188. DOI:https://doi.org/10.1145/1450058.1450083

[31] Mark Karol, Michael Hluchyj, and Samuel Morgan. 1987. Input Versus Output Queueing on a Space-Division Packet Switch. *IEEE Transactions on Communications* 35, 12 (1987), 1347–1356.

[32] Andreas Kiefer, Nicos Gollan, and Jens Schmitt. 2010. Searching for Tight Performance Bounds in Feed-Forward Networks. In *Proceedings of GI/ITG International Conference on Measurement, Modelling and Evaluation of Dependable Computer and Communication Systems (MMB & DFT)*.

[33] Kai Lampka, Steffen Bondorf, and Jens B. Schmitt. 2016. Achieving Efficiency without Sacrificing Model Accuracy: Network Calculus on Compact Domains. In *In Proceedings of the 24th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*. 313–318. DOI:https://doi.org/10.1109/MASCOTS.2016.9

[34] Jean-Yves Le Boudec and Patrick Thiran. 2001. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet.* Springer-Verlag, Berlin, Germany.

[35] Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. 2008. A Methodology for Computing End-to-End Delay Bounds in FIFO-Multiplexing Tandems. *Elsevier Performance Evaluation* 65, 11–12 (2008), 922–943.

[36] Xiaoting Li, Jean-Luc Scharbarg, and Christian Fraboul. 2010. Improving end-to-end delay upper bounds on an AFDX network by integrating offsets in worst-case analysis. In *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. 1–8.

[37] Renato Mancuso, Andrew V. Louis, and Marco Caccamo. 2015. Using Traffic Phase Shifting to Improve AFDX Link Utilization. In *Proceedings of the International Conference on Embedded Software (EMSOFT)*. 256–265. DOI:https://doi.org/10.1109/EMSOFT.2015.7318281

[38] Frank Ruskey. 2003. *Combinatorial Generation.* Working version of book in progress.

[39] Henrik Schiøler, Jan Jakob Jessen, Jens Dalsgaard Nielsen, and Kim Guldstrand Larsen. 2005. Network Calculus for Real Time Analysis of Embedded Systems with Cyclic Task Dependencies. In *Proceedings of the 20th Conference on Computers and Their Applications (CATA)*. 326–332.

[40] Jens B. Schmitt, Frank A. Zdarsky, and Markus Fidler. 2008. Delay Bounds under Arbitrary Multiplexing: When Network Calculus Leaves You in the Lurch .... In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*. 1669–1677.

[41] Jens B. Schmitt, Frank A. Zdarsky, and Ivan Martinovic. 2008. Improving Performance Bounds in Feed-Forward Networks by Paying Multiplexing Only Once. In *Proceedings of GI/ITG International Conference on Measurement, Modelling and Evaluation of Computer and Communication Systems (MMB)*. 1–15.

[42] David Starobinski, Mark Karpovsky, and Lev A. Zakrevski. 2003. Application of Network Calculus to General Topologies Using Turn-Prohibition. *IEEE/ACM Transactions on Networking* 11, 3 (2003), 411–421.

[43] Joanna Tomasik and Marc-Antoine Weisser. 2010. Internet topology on AS-level: Model, generation methods and tool. In *Proceedings of the IEEE International Performance Computing and Communications Conference (IPCCC)*. 263–270. DOI:https://doi.org/10.1109/PCCC. 2010.5682300

[44] Yaakov L. Varol and Doron Rotem. 1981. An algorithm to generate all topological sorting arrangements. *Comput. J.* 24, 1 (1981), 83–84.

[45] Timothy Zhu, Alexey Tumanov, Michael A. Kozuch, Mor Harchol-Balter, and Gregory R. Ganger. 2014. PriorityMeister: Tail Latency QoS for Shared Networked Storage. In *Proceedings of the ACM Symposium on Cloud Computing (SOCC)*. ACM, Article 29, 14 pages. DOI: https://doi.org/10.1145/2670979.2671008

A GENERATION OF NETWORKS FOR DNC SCALABILITY TESTS

A.1 Device Graph and Server Graph

Data communication networks are commonly modeled as graphs where nodes represent individual devices like a router or a switch. These devices can have multiple outputs to connect to other devices. Network calculus analyzes flows that cross the servers at the output of devices. Therefore, it needs to transform the device graph representation of a network into a server graph representation [4, 6] where these servers are directly connected. We use the term *network* to refer to a server graph crossed by flows.

A.2 Network Generation

Currently, there is neither standard set of feed-forward networks to test a DNC analysis on nor a common procedure to create them. Previous work provided small networks tailored to illustrate a specific advancement [9, 14, 40] and oftentimes restricted its tool support to these networks. In contrast, we created an extensible set of networks as follows:

We start with a device graph provided by a topology generator (aSHIIP [43]). We used the general linear preference (GLP) model [20] with its provided default parameter setting ($m_0 = 20$, $m = 1$, $p = 0.4695$, $\beta_{\text{GLP}} = 0.6447$) to create Internet-like topologies of different sizes. Then, we transformed them to their server graph representation where servers resemble the capability to transmit via 10Gbps links (service curves $\beta_{10\text{Gbps},0}$). Next, we applied the turn prohibition algorithm [42] to break potential cycles ("feed-forwardize the network") and added flows with a fixed server-to-flow ratio of 1:4 for all network sizes. Flows' arrivals were uniformly shaped to token buckets with rate 5Mbps and bucket size 5Mb, i.e., $\gamma_{5\text{Mbps},5\text{Mb}}$, to scale up from the unit-sized arrivals used in the AFDX evaluation. For path creation, a pair of source/sink devices were randomly selected from the device graph. The shortest path between these devices, yet, in the feed-forwardized server graph, was then computed. Table 2 show the resulting network sized we evaluate in Sections 9a and 9b.

Table 2. Networks to Evaluate Quality and Cost of DNC.

