# Buffer Sizing for Rate-optimal Single-rate Dataflow Scheduling Revisited

Orlando Moreira[1], Twan Basten[2,3], Marc Geilen[2] and Sander Stuijk[2]

[1]ST-Ericsson, Eindhoven, Netherlands
orlando.moreira@stericsson.com
[2]Eindhoven University of Technology, Eindhoven, Netherlands
{a.a.basten,m.c.w.geilen,s.stuijk}@tue.nl
[3]Embedded Systems Institute, Eindhoven, Netherlands

***Abstract*— Single-Rate Dataflow (SRDF) graphs, also known as Homogeneous Synchronous Dataflow (HSDF) graphs or Marked Graphs, are often used to model the implementation and do temporal analysis of concurrent DSP and multimedia applications. An important problem in implementing applications expressed as SRDF graphs is the computation of the minimal amount of buffering needed to implement a static periodic schedule that is optimal in terms of execution rate, or throughput. Ning and Gao propose in [1] a linear-programming-based polynomial algorithm to compute this minimal storage amount, claiming optimality. We show via a counter-example that the proposed algorithm is not optimal. We prove that the problem is in fact NP-complete. We give an exact solution, and experimentally evaluate the degree of inaccuracy of the algorithm of Ning and Gao.**

***Index Terms*— Scheduling, Single-rate Dataflow, Homogeneous Synchronous Dataflow, Buffer Minimization, Throughput Optimization.**

## I. Introduction

MANY flavors of dataflow formalisms are used to express, model, analyse, and map signal processing and multimedia streaming applications. Dataflow paradigms fit well with these application domains, as they can represent the inherent concurrency, the pipelined behavior, and data-oriented style of streaming algorithms, while at the same time allowing analysis and synthesis. One of the simplest but still rather expressive dataflow models is the Single-Rate DataFlow (SRDF) model, also referred to as Homogeneous Synchronous DataFlow (HSDF) or Marked Graphs. SRDF describes an application as a graph where nodes – typically referred to as actors – represent computation functions and edges represent first-in-first-out (FIFO) communication channels.

SRDF graphs have strong analytical properties. If a worst-case execution time is known for every actor, polynomial algorithms [2] can be used to derive the maximum guaranteed rate of output production (the throughput) that the iterative execution of the graph may achieve. This allows to verify real-time requirements. It is furthermore possible to use the abstract concepts of actors and edges to model properties of communication channels, memory mapping, and scheduler settings [3]–[5], therefore allowing for the analysis of dataflow graph implementations onto specific architectures.

When implementing an application described as an SRDF graph onto an architecture, finite sizes for the implementation of FIFO buffers need to be determined. Several variants of the buffer-size minimization problem exist. The most elementary one is the problem of finding minimal buffer sizes that guarantee a deadlock-free execution. However, in signal and multimedia processing, deadlock-free execution is often insufficient. It is often important to maximize the throughput, i.e., the rate of execution and output production. In that context, a very relevant variant of the buffer sizing problem for SRDF is the one described by Ning and Gao in a paper of 1993 [1], the Optimal Scheduling and Buffer Allocation (OSBA) problem. The goal in this problem formulation is to find a *periodic rate-optimal schedule* of execution for an SRDF graph with *minimal buffer requirements*, i.e., buffer sizes whose sum is minimal among all those buffer allocations that allow periodic rate-optimal graph execution. The restriction to periodic execution schedules is advantageous from the code-size point of view. The authors claim a linear-programming-based solution of polynomial complexity for this problem.

The complexity result of Ning and Gao has been unchallenged until now, and based on the results presented in [1] it has been widely accepted that the problem of finding the minimum buffer capacity that allows an SRDF graph to execute at its maximum throughput is a problem of polynomial complexity (see, for instance, [3], [6]–[10]). This is incorrect.

With a counter-example we show in this paper that the algorithm proposed in [1] is not optimal. We in fact prove that (the decision variant of) the OSBA problem is NP-complete. Based on the buffer sizing technique for multi-rate dataflow graphs of [11], [12], we provide an exact solution to the OSBA problem, that is efficient in practice. We estimate the difference between the results of the algorithm of [1] and the exact solution. The NP-completeness proof for the original OSBA formulation goes via a generalization of OSBA in which non-periodic schedules are allowed, while making assumptions on required buffer space that are more conservative than in [1]. We argue that in this generalized setting an SRDF graph always has a static periodic schedule that combines maximum throughput with buffer sizes that are minimal among *all* rate-optimal, not necessarily periodic, schedules. We show that this generalized OSBA variant is NP-complete, and use this result in the NP-completeness proof for the original OSBA problem.

