Exact Convex Formulations of Network-Oriented Optimal Operator Placement

Ben W. Carabelli, Andreas Benzing, Frank Dürr, Boris Koldehofe, Kurt Rothermel
Institute for Parallel and Distributed Systems
University of Stuttgart
70569 Stuttgart, Germany
Email: {firstname.lastname}@ipvs.uni-stuttgart.de

Georg Seyboth, Rainer Blind, Mathias Bürger, Frank Allgöwer
Institute for Systems Theory and Automatic Control
University of Stuttgart
70550 Stuttgart, Germany
Email: {firstname.lastname}@ist.uni-stuttgart.de

***Abstract*—Data processing tasks are increasingly spread across the internet to account for the spatially distributed nature of many data sources. In order to use network resources efficiently, subtasks need to be distributed in the network so data can be filtered close to the data sources. Previous approaches to this operator placement problem relied on various heuristics to constrain the complexity of the problem. In this paper, we propose two generic integer constrained problem formulations: a topology aware version which provides a placement including the specific network links as well as an end-to-end delay aware version which relies on the routing capabilities of the network. A linear programming relaxation for both versions is provided which allows exact and efficient solution using common solvers.**

I. Introduction

Many data-processing tasks like context reasoning [1], estimation using Kalman filters in wireless sensor networks [2] or data aggregation in global sensor networks [3], require the processing of data received at spatially distributed sources for the use at specific locations. Rather than gathering all data at a central location, it is favourable to process the data already in the underlying communication network. Such solutions can significantly reduce the network usage. In order to distribute the data-processing task, it is split into several subtasks. Local operators receive input data, perform their subtask and stream processed data further across the network to other operators. The operators are independent processing entities and can be executed on any node in the underlying network.

In many use cases, high-bandwidth data streams are required to connect operators and provide results in real-time. Therefore, the placement of operators has a severe impact not only on the performance of the networked application, but also on the use of resources of the underlying network [4]. Network usage can colloquially be described as the amount of data that is currently being transmitted. This quantity scales with the amount of data transmitted and the time it takes for that data to reach its destination. In technical terms, network usage on a particular link is therefore given by the *bandwidth-delay product* for that link, and the network usage induced by a stream processing task is given by the sum of the network usage on all links occupied by this task. In order to achieve optimal use of network resources, operators that are not fixed to a certain location have to be placed in the network such that the bandwidth-delay product is minimized.

With the emergence of internet-scale networked applications, the optimal operator placement problem has moved into the focus of research. A generic overview on placement strategies for internet-scale systems can be found in [4]. Two major directions have been pursued in the recent literature.

On the one hand, application oriented placement approaches focus on the execution cost in terms of CPU and memory resources. In this context, cost models of centralized approaches, originally developed for database applications, have been adapted to networked data-stream-based applications [5]–[7]. As the system is required to be fully aware of the operator behaviour, the workload is distributed among operators in a cost optimal fashion. However, the high bandwidth applications considered in the present paper require the optimization of network usage rather than processing cost.

This aspect is addressed by the second major research area. Network oriented approaches focus on the optimization of the network usage. Some fully distributed algorithms, which aim to minimize the network usage, were recently proposed [8]–[10]. These approaches require only statistical information about the data rates between operators but no information about their functionality. The distributed approaches are self-organizing, i.e., each operator periodically tries to find a better position in the network based on local knowledge. To make this problem computationally feasible, the underlying network is abstracted by a euclidean latency space, used to estimate the delay between two network nodes. Over time, the placement continuously improves towards a locally optimal placement. Since only the abstracted latency space is considered, instead of a graph model of the actual network, these approaches cannot be guaranteed to find a globally optimal solution. In fact, all approaches mentioned above either employ heuristics for generic placement frameworks or require very specific information about the operators used by the application. In contrast, we propose in this paper an exact, network oriented formulation of the optimal operator placement problem and show how it can be efficiently solved.

The contributions of this paper are as follows. We model the network oriented operator placement problem with two exact integer constrained optimization problems. We then show, that both problem formulations can be reformulated as convex linear programs without losing the integrality structure of the solution. The first problem formulation explicitly considers the topology of the underlying network. This problem formulation follows from a direct modelling of the problem as an interconnection of several shortest-path problems. We show that the convex relaxation of the integer linear program yields the exact integer solution.

The second problem formulation we consider ignores the topology of the underlying network, but requires that the end-to-end delays between the nodes of the underlay network are known. This assumption can be useful in order to leverage routing capabilities provided by the network. The operator placement problem is formulated as a bilinear integer program. We show that this bilinear problem can again be reformulated as a linear program with integer solutions, and the integer solutions are the exact solutions of the original problem. Both problem formulations can be solved efficiently for large-scale networks using standard optimization tools and, in contrast to the available heuristics, guarantee an optimal solution of the operator placement problem.

The remainder of this paper is structured as follows. In Sec. II, the optimal operator placement problem is presented in a general formulation. Then, in Sec. III, the operator placement problem is formulated as an integer constrained linear program. It is shown that the convex relaxation of the integer constraint leads to the exact solution. An alternative problem formulation as an integer constraint bilinear program, which does not require information about the topology of the underlay network, is presented in Sec. IV. It is shown that the integer bilinear program can be reformulated as a convex linear program. To show the practical applicability of our approach, a simulation study is provided in Sec. V. Finally, in Sec. VI concluding discussions are provided.