Devices	Servers	Flows	Devices	Servers	Flows
20	38	152	200	740	2960
40	118	472	220	744	2976
60	164	656	240	882	3528
80	282	1128	260	976	3904
100	364	1456	280	994	3976
120	398	1592	300	1124	4496
140	512	2048	400	1478	5912
160	572	2288	500	1876	7504
180	646	2584	1000	3626	14504

B (MIN,+)-EQUATIONS FOR FIGURES 4 AND 5

We show that the previously neglected decomposition alternative for an algDNC analysis, Alt3 shown in Figure 5, can arbitrarily outperform the existing analyses SFA and PMOO. First, we derive the respective left-over service curves, $\beta_{\langle s_1, s_2 \rangle}^{\mathrm{l.o.Alt3}}$, $\beta_{\langle s_1, s_2 \rangle}^{\mathrm{l.o.SFA}}$, and $\beta_{\langle s_1, s_2 \rangle}^{\mathrm{l.o.PMOO}}$, for the flow of interest (foi) in Figure 4a. They are crucial for each (min,+)-equation bounding the foi's delay.

$$
\begin{aligned}
\beta_{\langle s_1, s_2 \rangle}^{\mathrm{l.o.Alt3}} \;=\;& \beta_{\langle s_1, s_2 \rangle} \ominus \left(\alpha_{s_1}^{[xf_1, xf_2]}, \alpha_{s_2}^{xf_2} \right) \\
=\;& \beta_{s_1}^{\mathrm{l.o.foi}} \otimes \beta_{s_2}^{\mathrm{l.o.foi}} \\
=\;& \left(\beta_{s_1} \ominus \alpha_{s_1}^{[xf_1, xf_2]} \right) \otimes \left(\beta_{s_2} \ominus \alpha_{s_2}^{xf_2} \right) \\
=\;& \left(\beta_{s_1} \ominus \left(\alpha_{s_0}^{[xf_1, xf_2]} \oslash \beta_{s_0} \right) \right) \otimes \left(\beta_{s_2} \ominus \left(\alpha_{s_0}^{xf_1} \oslash \beta_{\langle s_0, s_1 \rangle}^{\mathrm{l.o.}xf_2} \right) \right) \\
=\;& \left(\beta_{s_1} \ominus \left(\alpha^{[xf_1, xf_2]} \oslash \beta_{s_0} \right) \right) \otimes \left(\beta_{s_2} \ominus \left(\alpha^{xf_1} \oslash \left(\beta_{\langle s_0, s_1 \rangle} \ominus \left(\alpha^{xf_2} \right) \right) \right) \right)
\end{aligned}
$$

$$
\begin{aligned}
\beta_{\langle s_1, s_2 \rangle}^{\mathrm{l.o.SFA}} \;=\;& \beta_{\langle s_1, s_2 \rangle} \ominus \left(\alpha_{s_1}^{[xf_1, xf_2]}, \alpha_{s_2}^{xf_2} \right) \\
=\;& \beta_{s_1}^{\mathrm{l.o.foi}} \otimes \beta_{s_2}^{\mathrm{l.o.foi}} \\
=\;& \left(\beta_{s_1} \ominus \alpha_{s_1}^{[xf_1, xf_2]} \right) \otimes \left(\beta_{s_2} \ominus \alpha_{s_2}^{xf_2} \right) \\
=\;& \left(\beta_{s_1} \ominus \left(\alpha_{s_0}^{[xf_1, xf_2]} \oslash \beta_{s_0}^{\mathrm{l.o.}[xf_1, xf_2]} \right) \right) \otimes \left(\beta_{s_2} \ominus \left(\alpha_{s_2}^{xf_2} \oslash \beta_{\langle s_0, s_1 \rangle}^{\mathrm{l.o.}xf_2} \right) \right) \\
=\;& \left(\beta_{s_1} \ominus \left(\left(\alpha_{s_0}^{xf_1} + \alpha_{s_0}^{xf_2} \right) \oslash \beta_{s_0} \right) \right) \otimes \left(\beta_{s_2} \ominus \left(\alpha^{xf_2} \oslash \left(\beta_{s_0}^{\mathrm{l.o.}xf_2} \otimes \beta_{s_1}^{\mathrm{l.o.}xf_2} \right) \right) \right) \\
=\;& \left(\beta_{s_1} \ominus \left(\left(\alpha^{xf_1} + \alpha^{xf_2} \right) \oslash \beta_{s_0} \right) \right) \otimes \left(\beta_{s_2} \ominus \left(\alpha^{xf_2} \oslash \left(\left(\beta_{s_0} \ominus \alpha_{s_0}^{xf_1} \right) \otimes \left(\beta_{s_1} \ominus \alpha_{s_1}^{xf_1} \right) \right) \right) \right) \\
=\;& \left(\beta_{s_1} \ominus \left(\left(\alpha^{xf_1} + \alpha^{xf_2} \right) \oslash \beta_{s_0} \right) \right) \otimes \left(\beta_{s_2} \ominus \left(\alpha^{xf_2} \oslash \left(\left(\beta_{s_0} \ominus \alpha^{xf_1} \right) \otimes \left(\beta_{s_1} \ominus \left(\alpha^{xf_1} \oslash \beta_{s_0}^{\mathrm{l.o.}xf_1} \right) \right) \right) \right) \right) \\
=\;& \left(\beta_{s_1} \ominus \left(\left(\alpha^{xf_1} + \alpha^{xf_2} \right) \oslash \beta_{s_0} \right) \right) \otimes \left(\beta_{s_2} \ominus \left(\alpha^{xf_2} \oslash \left(\left(\beta_{s_0} \ominus \alpha^{xf_1} \right) \otimes \left(\beta_{s_1} \ominus \left(\alpha^{xf_1} \oslash \left(\beta_{s_0} \ominus \alpha^{xf_2} \right) \right) \right) \right) \right) \right)
\end{aligned}
$$

$$
\begin{aligned}
\beta_{\langle s_1, s_2 \rangle}^{\mathrm{l.o.PMOO}} \;=\;& \beta_{\langle s_1, s_2 \rangle} \ominus \left(\alpha_{s_1}^{xf_1}, \alpha_{s_1}^{xf_2} \right) \\
=\;& \beta_{\langle s_1, s_2 \rangle} \ominus \left(\alpha^{xf_1} \oslash \beta_{s_0}^{\mathrm{l.o.}xf_1}, \alpha^{xf_2} \oslash \beta_{s_0}^{\mathrm{l.o.}xf_2} \right) \\
=\;& \beta_{\langle s_1, s_2 \rangle} \ominus \left(\alpha^{xf_1} \oslash \left(\beta_{s_0} \ominus \alpha^{xf_2} \right), \alpha^{xf_2} \oslash \left(\beta_{s_0} \ominus \alpha^{xf_1} \right) \right)
\end{aligned}
$$

Next, we construct a sample parameter setting for the curves in our three (min,+)-equations that simplifies the complex tandem left-over derivations $\beta_{\mathcal{T}}^{\mathrm{l.o.}}$. This parameter setting allows us to continue by arguing over the exact curve shapes.