The next section introduces the SRDF model and some

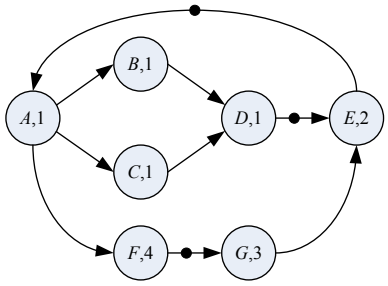


Fig. 1. An example SRDF graph.

of its properties. Section III defines the OSBA problem, and Section IV presents the solution proposed in [1]. Section V gives the counter-example that shows the sub-optimality of the OSBA solution of [1]. Section VI studies the mentioned generalization of the OSBA problem, and Section VII proves the NP-completeness of OSBA. Section VIII presents an exact solution, and experimentally evaluates the degree of sub-optimality of the approach of [1]. Section IX concludes.

## II. The Model and its Properties

### A. Notations and Terminology

Let $\mathbb{N}$ and $\mathbb{Z}$ be the sets of natural numbers and integers, and $\mathbb{N}_0$ and $\mathbb{N}_0^\infty$ the natural numbers plus 0 resp. 0 and infinity. A directed graph $G$ is a pair $(V, E)$, with $V$ the set of vertices, or nodes, and $E \subseteq V^2$ the set of directed edges. For $(i, j) \in E$, $i$ is the source and $j$ the sink of the edge. A (directed) path in the graph is simple if the source nodes of all edges on the path are all different. A path is a cycle if and only if it is simple and the source of the first edge equals the sink of the last edge. An undirected path is a path that ignores edge directions. A graph is connected if and only if there is an undirected path between any pair of nodes; it is strongly connected if and only if there is a directed path between any pair of nodes.

### B. Single-rate Dataflow Graphs

A Single-Rate DataFlow (SRDF) graph – also known as Homogeneous Synchronous Dataflow graph [13] – is a directed graph, where nodes are referred to as *actors* and represent data transformation or control entities, and edges represent FIFO queues that direct values from an actor output to an actor input. Data is transported in discrete chunks, called *tokens*. When an actor is activated by data availability it is said to be *fired*. The *firing rule* defines what happens upon firing an actor. SRDF prescribes that the number of tokens produced (consumed) by an actor on each output (from each input) per firing is always one. We assume SRDF graphs to be connected (but not necessarily strongly connected). Assumptions made throughout this section are listed in Table I.

For performance analysis, actors in an SRDF graph $(V, E)$ have a valuation $e : V \to \mathbb{N}_0$; $e(i)$ is the *execution time* of actor $i$. Edges have a valuation $d : E \to \mathbb{N}_0$; $d(i, j)$ is called the *delay* of edge $(i, j)$ and represents the number of initial tokens in $(i, j)$. An SRDF is completely defined by the tuple $(V, E, e, d)$. Figure 1 illustrates an example SRDF graph, with tokens shown as black dots on the edges and execution time annotations inside the actors. (Note that an actor execution time of zero is allowed. This will not occur for any realistic data transformation entities, but it is sometimes convenient to model certain (control) aspects.)

TABLE I
Overview of assumptions (all consistent with [1]).

| | |
|---|---|
| SRDF | - graphs are connected and live<br>- positive cycle means, integer MCM |
| Scheduling | - non-negative integer firing start and end times<br>- actor firings may not overlap in time<br>(actors have implicit self-edges with 1 initial token) |
| Buffering | - all output edges of an actor share one buffer<br>- input space is released at firing start and output space is claimed at firing start<br>- buffers can be read and written simultaneously<br>- implicit self-edges do not require buffering |