II. The Optimal Operator Placement Problem

We consider a network of physical processors, that are each capable of hosting multiple of the operators required for a stream processing task. The processors are all connected through a communication network, e.g., the internet. We refer to this network in the following as the *underlay network*, and model it by a directed graph $\mathcal{G}_u = (\mathcal{N}_u, \mathcal{E}_u)$. Each edge $i \in \mathcal{E}_u$ corresponds to a link in the underlay network used by a node in $\mathcal{N}_u$ to transmit data to another node. The time-delay affecting the data transmission over link i

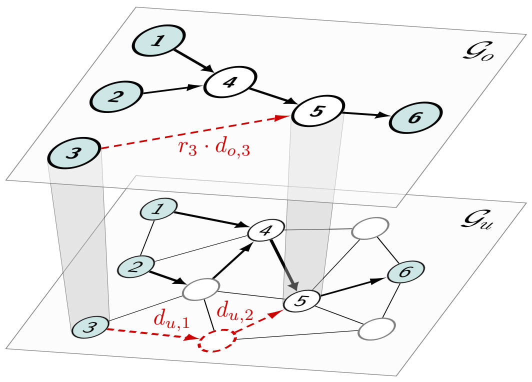

Figure 1. Example of an overlay graph $\mathcal{G}_o$ (top) and an underlay graph $\mathcal{G}_u$ (bottom). The overlay nodes 1, 2, 3 and 6 have fixed positions in the underlay graph, whereas 4 and 5 are freely placeable. The mapping of the overlay edge $(3, 5)$ to the underlay graph is highlighted (- - -). The cost induced by this edge would be given by $r_3 \cdot d_{o,3}$ with $d_{o,3} = d_{u,1} + d_{u,2}$ in this example.

is denoted with $d_{u,i} > 0$, $i = 1, \ldots, |\mathcal{E}_u|$. For simplicity, we assume throughout this paper that the time-delay over a communication link is constant and has been measured a-priori. As a notational convention, we collect the time-delays over all edges in the vector $\mathbf{d}_u = [d_{u,1}, \ldots, d_{u,|\mathcal{E}_u|}]^\mathsf{T}$.

The objective of this paper is to optimally place the operators required for a stream processing task on the nodes of the underlay network. Thus, corresponding to the underlay network, also the stream processing task is modelled as a graph, i.e., the operator network also referred to as *overlay network*. The overlay network is described by the graph $\mathcal{G}_o = (\mathcal{N}_o, \mathcal{E}_o)$, which is assumed to be simple and directed. Each edge $l \in \mathcal{E}_o$ corresponds to a link in the overlay network between two stream processing operators in $\mathcal{N}_o$, which are denoted by $\mathrm{s}(l)$ and $\mathrm{e}(l)$. Again, the time-delays over all edges are collected in the vector $\mathbf{d}_o = [d_{o,1}, \ldots, d_{o,|\mathcal{E}_o|}]^\mathsf{T}$. The required bandwidth of link l is denoted by $r_l > 0$, $l = 1, \ldots, |\mathcal{E}_o|$, and we write $\mathbf{r} = [r_1, \ldots, r_{|\mathcal{E}_o|}]^\mathsf{T}$. The incidence matrix of $\mathcal{G}_o$ is denoted by E_o.

Some nodes of the operator graph $\mathcal{N}_o$ represent sources or sinks, where the data stream is generated or terminates, respectively. We denote those nodes with $\mathcal{F}_o \subset \mathcal{N}_o$. All nodes in $\mathcal{F}_o$ have a fixed location in the underlay network and cannot be moved. All other nodes, i.e. those nodes in $\mathcal{N}_o \setminus \mathcal{F}_o$, represent freely placeable operators. The relationship between the underlay and overlay graph as well as the processors and the freely placeable operators is depicted exemplarily in Fig. 1. Note that each flow is assumed to be mapped to a single path, which always leads to an optimal placement, as our analysis will show. In case of bandwidth restrictions on underlay links, the splitting of flows over several paths may be necessary or beneficial. However, we here consider the case without bandwidth restrictions.

The *Operator Placement Problem* is now to find an optimal assignment of the freely placeable operators to the nodes of the underlay network, such that the network usage is minimized. Thus, we aim to minimize the *bandwidth-delay* product for the stream processing task

$$\min \sum_{l \in \mathcal{E}_o} d_{o,l} r_l. \tag{1}$$

subject to all constraints posed by the stream processing task.

III. Topology Aware Formulation

In this section, we assume that the entire topology of the underlay network is known and the network routes must be configured. The topology of the underlay network is described by the incidence matrix $E_u \in \mathbb{R}^{|\mathcal{N}_u| \times |\mathcal{E}_u|}$ of the graph $\mathcal{G}_u$. The incidence matrix E_u is a $\{0, \pm 1\}$-matrix with rows and columns indexed by the nodes and edges of $\mathcal{G}_u$ such that $\left[E_u\right]_{ik}$ has value $+1$ if $i = \mathrm{s}(k)$ (i.e. node i is the initial node of edge k), -1 if $i = \mathrm{e}(k)$ (i.e. i is the terminal node of k), and 0 otherwise.