Assume the following arrival curves from $\mathcal{F}_{\mathrm{TB}}$: $\alpha^{xf_1}(d) = r^{xf_1} \cdot d$, $\alpha^{xf_2}(d) = r^{xf_2} \cdot d$, and $\alpha^{\mathrm{foi}}(d) = r^{\mathrm{foi}} \cdot d$. Moreover assume the following strict service curves from $\mathcal{F}_{\mathrm{RL}}$: $\beta_{s_n}(d) = R_{s_n}$, $n \in \{1, 2\}$, $\beta_{s_0}(d) = R_{s_0} \cdot \max\left\{ 0, d - T_{s_0} \right\}$, with $R_{s_m} \geq r^{xf_1} + r^{xf_2} + r^{\mathrm{foi}}$, $m \in \{0, 1, 2\}$ for finite delay bounds. For positive burstiness increase of the cross-flows, assume $T_{s_0} > 0$, i.e., the burst terms (bucket sizes of $\mathcal{F}_{\mathrm{TB}}$) $b_{s_n}^{\mathrm{AB}\mathbb{F}}$ become positive for each arrival bounding $\mathrm{AB} \in \{\mathrm{SFA}, \mathrm{PMOO}, \mathrm{Alt3}\}$ of any cross-flow (aggregate) $\mathbb{F} \in \{xf_1, xf_2, [xf_1, xf_2]\}$, at servers s_1 and s_2.

In this simplified setting, the flow of interest's delay bound equals the $\beta_{\langle s_1, s_2 \rangle}^{\mathrm{l.o.}}$'s latency term $T_{\langle s_1, s_2 \rangle}^{\mathrm{l.o.}}$. Therefore, we derive the latency terms for all three decomposition. The SFA left-over latency, abbreviated $T_{\langle s_1, s_2 \rangle}^{\mathrm{l.o.SFA}}$, respectively

the SFA delay D^{SFA}, are

$$D^{\mathrm{SFA}} = T^{\mathrm{l.o.SFA}}_{\langle s_1, s_2 \rangle} = T^{\mathrm{l.o.SFA}}_{s_1} + T^{\mathrm{l.o.SFA}}_{s_2} = \frac{b^{\mathrm{SFA}xf_1}_{s_1} + b^{\mathrm{SFA}xf_2}_{s_1}}{R_{s_1} - r^{xf_1} - r^{xf_2}} + \frac{b^{\mathrm{SFA}xf_2}_{s_2}}{R_{s_2} - r^{xf_2}},$$

the PMOO left-over latency $T^{\mathrm{l.o.PMOO}}_{\langle s_1, s_2 \rangle}$ and delay D^{PMOO} are

$$T^{\mathrm{l.o.\,PMOO}}_{\langle s_1, s_2 \rangle} = D^{\mathrm{PMOO}} = \frac{b^{\mathrm{PMOO}xf_1}_{s_1} + b^{\mathrm{PMOO}xf_2}_{s_1}}{\left(R_{s_1} - r^{xf_1} - r^{xf_2}\right) \wedge \left(R_{s_2} - r^{xf_2}\right)},$$

and Alt3's left-over latency $T^{\mathrm{l.o.Alt3}}_{\langle s_1, s_2 \rangle}$ and delay D^{Alt3} are

$$D^{\mathrm{Alt3}} = T^{\mathrm{l.o.Alt3}} = T^{\mathrm{l.o.Alt3}}_{s_1} + T^{\mathrm{l.o.Alt3}}_{s_2} = \frac{b^{\mathrm{Alt3}[xf_1, xf_2]}_{s_1}}{R_{s_1} - r^{xf_1} - r^{xf_2}} + \frac{b^{\mathrm{Alt3}xf_2}_{s_2}}{R_{s_2} - r^{xf_2}}.$$

We see that $D^{\mathrm{PMOO}} \leq D^{\mathrm{SFA}}$, iff

$$\frac{b^{\mathrm{PMOO}xf_1}_{s_1} + b^{\mathrm{PMOO}xf_2}_{s_1}}{\left(R_{s_1} - r^{xf_1} - r^{xf_2}\right) \wedge \left(R_{s_2} - r^{xf_2}\right)} \quad \leq \quad \frac{b^{\mathrm{SFA}xf_1}_{s_1} + b^{\mathrm{SFA}xf_2}_{s_1}}{R_{s_1} - r^{xf_1} - r^{xf_2}} + \frac{b^{\mathrm{SFA}xf_2}_{s_2}}{R_{s_2} - r^{xf_2}}$$

1) $R_{s_1} - r^{xf_1} - r^{xf_2} > R_{s_2} - r^{xf_2}$:

$$\frac{b^{\mathrm{PMOO}xf_1}_{s_1} + b^{\mathrm{PMOO}xf_2}_{s_1}}{R_{s_2} - r^{xf_2}} \quad \leq \quad \frac{b^{\mathrm{SFA}xf_1}_{s_1} + b^{\mathrm{SFA}xf_2}_{s_1}}{R_{s_1} - r^{xf_1} - r^{xf_2}} + \frac{b^{\mathrm{SFA}xf_2}_{s_2}}{R_{s_2} - r^{xf_2}}$$

$$\frac{b^{\mathrm{PMOO}xf_1}_{s_1} + b^{\mathrm{PMOO}xf_2}_{s_1} - b^{\mathrm{SFA}xf_2}_{s_2}}{R_{s_2} - r^{xf_2}} \quad \leq \quad \frac{b^{\mathrm{SFA}xf_1}_{s_1} + b^{\mathrm{SFA}xf_2}_{s_1}}{R_{s_1} - r^{xf_1} - r^{xf_2}}$$

$$\frac{R_{s_1} - r^{xf_1} - r^{xf_2}}{R_{s_2} - r^{xf_2}} \quad \leq \quad \frac{b^{\mathrm{SFA}xf_1}_{s_1} + b^{\mathrm{SFA}xf_2}_{s_1}}{b^{\mathrm{PMOO}xf_1}_{s_1} + b^{\mathrm{PMOO}xf_2}_{s_1} - b^{\mathrm{SFA}xf_2}_{s_2}}$$