We are interested in applications that process data streams, which typically involve computations on indefinitely long data sequences. Therefore, we are only interested in SRDF graphs that can execute in a non-terminating fashion. We want schedules that execute all actors indefinitely within a finite amount of memory. An SRDF graph that allows to fire all actors infinitely often is called *live*. It follows from the results in [14, Sec. 3] for multi-rate dataflow graphs that a live SRDF graph can always be scheduled using a finite amount of memory, and that an SRDF graph is live if and only if all cycles contain at least one initial token. A consequence of the latter is that liveness of an SRDF graph can be verified efficiently by performing e.g. an all-pairs shortest path algorithm on the SRDF graph with the initial delays as edge weights or a cycle-detection via a depth-first search on the graph with all edges with initial tokens removed. Therefore, in the remainder, we limit ourselves to live SRDF graphs[1].

### C. Throughput

The cycle mean $\mu(c)$ of a cycle $c$ in an SRDF graph $G = (V, E, e, d)$ is defined as:

$$\mu(c) = \frac{\sum_{i \in N(c)} e(i)}{\sum_{(i,j) \in E(c)} d(i, j)}, \tag{1}$$

where $N(c)$ is the set of all nodes traversed by cycle $c$ and $E(c)$ is the set of all edges traversed by cycle $c$.

The *Maximum Cycle Mean (MCM)* $\mu(G)$ of SRDF graph $G$, with set of cycles $C(G)$, is defined as:

$$\mu(G) = \max_{c \in C(G)} \mu(c). \tag{2}$$

The inverse of the MCM of an SRDF graph provides a fundamental upperbound to its maximal attainable throughput, and it is known that every SRDF graph has at least one execution that achieves this maximal attainable throughput [15]. Many algorithms of polynomial complexity have been

[1]Note that liveness corresponds to deadlock-freeness for strongly connected SRDF graphs; for arbitrary (connected) graphs, liveness is a stronger property than deadlock-freeness, and the latter is insufficient to always allow execution within bounded memory.

proposed to find the MCM (see [2] for an overview). The MCM of the SRDF graph of Figure 1 is 5, resulting from the cycle through actors $A$, $F$, $G$, and $E$. The maximum attainable throughput is therefore 1/5 firings per time unit.

An MCM may be a fractionary value. The original OSBA formulation of [1] assumes that the MCM of an SRDF graph is a positive integer, as it is always possible to unroll a given SRDF graph to a graph with an integer MCM. We keep the assumption of [1] of a positive integer MCM, although this is not strictly necessary. We furthermore exclude cycles with cycle mean zero (i.e., with execution time zero) because such cycles are not meaningful from a practical point of view whereas they complicate the reasoning in the remainder.

### D. Schedule Functions and Buffer Capacities

Since actors are supposed to fire indefinitely, we can per actor index firings using the natural numbers, starting from 0. A *schedule function* is a function $s : V \times \mathbb{N}_0 \rightarrow \mathbb{N}_0$, where $s(i,k)$ represents the time at which instance $k$ of actor $i$ is fired. We denote the finishing time of firing $k$ of actor $i$ by $f(i,k) = s(i,k) + e(i)$. As in [1], we assume that scheduling times are non-negative integer values, although this could be relaxed to non-negative fractional values.

Not all schedule functions can be realized. A schedule is *admissible* if and only if all actor firing start times satisfy the firing rule. This means that all input edges of an actor must always have sufficient tokens to accommodate the firings. In [15], necessary and sufficient conditions for an admissible schedule are given. For notational convenience, assume that for any schedule $s$ and actor $i$, $s(i,k) = -e(i)$ if $k$ is negative, which allows us to see initial tokens as being produced at time 0 by a firing with negative start time. A schedule $s$ is admissible if and only if, for every $(i,j) \in E$ and any $k \in \mathbb{N}_0$,

$$s(j,k) \geq s(i, k - d(i,j)) + e(i). \tag{3}$$

Consider the example of Figure 1. Schedule $s$ with $s(A,k) = s(G,k) = 5k$, $s(B,k) = s(C,k) = s(F,k) = 1{+}5k$, $s(D,k) = 2{+}5k$, and $s(E,k) = 3{+}5k$ is an admissible schedule that realizes the maximal throughput of 1/5 firings per time unit. All actors fire as soon as they are enabled.