A. Linear Program Formulation

To encode the paths in the underlay network that realize the overlay edges, we define the path indicator vectors $\mathbf{x}_l \in \{0,1\}^{|\mathcal{E}_u|}$, $l = 1, \ldots, |\mathcal{E}_o|$. The elements of $\mathbf{x}_l$ are one if the corresponding underlay edge belongs to the path, and zero otherwise. Furthermore, we define variables $\mathbf{p}_j \in \{0,1\}^{|\mathcal{N}_u|}$, $j = 1, \ldots, |\mathcal{N}_o|$, which encode the position of operator j in the underlay network. The indicator vector $\mathbf{p}_j$ contains only a single non-zero entry 1 in the element corresponding to the node in $\mathcal{N}_u$ where operator j is located. Hence, $\mathbf{1}^\mathsf{T} \mathbf{p}_j = 1$ for all $j = 1, \ldots, |\mathcal{N}_o|$.

Having defined these variables, the operator placement problem can be stated as follows. By using the path indicator vector $\mathbf{x}_j$, the delay of the corresponding overlay edge j can be calculated from the underlay delay vector as

$$d_{o,l} = \mathbf{d}_u^\mathsf{T} \mathbf{x}_l. \tag{2}$$

Moreover, $\mathbf{x}_l$ represents a path from overlay node $\mathrm{s}(l)$ to overlay node $\mathrm{e}(l)$ if and only if $E_u \mathbf{x}_l = \mathbf{p}_{\mathrm{e}(l)} - \mathbf{p}_{\mathrm{s}(l)}$. Thus, the optimal operator placement problem becomes

$$\begin{aligned} \min \quad & \mathbf{d}_u^\mathsf{T} \sum_{l=1}^{|\mathcal{E}_o|} r_l \mathbf{x}_l && \\ \text{s.t.} \quad & E_u \mathbf{x}_l = \mathbf{p}_{\mathrm{e}(l)} - \mathbf{p}_{\mathrm{s}(l)} && l = 1, \ldots, |\mathcal{E}_o| \\ & \mathbf{x}_l \in \{0,1\}^{|\mathcal{E}_u|} && l = 1, \ldots, |\mathcal{E}_o| \\ & \mathbf{p}_j \in \{0,1\}^{|\mathcal{E}_u|},\ \mathbf{1}^\mathsf{T} \mathbf{p}_j = 1 && j = 1, \ldots, |\mathcal{N}_o| \\ & \mathbf{p}_k = \bar{\mathbf{p}}_k && k \in \mathcal{F}_o. \end{aligned} \tag{3}$$

Note that the positions of the source and sink operators $k \in \mathcal{F}_o$ are constrained to the fixed positions $\bar{\mathbf{p}}_k$. The optimization problem (3) is an integer constrained linear program. Problems of this kind are in general non-convex and therefore hard to solve. However, the present problem admits an exact integer relaxation, which is presented in the following section.

B. Relaxation

We first relax the integer constraints in (3) and then show that the resulting linear program (LP) solves the original integer constrained problem exactly. The relaxed linear program is given by

$$\begin{aligned} \min \quad & \mathbf{d}_u^\mathsf{T} \sum_{l=1}^{|\mathcal{E}_o|} r_l \mathbf{x}_l && \\ \text{s.t.} \quad & E_u \mathbf{x}_l = \mathbf{p}_{\mathrm{e}(l)} - \mathbf{p}_{\mathrm{s}(l)} && l = 1, \ldots, |\mathcal{E}_o| \\ & \mathbf{x}_l \geq 0 && l = 1, \ldots, |\mathcal{E}_o| \\ & \mathbf{p}_j \geq 0,\ \mathbf{1}^\mathsf{T} \mathbf{p}_j = 1 && j = 1, \ldots, |\mathcal{N}_o| \\ & \mathbf{p}_k = \bar{\mathbf{p}}_k && k \in \mathcal{F}_o. \end{aligned} \tag{4}$$

In order to show that the solution of (4) solves the integer constrained problem (3), we rewrite (4) in standard form

$$\begin{aligned} \min \quad & \mathbf{c}^\mathsf{T} \mathbf{y} \\ \text{s.t.} \quad & A\mathbf{y} = \mathbf{b} \\ & \mathbf{y} \geq 0, \end{aligned} \tag{5}$$

with decision variable

$$\mathbf{y} = \begin{bmatrix} \mathbf{x}_1^\mathsf{T} \cdots \mathbf{x}_{|\mathcal{E}_o|}^\mathsf{T} \ \mathbf{p}_1^\mathsf{T} \cdots \mathbf{p}_{|\mathcal{N}_o|}^\mathsf{T} \end{bmatrix}^\mathsf{T} \tag{6}$$

and cost gradient $\mathbf{c}^\mathsf{T} = \begin{bmatrix} (\mathbf{r}^\mathsf{T} \otimes \mathbf{d}_u^\mathsf{T}) & 0 \end{bmatrix}$. The constraint matrix A and constraint vector $\mathbf{b}$ can be constructed as

$$A = \begin{bmatrix} (I_{|\mathcal{E}_o|} \otimes E_u) & -(E_o^\mathsf{T} \otimes I_{|\mathcal{N}_u|}) \\ 0 & (I_{|\mathcal{N}_o|} \otimes \mathbf{1}_{|\mathcal{N}_u|}^\mathsf{T}) \\ 0 & (F \otimes I_{|\mathcal{N}_u|}) \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ \mathbf{1}_{|\mathcal{N}_o|} \\ \bar{\mathbf{p}} \end{bmatrix}. \tag{7}$$

The matrix $F \in \{0,1\}^{|\mathcal{F}_o| \times |\mathcal{N}_o|}$ has one row for each fixed operator $k \in \mathcal{F}_o$, with a single 1 in column k. The vector $\bar{\mathbf{p}}$ is the stack vector of the corresponding fixed positions $\bar{\mathbf{p}}_k$.