2) $R_{s_2} - r^{xf_2} \geq R_{s_1} - r^{xf_1} - r^{xf_2}$:

$$\frac{b^{\mathrm{PMOO}xf_1}_{s_1} + b^{\mathrm{PMOO}xf_2}_{s_1}}{R_{s_1} - r^{xf_1} - r^{xf_2}} \quad \leq \quad \frac{b^{\mathrm{SFA}xf_1}_{s_1} + b^{\mathrm{SFA}xf_2}_{s_1}}{R_{s_1} - r^{xf_1} - r^{xf_2}} + \frac{b^{\mathrm{SFA}xf_2}_{s_2}}{R_{s_2} - r^{xf_2}}$$

$$\frac{b^{\mathrm{PMOO}xf_1}_{s_1} + b^{\mathrm{PMOO}xf_2}_{s_1} - b^{\mathrm{SFA}xf_1}_{s_1} + b^{\mathrm{SFA}xf_2}_{s_1}}{R_{s_1} - r^{xf_1} - r^{xf_2}} \quad \leq \quad \frac{b^{\mathrm{SFA}xf_2}_{s_2}}{R_{s_2} - r^{xf_2}}$$

$$\frac{R_{s_2} - r^{xf_2}}{R_{s_1} - r^{xf_1} - r^{xf_2}} \quad \leq \quad \frac{b^{\mathrm{SFA}xf_2}_{s_2}}{b^{\mathrm{PMOO}xf_1}_{s_1} - b^{\mathrm{SFA}xf_1}_{s_1} + b^{\mathrm{PMOO}xf_2}_{s_1} - b^{\mathrm{SFA}xf_2}_{s_1}}$$

Neither condition is strictly fulfilled as both depend on the service rates at s_1 and s_2. In [40], it was shown that the PMOO *tandem* analysis cannot exploit fast residual service rates at the end; this derivation shows that the PMOO *network* analysis also struggles with fast rates at the front of the flow of interest's path due the enforced PSOO violation.

Last, we show that the previously neglected Alt3 can outperform the existing compositional feed-forward analyses SFA and PMOO. The relations between D^{Alt3}, D^{SFA} and D^{PMOO} are:

1) $D^{\text{Alt3}} \leq D^{\text{SFA}}$, iff

$$\frac{b_{s_1}^{\text{Alt3}[xf_1, xf_2]}}{R_{s_1} - r^{xf_1} - r^{xf_2}} + \frac{b_{s_2}^{\text{Alt3}xf_2}}{R_{s_2} - r^{xf_2}} \quad \leq \quad \frac{b_{s_1}^{\text{SFA}xf_1} + b_{s_2}^{\text{SFA}xf_2}}{R_{s_1} - r^{xf_1} - r^{xf_2}} + \frac{b_{s_2}^{\text{SFA}xf_2}}{R_{s_2} - r^{xf_2}}$$

$$\frac{b_{s_1}^{\text{Alt3}[xf_1, xf_2]} - b_{s_1}^{\text{SFA}xf_1} - b_{s_1}^{\text{SFA}xf_2}}{R_{s_1} - r^{xf_1} - r^{xf_2}} \quad \leq \quad \frac{b_{s_2}^{\text{SFA}xf_2} - b_{s_2}^{\text{Alt3}xf_2}}{R_{s_2} - r^{xf_2}}$$

$$\frac{R_{s_2} - r^{xf_2}}{R_{s_1} - r^{xf_1} - r^{xf_2}} \quad \leq \quad \frac{b_{s_2}^{\text{SFA}xf_2} - b_{s_2}^{\text{Alt3}xf_2}}{b_{s_1}^{\text{Alt3}[xf_1, xf_2]} - b_{s_1}^{\text{SFA}xf_1} - b_{s_1}^{\text{SFA}xf_2}}$$

2) $D^{\text{Alt3}} \leq D^{\text{PMOO}}$, iff

   a) $R_{s_1} - r^{xf_1} - r^{xf_2} > R_{s_2} - r^{xf_2}$:

$$\frac{b_{s_1}^{\text{Alt3}[xf_1, xf_2]}}{R_{s_1} - r^{xf_1} - r^{xf_2}} + \frac{b_{s_2}^{\text{Alt3}xf_2}}{R_{s_2} - r^{xf_2}} \quad \leq \quad \frac{b_{s_1}^{\text{PMOO}xf_1} + b_{s_1}^{\text{PMOO}xf_2}}{R_{s_2} - r^{xf_2}}$$

$$\frac{b_{s_1}^{\text{Alt3}[xf_1, xf_2]}}{R_{s_1} - r^{xf_1} - r^{xf_2}} \quad \leq \quad \frac{b_{s_1}^{\text{PMOO}xf_1} + b_{s_1}^{\text{PMOO}xf_2} - b_{s_2}^{\text{Alt3}xf_2}}{R_{s_2} - r^{xf_2}}$$

$$\frac{R_{s_2} - r^{xf_2}}{R_{s_1} - r^{xf_1} - r^{xf_2}} \quad \leq \quad \frac{b_{s_1}^{\text{PMOO}xf_1} + b_{s_1}^{\text{PMOO}xf_2} - b_{s_2}^{\text{Alt3}xf_2}}{b_{s_1}^{\text{Alt3}\{xf_1, xf_2\}}}$$

   b) $R_{s_2} - r^{xf_2} \geq R_{s_1} - r^{xf_1} - r^{xf_2}$:

$$\frac{b_{s_1}^{\text{Alt3}[xf_1, xf_2]}}{R_{s_1} - r^{xf_1} - r^{xf_2}} + \frac{b_{s_2}^{\text{Alt3}xf_2}}{R_{s_2} - r^{xf_2}} \quad \leq \quad \frac{b_{s_1}^{\text{PMOO}xf_1} + b_{s_1}^{\text{PMOO}xf_2}}{R_{s_1} - r^{xf_1} - r^{xf_2}}$$

$$\frac{b_{s_2}^{\text{Alt3}xf_2}}{R_{s_2} - r^{xf_2}} \quad \leq \quad \frac{b_{s_1}^{\text{PMOO}xf_1} + b_{s_1}^{\text{PMOO}xf_2} - b_{s_1}^{\text{Alt3}[xf_1, xf_2]}}{R_{s_1} - r^{xf_1} - r^{xf_2}}$$

$$\frac{R_{s_1} - r^{xf_1} - r^{xf_2}}{R_{s_2} - r^{xf_2}} \quad \leq \quad \frac{b_{s_1}^{\text{PMOO}xf_1} + b_{s_1}^{\text{PMOO}xf_2} - b_{s_1}^{\text{Alt3}[xf_1, xf_2]}}{b_{s_2}^{\text{Alt3}xf_2}}$$

The observations about the relation of delay bounds still hold: Relations 1), 2a), and 2b) reflect the influence of the rates on the flow of interest's path (left terms). A large service rate R_{s_1} can, in fact, best be exploited by Alt3. Alt3 can thus simultaneously outperform both existing analyses. However, there is no strict ordering with respect to delay bounds; each analysis can potentially outperform the others.