Equation (3) assumes that tokens are consumed at the beginning of a firing and produced at the end of a firing. An admissible schedule allows multiple firings of the same actor to overlap in time (so-called auto-concurrency). This is consistent with the general interpretation of SRDF graph execution, but in [1], it is assumed that firings of the same actor in a graph cannot overlap in time. We adhere to this assumption, excluding schedules in which firings of the same actor overlap. It is possible to enforce the restriction to non-auto-concurrent schedules in an SRDF graph, by introducing a self-edge for every actor in the graph with a single initial token for each of these edges. We implicitly assume that such self-edges are present when referring to the MCM of a graph and/or the maximal throughput that can be obtained. (This implies that the MCM is always at least the maximum actor execution time.) However, in line with [1], these implicit self-edges are not considered for buffer minimization.

In an implementation of a dataflow application, data may be consumed from input edges and produced on output edges at arbitrary points in time during an actor firing. To guarantee that buffers are sufficiently large, we need to make some assumptions. The most conservative buffer sizes are obtained when assuming that each SRDF edge needs its own buffer, that output space needed by actors is claimed at a firing start, and that input space is freed at the end of a firing. Claimed input and output space can then be used as working memory during actor execution. This type of behavior may typically be seen in multimedia applications in which an actor communicates substantial amounts of different data to different actors. Ning and Gao, however, assume that all output edges of an actor share buffer space and that input data is read at a firing start and stored internally, releasing input space at that point in time. They do assume that output space is claimed at the firing start. This type of behavior may be seen in DSP applications where actors reflect simple scalar operations of which the outcome is communicated to several successor actors simultaneously. We adhere to the choices made by Ning and Gao, but as part of our NP-completeness proof for OSBA we also study OSBA with the most conservative buffering assumptions in Section VI. Ning and Gao moreover assume that buffer space can be read and written simultaneously when actors $i$ and $j$ connected via edge $(i,j)$ start their firings at the same time. The space released by the start of $j$ can then be claimed as output space with the start of $i$. We also follow this assumption.

Minimum buffer sizes required to execute an admissible schedule can be derived from the schedule. We introduce an auxiliary notion, namely the token count $c(i,j,t)$ on an edge $(i,j)$ *after* the occurrence of all firing starts and endings scheduled at time $t$. Given the initial tokens $d$ in an SRDF graph and a schedule $s$, $c(i,j,t)$ is defined inductively as follows. Since execution times may be zero, the inductive computation starts at 'time' -1.

$$\begin{aligned} c(i,j,-1) &= d(i,j), \text{ and, for } t \in \mathbb{N}_0, \\ c(i,j,t) &= c(i,j,t-1) \\ &\quad -|\{s(j,k) = t \mid k \in \mathbb{N}_0\}| \\ &\quad +|\{f(i,k) = t \mid k \in \mathbb{N}_0\}|. \end{aligned} \tag{4}$$

The buffer space needed by an SRDF graph to execute a schedule is defined by a *buffer-capacity distribution function*. The buffering assumptions imply that buffer space can be assigned on a per-actor basis. Hence, a buffer-capacity distribution function $b : V \rightarrow \mathbb{N}_0^\infty$ is a function that assigns to each actor $i \in V$ the buffer capacity $b(i)$ required for the data on its output edges to implement a given schedule $s$. The basic idea is to take per actor per time point, including the artificial time point -1, the maximum of all token counts for all the actor's output edges and furthermore the maximum of all these values over time, while accommodating for actor firings that are in progress, because these firings imply the reservation of output space. At any point in time at most one firing of an actor can be in progress due to the restriction to non-auto-concurrent schedules. Assume $c(i,j,t) = 0$ if $(i,j) \notin E$.

$$\begin{aligned} b(i) = \max_{t \in \mathbb{N}_0 \cup \{-1\}} & \\ (\max_{j \in V} c(i,j,t) & \\ +|\{k \in \mathbb{N}_0 \mid s(i,k) \leq t < f(i,k)\}|). & \end{aligned} \tag{5}$$

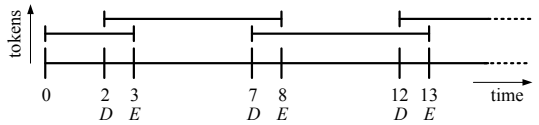


Fig. 2. Buffer requirements for tokens on edge $(D, E)$ in the example graph.