To show that the solution of the relaxed problem (4) solves the integer program (3), we employ the concept of totally unimodular matrices.

Definition 1 ([11]). A square integer matrix is called *unimodular* if its determinant is 0, +1 or −1.

Definition 2 ([11]). A matrix M is called *totally unimodular* if each square submatrix of M is unimodular.

Facts ([11]).

1) If M is totally unimodular, then $\begin{bmatrix} M & I \end{bmatrix}$ is also totally unimodular.
2) If M is totally unimodular, then M^T is also totally unimodular.
3) Total unimodularity is preserved, if a row with at most one $+1$ or -1 is inserted.

4) A $\{0, \pm 1\}$-matrix with exactly one $+1$ and one -1 in each column is totally unimodular. Hence, all incidence matrices are totally unimodular.

Theorem 1 ([11]). *The LP* (5) *has an integral optimal solution for all integer vectors* $\mathbf{b}$ *for which it has a finite optimal solution value, if* A *is totally unimodular.*

Theorem 2 ([12]). *A matrix* $M \in \{-1, 0, +1\}^{n \times m}$ *is totally unimodular if and only if each row-subset* $\mathcal{I} \subset \{1, \ldots, n\}$ *of* M *can be partitioned into index sets* $\mathcal{I}_1$ *and* $\mathcal{I}_2$*,* $\mathcal{I}_1 \cup \mathcal{I}_2 = \mathcal{I}$*,* $\mathcal{I}_1 \cap \mathcal{I}_2 = \emptyset$*, such that for each column* $j \in \{1, \ldots, m\}$*,*

$$\left(\sum_{i \in \mathcal{I}_1} M_{ij} - \sum_{i \in \mathcal{I}_2} M_{ij} \right) \in \{-1, 0, 1\}. \tag{8}$$

With these preliminaries in place, we are ready to state the main result of this section.

Theorem 3. *The optimal solution of the integer relaxed linear program* (4) *also solves the integer constrained linear program* (3).

Proof: According to Thm. 1, the integer relaxed linear program (4) also solves the integer constrained linear program (3) if $\mathbf{b}$ is integral and A is totally unimodular. The vector $\mathbf{b}$ in (7) is integral, therefore it remains to show that A is totally unimodular.

Each row of

$$\begin{bmatrix} 0 & (F \otimes I_{|\mathcal{N}_u|}) \end{bmatrix}$$

contains exactly one non-zero element. Hence, these rows can be removed from A and it remains to show that

$$M = \begin{bmatrix} (I_{|\mathcal{E}_o|} \otimes E_u) & -(E_o^{\mathsf{T}} \otimes I_{|\mathcal{N}_u|}) \\ 0 & (I_{|\mathcal{N}_o|} \otimes \mathbf{1}_{|\mathcal{N}_u|}^{\mathsf{T}}) \end{bmatrix} \tag{9}$$

is totally unimodular.

To prove that M is totally unimodular, we introduce

$$\hat{M} = \begin{bmatrix} -E_o^{\mathsf{T}} \\ I_{|\mathcal{N}_o|} \end{bmatrix}, \tag{10}$$

which is totally unimodular due to Facts 1, 2, and 4 and noting that E_o is an incidence matrix. By Thm. 2 we know that each collection $\hat{\mathcal{I}}$ of rows of $\hat{M}$ can be split into two index sets $\hat{\mathcal{I}}_1$ and $\hat{\mathcal{I}}_2$, $\hat{\mathcal{I}}_1 \cup \hat{\mathcal{I}}_2 = \hat{\mathcal{I}}$, $\hat{\mathcal{I}}_1 \cap \hat{\mathcal{I}}_2 = \emptyset$ so that the sum of the rows in $\hat{\mathcal{I}}_1$ minus the sum of the rows in $\hat{\mathcal{I}}_2$ is a vector with entries only 0, $+1$ or -1. Next, we show that for each given row-subset $\mathcal{I}$ of M, we can construct an adequate partition based on the partitioning of $\hat{\mathcal{I}}$.

In the following, we will exploit the similarities in the structure of M and $\hat{M}$. Each row in $-E_o^{\mathsf{T}}$ of $\hat{M}$ corresponds to a block of $|\mathcal{N}_u|$ rows in M of the form

$$R = \begin{bmatrix} 0 & \cdots & E_u & \cdots & 0 & a_1 I_{|\mathcal{N}_u|} & \cdots & a_{|\mathcal{N}_o|} I_{|\mathcal{N}_u|} \end{bmatrix},$$

where a_i are the entries of the corresponding row from $-E_o^{\mathsf{T}}$. Similarly, each row in $I_{|\mathcal{N}_o|}$ of $\hat{M}$ corresponds to one block in $I_{|\mathcal{N}_o|} \otimes \mathbf{1}_{|\mathcal{N}_u|}^{\mathsf{T}}$ of M.