Considering the parameters omitted for the ease of presentation, we can find a simple parameter setting that allows Alt3 to arbitrarily outperform SFA and PMOO. Alt3 scales better with increasing b^{xf_1} when parameters are set to: $\beta_{s_0} = \beta_{25,5}$, $\beta_{s_1} = \beta_{25,0}$, $\beta_{s_2} = \beta_{3,5}$, $\alpha^{\text{foi}} = \gamma_{0.5,5}$, $\alpha^{xf_1} = \gamma_{2.5, b^{xf_1}}$, $\alpha^{xf_2} = \gamma_{2.5,5}$. This is illustrated by the following Figure.
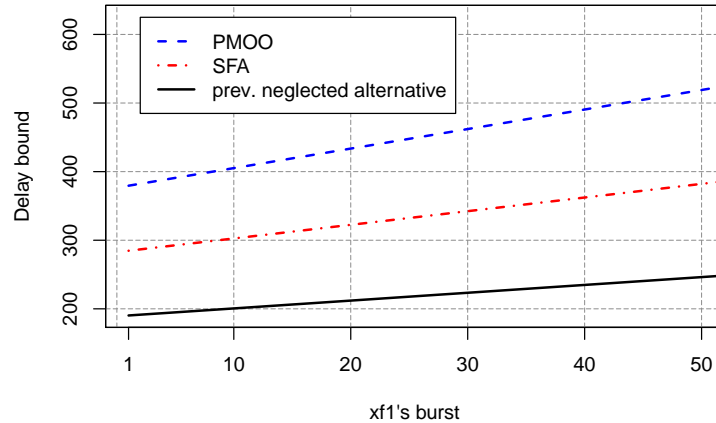
Fig. 13. The previously neglected alternative (see Section 4.2.2) can arbitrarily outperform PMOO and SFA in the network shown in of Figure 4a.

C COMPUTATIONAL COMPLEXITY

The computational complexity of algDNC algorithms mainly depends on two aspects:

(1) the shape of curves used in the model (Section 2.1) defines the complexity of the operations applied to them (Section 4.1).
(2) the network put into the algorithm, including the entanglement of flows crossing it, defines the operations in a permissible algDNC equation (Section 2.2).

For (1), complexity of basic operations, [17] gives results. Most notably, if arrival curves are from the set $\mathcal{F}_{\text{TB}}$ and service curves are from the set $\mathcal{F}_{\text{RL}}$, the (min,+)-algebraic DNC operations are in $O(1)$. The second impact factor, the analyzed network, has not been investigated in detail w.r.t. algDNC analyses presented in this article. We provide results about the amount of operations required for an analysis of tandems as well as sink trees.

C.1 Tandem of Length h

We start with an analysis of the two algebraic analyses presented in this article, SFA and algDNC, applied to a tandem of length h, crossed by m flows.

C.1.1 SFA. The presented SFA tandem analysis requires cross-traffic arrival bounds at every server on the tandem in order to compute the foi's left-over service curve (step 1 in Section 4.2). Then, the end-to-end left-over service curve can be computed in order to derive the foi's delay bound (step 2 in Section 4.2). For upper bounding the required amount of operations, we assume the foi crosses the maximum amount of servers, giving it a path of length h.

(1) Bounding the arrivals of all cross-flows at every server in the tandem.

The literature [10] proposes to segregately bound each cross-flow's arrival with a SFA. As the SFA only implements the PBOO principle, this procedure became known as segregated PBOO arrival bounding (segrPBOO) [7]. It starts at any server on the foi's path and recursively backtracks cross-flows to derive a bound on the arrival of each cross-flow there. We denote the distance to the analyzed tandem's last server with d, $d \in \{0, \ldots, h-1\}$.

- Assume a server crossed by the foi at distance d. At this server, there are $m - 1 < m$ cross-flows to be bounded, i.e., as many segrPBOO arrival bounding recursions are started.

- In this worst-case tandem, we can compute the number of recursions invoked for distance $d + i + 1$ by a server at distance $d + i$, $1 \leq i \leq h - d - 1$. A server will invoke one segrPBOO arrival bounding for each of its $m - 1 < m$ flows' $m - 2 < m$ cross-flows. Therefore, we can upper bound the amount of these segrPBOO invocations with m^2.

Next, we need to count the operations that are actually invoked at a distance $d + i$. Each invocation corresponds to the need to compute an output bound for a single flow. Thus, we need to aggregate this flow's $m - 1$ cross-flows with $m - 2$ operations, derive the flow's left-over service curve and compute the actual output bound. This results in m operations at each server.

For cross-traffic arrival bounding required by a server at distance d on the foi's path, we need to sum up the operations on the entire part of the tandem to be backtracked by segrPBOO arrival bounding, i.e., the servers at distances i, $d < i \leq h - 1$. We saw that these servers will be invoked multiple times, depending on the amount of server in distances j, $d < j \leq i - 1$, i.e., $\prod_{j=d+1}^{i-1} \left(m^2\right)^j$. The m operations for each flow at each server are scaled by this number. Last, the server crossed by the foi at distance d invokes m segrPBOO boundings itself. This gives the total number of operations:

$$m \cdot \sum_{i=d+1}^{h-1} m \cdot \prod_{j=d+1}^{i-1} \left(m^2\right)^j \quad = \quad m^{2-d(d+1)} \cdot \sum_{i=d+1}^{h-1} m^{(i-1)i}$$

Last, this procedure needs to be repeated for all servers on the foi's path. The total amount of operations to bound cross-traffic arrivals is thus upper bounded by

$$\sum_{d=0}^{h-1} m^{2-d(d+1)} \cdot \sum_{i=d+1}^{h-1} m^{(i-1)i} \tag{1}$$

(2) Executing a SFA for the flow of interest.