For example, to execute schedule $s$ for the running example introduced above, a buffer size of 1 token for every actor in the graph of Figure 1 is sufficient except for actor $D$, which needs a buffer of size 2 to store tokens on edge $(D, E)$. This yields a total buffer requirement of 8. Figure 2 visualizes the buffer requirements of actor $D$. Initially, one token is present in channel $(D, E)$, which is consumed by the first firing of $E$ at time 3. Furthermore, space for a token produced by $D$ is needed from the start of firing $k$ of $D$ at time $2 + 5k$ till the start of corresponding firing $k + 1$ of $E$ at time $8 + 5k$ that consumes the token. This leads to a maximal presence of two tokens in the channel, during intervals $[2 + 5k, 3 + 5k]$.

The definition of Equation (5) does typically not properly reflect the buffer capacity needed by an actor with execution time zero. However, as already explained, it is reasonable to assume that actor execution times are non-zero for any reasonable operation or data transformation. We only use actors with execution time zero later in various proofs, where an artificial (but precisely defined) number for certain buffer sizes is acceptable. We therefore prefer the simple definition of Equation (5) over a more involved definition that would be correct for actors with execution time zero. Such a definition would for example be possible using an operational semantics for SRDF graphs that makes all firing starts and endings at a single point in time explicit, as given in [11] and [12].

### E. Static Periodic Schedules

A *Static Periodic Schedule* (SPS) of an SRDF graph is a schedule such that, for all nodes $i \in V$:

$$s(i, k) = s(i, 0) + P \cdot k, \tag{6}$$

where $P$ is the period of the SPS. Given $P$, an SPS is defined by the values $s(i, 0)$, for all $i \in V$. We use $s(i)$ as an abbreviation for $s(i, 0)$ for an SPS $s$. The example schedule given above is an SPS with period 5. An important observation is that firing all actors the same number of times preserves the token distribution over the edges. Therefore, the minimal buffer sizes needed to execute an SPS can be computed from a prefix of an SPS consisting of one period plus any transient part that may be present. It is sufficient to consider an SPS $s$ up to time point $\max_{i \in V} s(i) + P$ (i.e., the latest start time of any actor plus one period). For the example schedule, it is sufficient to consider the execution up to time point 8.

The following theorem re-establishes, in a somewhat different form, a relation between SPSs and the MCM, first presented in [16].

*Theorem 1:* A live SRDF graph $G$ has an SPS with period $P$ if and only if $P \geq \mu(G)$.

*Proof:* Let $G = (V, E, e, d)$. According to Equation (3), every edge $(i, j)$ in $G$ imposes a constraint $s(j, k) \geq s(i, k - d(i, j)) + e(i)$ to any admissible schedule. For an SPS, from Equation (6), we obtain for every $(i, j) \in E$ a constraint $s(j) + P \cdot k \geq s(i) + P \cdot (k - d(i, j)) + e(i)$, which is equivalent to:

$$s(i) - s(j) \leq P \cdot d(i, j) - e(i). \tag{7}$$

These inequalities define a system of linear constraints. Since Equation (3) defines necessary and sufficient conditions, $G$ has an SPS if and only if the system of constraints of Equation (7) has a solution. The system consists of so-called difference constraints [17], that can be turned into a constraint graph $(V, E)$, with edge weights $w(i, j) = P \cdot d(i, j) - e(i)$. According to [17], the system of constraints has a solution if and only if the constraint graph does not contain any cycles with negative accumulative weight, which implies that $G$ has an SPS if and only if the constraint graph does not have any cycles with negative accumulative weight.

Recall Equation (2), defining the MCM. It follows that:

$$\begin{aligned} & P \geq \mu(G) \\ \Leftrightarrow\ & P \geq \max_{c \in C(G)} \frac{\sum_{i \in N(c)} e(i)}{\sum_{(i,j) \in E(c)} d(i,j)} \\ \Leftrightarrow\ & 0 \geq \max_{c \in C(G)} \left(\sum_{i \in N(c)} e(i) - \sum_{(i,j) \in E(c)} P \cdot d(i, j)\right) \\ \Leftrightarrow\ & \min_{c \in C(G)} \left(\sum_{(i,j) \in E(c)} P \cdot d(i, j) - \sum_{i \in N(c)} e(i)\right) \geq 0. \end{aligned}$$