Given $\mathcal{I}$, we obtain a row-subset $\hat{\mathcal{I}}$ of $\hat{M}$ by choosing those rows, which correspond to the blocks of M containing one or more rows of the set $\mathcal{I}$. As already stated, there exists an adequate partition of $\hat{\mathcal{I}}$ into $\hat{\mathcal{I}}_1$ and $\hat{\mathcal{I}}_2$. The corresponding partition of $\mathcal{I}$ into $\mathcal{I}_1$ and $\mathcal{I}_2$ can be constructed as follows. If a row in $\mathcal{I}$ is part of a block of M which corresponds to a row of $\hat{M}$ in $\hat{\mathcal{I}}_1$, then this row is assigned to $\mathcal{I}_1$ and vice versa. By construction, all rows of $\mathcal{I}$ within one block R are in the same index set $\mathcal{I}_1$ or $\mathcal{I}_2$. Since the support columns of the E_u-blocks in M are all disjunct and each arbitrary selection of rows of E_u sums up to a $\{0, \pm 1\}$-vector, it holds that

$$\left(\sum_{i \in \mathcal{I}_1} M_{ij} - \sum_{i \in \mathcal{I}_2} M_{ij} \right) \in \{-1, 0, 1\}$$

for each column $j = \{1, \ldots, |\mathcal{E}_o||\mathcal{E}_u|\}$ of (9). The same holds for the remaining columns $j = \{|\mathcal{E}_o||\mathcal{E}_u| + 1, \ldots, |\mathcal{E}_o||\mathcal{E}_u| + |\mathcal{N}_o||\mathcal{N}_u|\}$ due to the structure of (9) and (10). Therefore, M is totally unimodular by Thm. 2 and thus A is totally unimodular. ■

Thm. 3 shows that the optimal operator placement problem can be solved efficiently, despite the integer constraints in the original problem formulation. It is in fact possible to solve the exact operator placement problem using standard linear programming tools. Note that the problem formulation (3) made no special restrictions on the underlay network topology, but required complete knowledge thereof. In fact, the solution of (3) contains explicitly the optimal path vectors in the underlay network, corresponding to the routing path used for the data streaming. This implies that an implementation of the optimal operator placement, according to (3), additionally provides the routing mechanism of the underlying physical network. However, the requirement for explicit knowledge of the underlay network topology might be undesirable for some applications. Therefore, we now present an alternative formulation of the optimal operator placement problem, where only the end-to-end delays must be known.

IV. End-to-End Delay Aware Formulation

In this section, we assume that the underlay network is strongly connected and provides routing in a fashion transparent to the overlay application. In contrast to the previous problem formulation, we assume now that only the end-to-end delay between any two underlay nodes can be directly measured, while the underlying network topology is unknown. The end-to-end delays are then collected in the *latency matrix* $D \in \mathbb{R}_+^{|\mathcal{N}_u| \times |\mathcal{N}_u|}$, where $D_{i_1 i_2}$ denotes the time-delay from underlay node i_1 to underlay node i_2, and we assume $D_{ii} = 0$, $\forall i \in \mathcal{E}_u$.

A. Bilinear Program Formulation

We consider now only the operator position vectors $\mathbf{p}_j \in \{0,1\}^{|\mathcal{N}_u|}$, as already defined for the topology aware problem formulation (3), as decision variables. Using the latency matrix D, the optimal operator placement problem can now be formulated as the bilinear, integer constrained problem

$$\min \sum_{l\in\mathcal{E}_o} r_l\, \mathbf{p}_{\mathrm{s}(l)}^{\mathsf{T}} D\, \mathbf{p}_{\mathrm{e}(l)} \tag{11a}$$

$$\text{s.t.} \quad \mathbf{p}_j \in \{0,1\}^{|\mathcal{N}_u|},\ \mathbf{1}^{\mathsf{T}}\mathbf{p}_j = 1, \qquad j \in \mathcal{N}_o \tag{11b}$$

$$\mathbf{p}_k = \bar{\mathbf{p}}_k, \qquad k \in \mathcal{F}_o. \tag{11c}$$

The formulation (11) results from a direct modelling of the operator placement problem. Note, however, that the bilinear integer program is non-convex and does not necessarily possess a unique minimizer. Solving problem (11) directly using numerical solvers can become computationally expensive. Due to the non-convexity, it is also not guaranteed whether numerical solvers can find the global optimal solution. To overcome these issues, we propose in the following an exact reformulation of (11) as a linear program, which can be solved very efficiently.

B. Relaxation

We show first that the problem (11) can be reformulated as an equivalent integer relaxed linear program, which solves the original bilinear integer problem exactly. This is done by introducing the auxiliary variables $\mathbf{x}_l = \mathrm{vec}\big(\mathbf{p}_{\mathrm{e}(l)}\mathbf{p}_{\mathrm{s}(l)}^{\mathsf{T}}\big)$, $l \in \mathcal{E}_o$, where $\mathrm{vec}(\cdot)$ denotes the column-first vectorization of a matrix, and relaxing the integer constraints (11b) to

$$\mathbf{p}_j \geq 0 \quad \text{and} \quad \mathbf{1}^{\mathsf{T}}\mathbf{p}_j = 1.$$