The second analysis step corresponds to the compFFA step 2, the tandem analysis on the foi's path. It consists of the basic algDNC operations, cross-flow aggregation, left-over service curve derivation, convolution to an end-to-end service curve, and the eventual bounding of the foi's delay. These operations have to be executed at each of the h servers crossed by the foi. Remember that there are m flows at each server, one of which is the foi.

Cross-flow Aggregation: After step 1, we know all $m - 1$ segregately derived cross-flow arrivals to be aggregated with $m - 2$ operations. I.e., there are $h \cdot (m - 2)$ aggregations in total.

Left-over Service Curve Derivation: The aggregated cross-traffic arrivals are used to compute the left-over service curve. There are h left-over operations in total.

End-to-end Service Curve Computation: Convolving the h left-over service curves requires $h - 1$ operations.
Delay Bounding: This is done with a single operation.

Total Amount of Operations for foi Analysis:

$$h \cdot (m - 2) + h + (h - 1) + 1 \quad = \quad h \cdot m. \tag{2}$$

Total Amount of Operations for the tandem SFA (Equations (1) + (2)):

$$h \cdot m \cdot \sum_{d=0}^{h-1} m^{2-d(d+1)} \cdot \sum_{i=d+1}^{h-1} m^{(i-1)i}$$

which defines the an upper bound on the complexity of the SFA in a tandem of length h, depending on the complexity imposed on the operations by the shape of curves chosen to model flow arrivals and service curves.

C.1.2 The novel, exhaustive algDNC analysis. Next, we evaluate the amount of operations required in our novel algDNC analysis. For comparison with the SFA above, we use the same tandem of h servers entirely crossed by m flows.

(1) Bounding the arrivals of all cross-flows at every server in the tandem.

In contrast to the SFA, we employ aggregation during arrival bounding to mitigate the most basic cause of PSOO violations. The corresponding procedure backtracks along paths of cross-flow aggregates. It is known as aggrAB. We will combine aggrAB with the exhaustive decomposition and the efficiency improvements of our novel algDNC analysis.

In a tandem of h servers that is entirely crossed by m flows, aggrAB only separates the foi and its single cross-traffic aggregate of $m - 1$ flows. I.e., the cross-traffic arrival bounding will not become recursive like the SFA's segrPBOO. The cross-traffic aggregate has a path of h servers, i.e, it contains $h - 1$ links which results in 2^{h-1} different decompositions into sub-tandem sequences. This leads to an average of $\frac{h+1}{2}$ tandems per decomposition, as Proposition 10 shows. Each tandem requires one output bound operation. Left-over service curve operations are not required as there is not recursive bounding of cross-traffic. However, tandems need to be convolved for the derivation of output bounds arriving further to the end of the tandem. That results in the need for $\frac{h+1}{2} - 1 = \frac{h-1}{2}$ convolution operations on average. In total, these steps give us

$$2^{h-1}\left(\frac{h+1}{2} + \frac{h-1}{2}\right) = h \cdot 2^{h-1} \tag{3}$$

algDNC operations to bound the foi's cross-traffic.

Proposition 10. *In a tandem of length h, a decomposition's average amount of sub-tandem is $\frac{h+1}{2}$.*

PROOF. There are $\binom{h-1}{k}$ possibilities to find $k - 1$ subtandems in a tandem of length h: only one $\left(\binom{h-1}{0}\right)$ configuration to have one subtandem, $h - 1 = \binom{h-1}{1}$ to have 2 subtandem etc. Since the total numbers of tandem matchings equals 2^{h-1}, we obtain for the average number of subtandems

$$\frac{\sum_{k=0}^{h-1}\binom{h-1}{k}(k+1)}{2^{h-1}} = \frac{\sum_{k=0}^{h-1}\binom{h-1}{k} + \sum_{k=0}^{h-1}k\binom{h-1}{k}}{2^{h-1}} = \frac{2^{h-1} + (h-1)\cdot 2^{h-2}}{2^{h-1}} = \frac{(h+1)\,2^{h-2}}{2^{h-1}} = \frac{h+1}{2}.$$

□

(2) Executing the novel algDNC analysis for the flow of interest.

On the foi's path, we proceed similarly. The tandem consists of h servers and can be decomposed into 2^{h-1} different sub-tandem sequences. In contrast to arrival bounding, we need to derive one left-over service curve per sub-tandem, yet, no output bound. This gives us $\frac{h+1}{2}$ as in aggrAB. Additionally, we convolve $\frac{h+1}{2} - 1 = \frac{h-1}{2}$ sub-tandems per decomposition on average, again. Finally, the foi analysis requires one delay bounding. Thus, the total amount of operations is

$$2^{h-1}\left(\frac{h+1}{2} + \frac{h-1}{2}\right) + 1 = h \cdot 2^{h-1} + 1. \tag{4}$$

Total Amount of Operations for the exhaustive tandem algDNC analysis (Equations (3) + (4)):

$$h \cdot 2^{h-1} + h \cdot 2^{h-1} + 1 = h \cdot 2^{h} + 1$$

which defines the complexity of the exhaustive algDNC in a tandem of h servers that is entirely crossed by m flows. Note, that the result is independent of the actual size of m as the aggrAB convolves all cross-flows into a single cross-traffic aggregate. This constitutes a major advantage over the segrPBOO arrival bounding of the literature's SFA [10].

C.2 Full k-ary Sink-Tree Networks of Height h

We continue with an analysis of SFA's and algDNC's application to full k-ary sink-tree networks of a maximum height h. These sink trees were already used to illustrate the combinatorial explosion of optDNC's LP analysis (Section 3.1).

Such a sink tree has $n = \frac{k^{h+1}-1}{k-1}$ nodes, each node corresponds to one server, and we assume the foi to originate at a leaf node in order to cross the maximum amount of nodes, $h + 1$ (the sink is at height 0).

C.2.1 SFA. For the SFA, the amount of operations in a full k-ary sink tree of height h is composed of the following parts:

(1) Bounding the arrivals of all cross-flows at every node in the tree.