The auxiliary variables $\mathbf{x}_l$ now correspond to the mapping of an overlay edge l to an underlay path. In order to set up the linear cost function, we first note that

$$\mathbf{p}_{j_1}^{\mathsf{T}} D\, \mathbf{p}_{j_2} = \mathrm{tr}\big(D\mathbf{p}_{j_1}\mathbf{p}_{j_2}^{\mathsf{T}}\big) = \mathrm{vec}\big(D^{\mathsf{T}}\big)^{\mathsf{T}}\,\mathrm{vec}\big(\mathbf{p}_{j_1}\mathbf{p}_{j_2}^{\mathsf{T}}\big).$$

Since for every node $j \in \mathcal{N}_o$, any valid solution $\mathbf{p}_j$ must be a canonical basis vector as per (11b), we can set up constraints for the auxiliary variables using the following identities, which hold for any pair of nodes j_1 and j_2:

$$\begin{aligned} \mathbf{1}^{\mathsf{T}}\big(\mathbf{p}_{j_1}\mathbf{p}_{j_2}^{\mathsf{T}}\big)\,\mathbf{1} &= 1, \\ \mathbf{1}^{\mathsf{T}}\big(\mathbf{p}_{j_1}\mathbf{p}_{j_2}^{\mathsf{T}}\big) \quad &= \mathbf{p}_{j_2}^{\mathsf{T}}, \\ \text{and} \qquad \big(\mathbf{p}_{j_1}\mathbf{p}_{j_2}^{\mathsf{T}}\big)\,\mathbf{1} &= \mathbf{p}_{j_1}. \end{aligned}$$

Now, the bilinear program (11) can be rewritten as a linear program in standard form (5) and (6) with:

$$\mathbf{c} = \begin{bmatrix} \mathbf{r}^{\mathsf{T}} \otimes \mathrm{vec}\big(D^{\mathsf{T}}\big)^{\mathsf{T}} & \mathbf{0}_{|\mathcal{N}_o|\cdot|\mathcal{N}_u|}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} \tag{12a}$$

$$A = \begin{bmatrix} I_{|\mathcal{E}_o|} \otimes \mathbf{1}_{|\mathcal{N}_u|^2}^{\mathsf{T}} & 0 \\ I_{|\mathcal{E}_o|\cdot|\mathcal{N}_u|} \otimes \mathbf{1}_{|\mathcal{N}_u|}^{\mathsf{T}} & -{E_o^+}^{\mathsf{T}} \otimes I_{|\mathcal{N}_u|} \\ I_{|\mathcal{E}_o|} \otimes \mathbf{1}_{|\mathcal{N}_u|}^{\mathsf{T}} \otimes I_{|\mathcal{N}_u|} & -{E_o^-}^{\mathsf{T}} \otimes I_{|\mathcal{N}_u|} \\ 0 & F \otimes I_{|\mathcal{N}_u|} \\ 0 & \bar{F} \otimes \mathbf{1}_{|\mathcal{N}_u|}^{\mathsf{T}} \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} \mathbf{1} \\ 0 \\ 0 \\ \bar{\mathbf{p}} \\ \mathbf{1} \end{bmatrix} \tag{12b}$$

Here, the matrix $E_o^+ \in \{0,1\}^{|\mathcal{N}_o|\times|\mathcal{E}_o|}$ is composed of the non-negative entries of E_o, whereas $E_o^- \in \{0,1\}^{|\mathcal{N}_o|\times|\mathcal{E}_o|}$ is composed of the non-negative entries of $-E_o$, so that $E_o = E_o^+ - E_o^-$. In other words, E_o^+ and E_o^- contain the start and end nodes, respectively, of all overlay edges $\mathcal{E}_o$. Again, $\bar{\mathbf{p}}$ denotes the stack vector of all fixed operator positions $\bar{\mathbf{p}}_k$, and the matrix F has one row for each *fixed* operator $k \in \mathcal{F}_o$, with a single 1 in column k. Correspondingly, the matrix $\bar{F}$ has one row for each *free* operator $k \in \mathcal{N}_o \setminus \mathcal{F}_o$, with a single 1 in column k. We show now that the linear program with cost and constraints defined according to (12) has an integer solution corresponding to the optimal operator placement.

Theorem 4. *The solution of the linear program* (5) *with A,* $\mathbf{b}$ *and* $\mathbf{c}$ *as in* (12) *also solves the integer constrained bilinear program* (11)*.*

We prove this theorem by showing that the feasible set of the linear program in question is given by an integral polyhedron. The total unimodularity argument used in Sec. III-B is not applicable in this case, and we therefore apply an alternative argument based on the following theorem.

Theorem 5 ([13])**.** *Consider an integer $r \times m$ matrix M with full row rank r. The polyhedron $P(M,\mathbf{v}) = \big\{\mathbf{x} \mid M\mathbf{x} = \mathbf{v}, \mathbf{x} \geq \mathbf{0}\big\}$ is integral for all integer vectors* $\mathbf{v}$ *if and only if every $r \times r$ submatrix of M is unimodular.*

Lemma 1. *Let I_n denote the $n \times n$ identity matrix, I_n' the $(n-1) \times n$ matrix obtained by deleting the first row of I_n, and*

$$B_n = \begin{bmatrix} I_n' \otimes \mathbf{1}_n^{\mathsf{T}} \\ \mathbf{1}_n^{\mathsf{T}} \otimes I_n \end{bmatrix}.$$

For any positive integer n, the following holds.

(i) B_n has full row rank $(2n-1)$.

(ii) B_n is totally unimodular.

Proof: The matrix B_n has $(2n-1)$ rows and contains the nonsingular $(2n-1) \times (2n-1)$ submatrix $\big[0\ I_n'; I_n\ I_n\big]$, which proves (i). Furthermore, (ii) can easily be seen to be true by recognizing that B_n can be obtained from an incidence matrix of a bipartite graph through negation of rows and addition of unity columns. ∎

As the matrix A in (12b) is rank deficient, we first find a full rank representation $\tilde{A}\mathbf{x} = \tilde{\mathbf{b}}$ of the equality constraint given by (12b) using Lem. 1. We then prove Thm. 4 by showing that every basis of $\tilde{A}$ is unimodular.

Proof of Thm. 4: In order to find a full rank representation of $A\mathbf{x} = \mathbf{b}$ with (12b), we first partition the rows of A into index sets $\mathcal{I}_\mathcal{E}, \mathcal{I}^+, \mathcal{I}^-, \mathcal{I}_\mathcal{N}$, such that $\mathcal{I}^+$ contains the rows corresponding to E_o^+, $\mathcal{I}^-$ contains the rows corresponding to E_o^-, and $\mathcal{I}_\mathcal{E}$ contains the rows above and $\mathcal{I}_\mathcal{N}$ the rows below.

Now, we identify rows of A which correspond to redundant constraints. Firstly, all rows $\mathcal{I}_\mathcal{E}$ are reduntant, as they can be expressed as linear combinations of $\mathcal{I}^+$ or $\mathcal{I}^-$, together with $\mathcal{I}_\mathcal{N}$. Secondly, the $(k \cdot |\mathcal{N}_u| - 1)^{\text{th}}$ rows out of $\mathcal{I}^+$, with $k = 0, \ldots, |\mathcal{E}_o| - 1$, can be expressed as linear combinations of the remaining rows of $\mathcal{I}^\pm$ and $\mathcal{I}_\mathcal{N}$, and are therefore redundant. After removal of these redundant rows from A, and permutations of rows in $\mathcal{I}^\pm$ and columns corresponding to nonzero entries in $\mathcal{I}_\mathcal{N}$, a full rank representation of A is given by

$$\tilde{A} = \begin{array}{c} \begin{array}{ccc} \mathcal{C}_1 & \mathcal{C}_2 & \mathcal{C}_3 \end{array} \\ \begin{bmatrix} I_{|\mathcal{E}_o|} \otimes B_{|\mathcal{N}_u|} & * & * \\ 0 & I_{|\mathcal{N}_u| \cdot |\mathcal{F}_o|} & 0 \\ 0 & 0 & I_{|\mathcal{N}_o \setminus \mathcal{F}_o|} \otimes \mathbf{1}^\mathsf{T}_{|\mathcal{N}_u|} \end{bmatrix} \end{array}, \tag{13}$$

where the columns are partitioned into sets $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ as shown above and the $*$ blocks contain some -1 entries. The corresponding vector $\tilde{\mathbf{b}}$ is given by a permuted entry subset of $\mathbf{b}$ and is therefore also integer. It is easy to see that $\tilde{A}$ has full row rank due to its block triangular structure and full row rank of the B-blocks due to Lem. 1(i).

All basis matrices of $\tilde{A}$ are also block triangular and must contain $\big(2 \cdot |\mathcal{N}_u| - 1\big)$ linear independent columns from each B-block in $\mathcal{C}_1$, all columns $\mathcal{C}_2$ and one column from each $\mathbf{1}^\mathsf{T}$-block in $\mathcal{C}_3$, in order to be non-singular. Recall that the determinant of a block triangular matrix equals the product of determinants of its diagonal blocks. Therefore, the determinants of all bases of $\tilde{A}$ are given by $\prod_{i=1}^{|\mathcal{E}_o|} \det \hat{B}^i$, where each $\hat{B}^i$ is an arbitrary basis of $B_{|\mathcal{N}_u|}$. By Lem. 1(ii), all these determinants equal ± 1.

This shows that the feasible polyhedron of (5) and (12) coincides with the integral polyhedron $P(\tilde{A}, \tilde{\mathbf{b}})$, thereby concluding the proof of Thm. 4. ∎

V. Simulation Study

The advantage of the results presented in this paper is that, now, exact and computationally attractive formulations of the optimal operator placement problem have been derived. In fact, both problem formulations (4) and (12) can be solved directly using standard linear programming solvers. We evaluated the completion time of Matlab's default large-scale interior-point linear programming solver for both optimization problems on a consumer-grade personal computer. The overlay graph used was the tree with six nodes shown in Fig. 1, with bandwidth values chosen as follows:

$$r_1 = 1.0,\ r_2 = 1.0,\ r_3 = 1.0,\ r_4 = 1.5,\ r_5 = 1.8$$

The underlay network was chosen as a strongly connected random graph with 200 nodes. Two different scenarios were considered: first, a loosely connected network with an edge probability of only about 5%, and second, a densely connected network with an edge probability of over 60%, both with random edge delays uniformly distributed between 1 and 10. Each scenario was evaluated using 100 realizations of the random graph.