The SFA applies segrPBOO arrival bounding. I.e., we need to recursively unfold the computations for each individual flow at every node in the sink tree. We use structural information of the sink tree to count the operations necessary to bound cross-traffic arrivals for the foi:

- The amount of nodes in the full k-ary sink tree of height h is $\frac{k^{h+1}-1}{k-1} < \frac{k}{k-1}k^h$. We assume that one flow originates at every node and all flows cross the sink. Then, there are $\frac{k}{k-1}k^h$ flows at the sink.
- Each node at distance $d \in \{0, \ldots, h\}$ from the sink is itself the root of a sub-tree. This sub-tree has $\frac{k^{h-d+1}-1}{k-1} < \frac{k}{k-1}k^{h-d}$ nodes, i.e., the it is crossed by $\frac{k}{k-1}k^{h-d}$ flows.
- There are k^d nodes at a distance of d from the sink.

The segrPBOO arrival bounding starts at any node on the foi's path and recursively derives a bound on the arrival of each cross-flow there.

- Assume a node crossed by the foi at distance d. There are at most $\frac{k}{k-1}k^{h-d} - 1 < \frac{k}{k-1}k^{h-d}$ cross-flows to be bounded, i.e., as many segrPBOO recursions are started. They result in $\frac{k}{k-1}k^{h-d}$ recursions invoking further recursions on the next level in the sink tree, i.e., distance $d + 1$ from the sink.
- In the sink tree, we can easily compute the number of times that the k^{d+i} nodes at distance $d+i$ will invoke operations at nodes at distance $d + i + 1$ from the sink, $1 \le i \le h - d$. The k^{d+i} nodes will each invoke one segrPBOO arrival bounding for each of their $\frac{k}{k-1}k^{h-(d+i)}$ flows' $\frac{k}{k-1}k^{h-(d+i)} - 1 < \frac{k}{k-1}k^{h-(d+i)}$ cross-flows. Note that, in the sink tree, it is guaranteed that all flows will need to bound their cross-traffic as the arrival bound at the next node closer to the sink is required. Also note, that we assume all cross-flows always arrive from a node one level further away from the source. Therefore, we can upper bound the amount of invocations at level $d + i + 1$ as we neglect that flows originate at distance $d + i$; the flows for which the arrival curve is already known. Thus, level $d + i$ will invoke $k^{d+i} \cdot \left(\frac{k}{k-1}k^{h-(d+i)}\right)^2$ segrPBOO arrival boundings at level $d + i + 1$.

Next, we need to count the operations that are actually invoked at a level $d + i$. For non-leaf nodes, these are $\frac{k}{k-1}k^{h-(d+i)} - 2$ aggregations of cross-flow arrival bounds, 1 left-over service curve derivation and 1 output bound. I.e., $\frac{k}{k-1}k^{h-(d+i)}$ operations in total. At leaf-nodes, we only have a single output bound operation each, as there in only a single flow present at this level.

For cross-traffic arrival bounding of a server at distance d on the foi's path, we need to sum up the operations in the entire sink tree backtracked by segrPBOO arrival bounding, i.e., the nodes at distances i, $d < i \le h$. We saw that these k^i nodes will be invoked depending on the size of the sink tree between d and i, i.e., distances j, $d < j < i - 1$, causing $\prod_{j=1}^{i-1} k^{d+j} \cdot \left(\frac{k}{k-1}k^{h-(d+j)}\right)^2$ invocations. These, in turn, cause $\frac{k}{k-1}k^{h-(d+i)}$ operations at non-leaf nodes and a single one at leaf nodes. Last, the node crossed by the foi at distance d invokes $\frac{k}{k-1}k^{h-d}$ segrPBOO boundings itself. This gives the total number of operations in segrPBOO arrival bounding for a node

at distance d from the sink:

$$\frac{k}{k-1}k^{h-d} \cdot \left(\sum_{i=d+1}^{h} \left(\frac{k}{k-1}k^{h-(d+i)} \cdot \prod_{j=1}^{i-1} k^{d+j} \left(\frac{k}{k-1}k^{h-(d+j)} \right)^2 \right) + \prod_{j=d}^{h+1} k^{h-j} \left(\frac{k}{k-1}k^{h-(d+j)} \right)^2 \right)$$

$$= \left(\frac{k}{k-1} \right)^3 k^{h-d} \cdot \left(\frac{k}{k-1}k^{3h-2d} \sum_{i=d+1}^{h} \left(k^{-\frac{1}{2}i(i-3)} \right) + k^{-\frac{3}{2}\left(h^2+6h+2-d^2-d\right)} \right)$$

$$\leq 8k^{4h-3d} \cdot \left(2 \sum_{i=d+1}^{h} \left(k^{-\frac{1}{2}i(i-3)} \right) + k^{\frac{3}{2}\left(-h^2-8h+d^2+3d-2\right)} \right),$$

where we have used that $k \geq 2$, the outdegree of a node defining the difference between tandem and tree networks.

Finally, this procedure needs to be repeated for all nodes on the foi's path. For upper bounding the required amount of operations, we assume the foi to cross the maximum amount of nodes, giving it a path of length $h + 1$, i.e., $d \in \{0, \ldots, h\}$. The total amount of operations to bound cross-traffic arrivals is thus upper bounded by

$$8k^{4h} \sum_{d=0}^{h} k^{-3d} \cdot \left(2 \sum_{i=d+1}^{h} \left(k^{-\frac{1}{2}i(i-3)} \right) + k^{\frac{3}{2}\left(-h^2-8h+d^2+3d-2\right)} \right).$$

(2) Executing a SFA for the flow of interest.

The second analysis step corresponds to the compFFA step 2. It equals the same step in the tandem, yet for $h + 1$ nodes.

Total Amount of Operations for the Sink-tree SFA

$$(h + 1) \cdot m \cdot 8k^{4h} \sum_{d=0}^{h} k^{-3d} \cdot \left(2 \sum_{i=d+1}^{h} \left(k^{-\frac{1}{2}i(i-3)} \right) + k^{\frac{3}{2}\left(-h^2-8h+d^2+3d-2\right)} \right).$$

which defines the complexity of the SFA in a full k-ary sink tree of maximum height h.

C.2.2 The novel, exhaustive algDNC analysis.

(1) Bounding the arrivals of all cross-flows at every node in the sink tree.

This analysis step covers compFFA step 1 presented in Section 4.2 and consists of the algDNC operations output bounding and aggregation of flows. For aggrAB with efficiency improvements, we can derive the amount of operations in this recursion as follows.

Output Bounding: Derivation of these bounds requires one (min,+)-deconvolution at all nodes except the sink whose output is not considered in any analysis. Neither do we consider the the foi's source node. Therefore, we require $n - 2$ deconvolution operations.

Aggregation: All non-leaf nodes have k inlinks, each contributing one flow (aggregate), all of which need to be aggregated during the arrival bounding in the sink tree. A full k-ary sink tree of height h has k^h leaf nodes and thus $n - k^h$ non-leaf nodes. Note, that the leaf nodes' share of the total network is increasing in k. The entire sink tree has $(n - k^h) \cdot k = n - 1$ links whose flows need to be aggregated. One exception exists: similar to output bounding, the link after the foi's source does not contribute to the aggregation requirement as it is only crossed by the foi. Therefore, we obtain a total of $n - 2$ algDNC aggregations operations.

Total Amount of Operations for Arrival Bounding:

$$n - 2 + n - 2 = 2n - 4. \tag{5}$$

(2) Executing the novel algDNC analysis for the flow of interest.

The foi's path is a tandem where we know all the required cross-traffic arrival bounds for our algDNC analysis. I.e., we take the same steps as shown in Section C.1.2's foi analysis. Note, however, that this tandem has an additional hop and consists of consisting of $h + 1$ nodes. Then, we obtain 2^h different decompositions into sub-tandem sequences with an average of $\frac{h+2}{2}$ sub-tandems. This results in

$$2^h \left(\frac{h+2}{2} + \frac{h}{2} \right) + 1 = 2^h (h+1) + 1 < 4n \cdot \log_2 (n) + 13 \tag{6}$$

algDNC operations for bounding the foi delay.

Total Amount of Operations for the exhaustive Sink-tree algDNC analysis (Equations (5) + (6)):

$$2n \cdot \left(1 + 2 \cdot \log_2 (n) \right) + 9,$$

which defines the complexity of the exhaustive algDNC in a full k-ary sink tree of maximum height h, depending on the complexity imposed on the operations by the shape of curves chosen to model flow arrivals and nodal service.

D   COST REDUCTION BY PARALLELIZED OPTIMIZATION

Figure 14 depicts the results of a second run of the same sample analyses of Section 9a and 9b. They were executed on a Colfax Ninja Workstation equipped with an Intel Xeon Phi 7210 CPU. In contrast to the Intel Xeon server CPU of the previous evaluation, the Xeon Phi is a modern many-core CPU offering 64 physical cores that can, employing simultaneous multithreading, process up to 256 threads in parallel. These cores do, however, clocked at only 1.3GHz. The workstation offers 110GB RAM to cope with the massively parallelized optimization's memory demand without swapping to disk.

The decreased single-thread performance negatively impacts the network analysis times of both algebraic DNC analyses. While only the increase in SFA analysis time is already visible in Figure 14, on the left, the measured algDNC times given in the table on the right show a significant increase as well. Relative to the server-grade CPU, SFA network analysis takes between 2.0 and 3.82 times as long, with an average of 3.27. For the algDNC analysis, the slowdown is between factor 4.05 and 4.51, with an average of 4.19. Similarly, the ULP analyses became slower. For both, the 4-threaded and the single-threaded analysis, this slowdown was between factor 1.16 and factor 3.33. Although this slowdown is smaller than algDNC's one, parallelized optimization did not allow optDNC to attain a similar performance in any network larger than 20 devices.

Most interestingly, this disadvantage of modern many-core CPUs could not be compensated with massively parallelizing the ULP's optimization. In fact, network analyses using all 256 logical cores were the slowest among the tested parallelization levels 1, 4, 16, 64, 128, and 256. Fastest analysis times could be achieved with a parallelization over a maximum of 16 threads, far below the amount of physical CPU cores (see Figure 14, inlets zooming into results for 120 and 140 devices). The difference between the best and worst setting for parallelization is $\approx 23\%$ in the smallest network but it decreases to constantly less than 8% in all larger networks. Thus, massively parallelized optimization did not yield considerable improvements for current optDNC analysis.
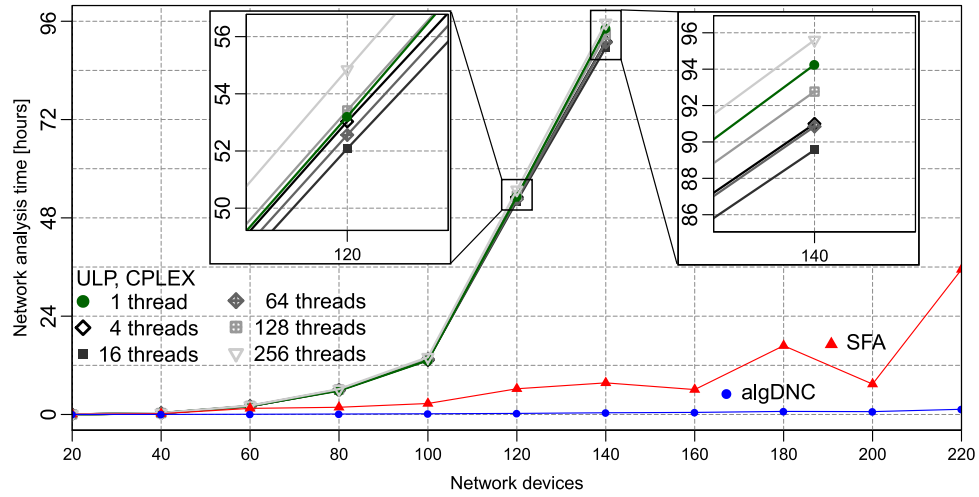
Net	ULP, CPLEX		algDNC
	(256 threads)	(16 threads)	(1 thread)
20	0:01:04	0:00:49	0:00:51
40	0:21:54	0:20:17	0:01:12
60	2:10:14	2:00:52	0:03:21
80	6:22:44	5:57:59	0:06:44
100	13:59:08	13:23:08	0:10:39
120	54:51:04	52:05:02	0:15:45
140	95:36:08	89:33:38	0:25:09
160	–	–	0:30:18
180	–	–	0:44:46
200	–	–	0:41:56
220	–	–	1:16:23
240	–	–	1:39:18

Network analysis time
(hh:mm:ss)

Fig. 14. Network analysis cost on a modern many-core architecture allowing for massively parallelized optimization over slower individual CPU cores. The slowdown for single-threaded execution increases algDNC network analysis times, yet, parallelized optimization does not allow optDNC to attain a similar performance in any network larger than 20 devices.