Formulation	TA (4)	EEDA (12)
Dimension	$10\,910 \pm 1\,262$	201 200
Constraints	1 806	2 811
Avg. time $E(t)$ $[s]$	1.3693	4.7511
Var(t) $[s]$	0.0358	0.1610

Table I
Simulation results for 100 random underlay graphs with $|\mathcal{N}_u| = 200$ and $|\mathcal{E}_u| = 1\,942 \pm 50$ (edge probability of 4.9%).

Formulation	TA (4)	EEDA (12)
Dimension	$126\,960 \pm 97\,230$	201 200
Constraints	1 806	2 811
Avg. time $E(t)$ $[s]$	3.0550	4.7749
Var(t) $[s]$	0.0611	0.1989

Table II
Simulation results for 100 random underlay graphs with $|\mathcal{N}_u| = 200$ and $|\mathcal{E}_u| = 25\,152 \pm 3\,889$ (edge probability 63.2%).

Tables I and II show the problem dimensions, number of constraints and average execution times of the LP solver for both the Topology Aware (TA, cf. Sec. III) and the End-to-End Delay Aware (EEDA, cf. Sec. IV) problem formulations. Note that the dimension of the TA problem formulation (4) depends on the number of underlay edges $|\mathcal{E}_u|$, resulting in a high impact on execution time. For the EEDA problem formulation (12), the latency matrix D was given by the all-to-all shortest path latencies of $\mathcal{G}_u$. The simulations show that our novel problem formulations can be solved very efficiently within a few seconds. Together with the fact that both formulations yield exact solutions of the optimal operator placement problem, these results clearly justify the theoretical approach pursued in this paper.

VI. Conclusion

In this paper, we studied the network oriented optimal operator placement problem. We provided two formulations for the problem: First, the problem was formulated as an integer constrained linear program which includes an exact assignment of data streams to links in the underlying network. Second, an integer constrained bilinear formulation of the problem was presented which reduces the complexity of the problem by relying on the routing capabilities of the underlay network. We have shown for both problems, that a linear programming relaxation of the integrality constraints maintains the exact solution. Therefore, both problem formulations can be reduced to standard linear programs, which can be solved very efficiently. Our simulation study has shown that the formulations allow the quick solution of the problem using standard LP solvers.

Based on these problem formulations, we are working on a distributed version of the optimal operator placement algorithm. This way, an exact solution to the problem can be provided without the need to gather data about the entire network at a central location.

Acknowledgement

The authors thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart.

R. B., M. B., and F. A. also acknowledge the financial support by the DFG Priority Program 1305 "Control Theory of Digitally Networked Dynamical Systems".

References

[1] S. Rizou, K. Häussermann, F. Dürr, N. Cipriani, and K. Rothermel, "A system for distributed context reasoning," in *Sixth Intl. Conf. Autonomic and Autonomous Systems (ICAS '10)*, Mar. 2010, pp. 84–89.

[2] L. Shi, "Kalman filtering over graphs: Theory and applications," *IEEE Trans. Automatic Control*, vol. 54, no. 9, pp. 2230–2234, Sept. 2009.

[3] A. Benzing, B. Koldehofe, M. Völz, and K. Rothermel, "Multilevel predictions for the aggregation of data in global sensor networks," in *Proc. 14th IEEE/ACM Intl. Symp. Distributed Simulation and Real Time Applications (DS-RT '10)*, Oct. 2010, pp. 169–178.

[4] G. Lakshmanan, Y. Li, and R. Strom, "Placement strategies for internet-scale data stream systems," *IEEE Internet Computing*, vol. 12, no. 6, pp. 50–60, Nov. 2008.

[5] L. Amini, N. Jain, A. Sehgal, J. Silber, and O. Verscheure, "Adaptive control of extreme-scale stream processing systems," in *Proc. 26th IEEE Intl. Conf. Distributed Computing Systems (ICDCS '06)*, 2006, pp. 71–71.

[6] O. Papaemmanouil, U. Çetintemel, and J. Jannotti, "Supporting generic cost models for wide-area stream processing," in *Proc. 25th Intl. Conf. Data Engineering (ICDE '09)*, 2009, pp. 1084–1095.

[7] U. Srivastava, K. Munagala, and J. Widom, "Operator placement for in-network stream query processing," in *Proc. 24th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS '05)*, 2005, pp. 250–258.

[8] Y. Ahmad and U. Çetintemel, "Network-aware query processing for stream-based applications," in *Proc. 30th Intl. Conf. Very Large Data Bases (VLDB '04)*, 2004, pp. 456–467.

[9] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, and M. Seltzer, "Network-aware operator placement for stream-processing systems," in *Proc. 22nd Intl. Conf. Data Engineering (ICDE '06)*, Apr. 2006, pp. 49–60.

[10] S. Rizou, F. Dürr, and K. Rothermel, "Fulfilling end-to-end latency constraints in large-scale streaming environments," in *Proc. 30th IEEE Intl. Performance Computing and Communications Conf. (IPCCC)*, Nov. 2011, pp. 1–8.

[11] A. Schrijver, *Theory of Linear and Integer Programming*. John Wiley & Sons Inc, 1998.

[12] A. Ghouila-Houri, "Caractérisation des matrices totalement unimodulaires," *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences*, vol. 254, pp. 1192–1194, 1962.

[13] A. F. Veinott, Jr. and G. B. Dantzig, "Integral extreme points," *SIAM Rev.*, vol. 10, no. 3, pp. 371–372, July 1968.