The last condition is equivalent to the condition that the constraint graph introduced above does not have any cycles with negative accumulative weight. Thus, $G$ has an SPS if and only if $P \geq \mu(G)$. ∎

Given an SRDF graph $G$ with MCM $\mu(G)$, $1/\mu(G)$ is the fastest possible firing rate of any actor in $G$ [15]. A schedule realizing this maximal throughput is *rate optimal*, where the throughput of a schedule is defined as the average number of firings of any actor over time. If an SPS has a period $P$ equal to the MCM $\mu(G)$, we say that this SPS is a *Static Periodic Rate-Optimal Schedule (SPROS)*. The example schedule given in the previous subsection for our running example is an SPROS. Theorem 1 means that an SPROS always exists. An SPS for graph $G$ can be constructed for any given period $P \geq \mu(G)$ by solving the system of constraints of Equation (7) in the proof of Theorem 1. A straightforward algorithm uses a shortest-path algorithm that can cope with negative weights, such as Bellman-Ford [15]. Thus, it is possible to construct an SPROS in polynomial time.

*Corollary 1:* A live SRDF graph has an SPROS, that can be constructed in polynomial time.

## III. The OSBA Problem

Ning and Gao formalize the Optimal Scheduling and Buffer Allocation (OSBA) problem in [1] as an Integer Programming (IP) problem. The following definition formalizes OSBA as an optimization problem.

**OSBA**

Given a live SRDF graph $G = (V, E, e, d)$, construct an SPROS for $G$ that has a buffer-capacity distribution function $b$ such that $\Sigma_{i \in V} b(i)$ is minimal among all buffer-capacity distribution functions for all SPROSs.

## IV. Ning and Gao's Approach to solve OSBA

As mentioned, Ning and Gao formalize OSBA in [1] as an IP problem. The IP formulation includes the sum of buffer capacities for a given schedule as an objective function, and the constraints necessary for an admissible schedule as in Equation (7) with period $P$ equal to the MCM of the graph. It furthermore contains a set of constraints implied by the limited buffer capacities[2]. Given an actor $i$ with buffer capacity $b(i)$, for every edge $(i, j)$, we have the constraint that

$$P \cdot b(i) + s(i) - s(j) \geq P \cdot (d(i, j) + 1) - 1. \quad (8)$$

These buffer-capacity constraints are only valid if firings of the same actor do not overlap. The derivation of these constraints can be found in [1]. The free variables in the integer program are the start times $s(i)$ of the first firings of all actors for an SPROS schedule $s$ and the buffer capacities $b(i)$.

**OSBA-IP** [1]

Let $G = (V, E, e, d)$ and $P = \mu(G)$.
Minimize $\sum_{i \in V} b(i)$
subject to
$\forall (i, j) \in E$, $P \cdot b(i) + s(i) - s(j) \geq P \cdot (d(i, j) + 1) - 1$,
$\forall (i, j) \in E$, $s(j) - s(i) \geq e(i) - P \cdot d(i, j)$,
$\forall i \in V$, $s(i), b(i)$ integers.

In general, (the decision variant of) the Integer Programming problem is NP-complete [18], [19]. The Linear Programming (LP) problem, dropping the requirement that the free variables should be integer, is known to be efficiently solvable. There are special classes of IP problems that can be solved as LP problems, because, for these classes, any solution of the LP problem is guaranteed to be integral. Ning and Gao transform the OSBA-IP problem into such a problem by means of a simple variable substitution. For every $i \in V$,

$$b'(i) = P \cdot b(i). \quad (9)$$

**OSBA-LP** [1]

Let $G = (V, E, e, d)$ and $P = \mu(G)$.
Minimize $\sum_{i \in V} b'(i)$
subject to
(a) $\forall (i, j) \in E$, $b'(i) + s(i) - s(j) \geq P \cdot (d(i, j) + 1) - 1$,
(b) $\forall (i, j) \in E$, $s(j) - s(i) \geq e(i) - P \cdot d(i, j)$.

The constraints are labeled to allow easy referencing.

Ning and Gao proceed to show that the constraint matrix of the OSBA-LP problem is unimodular, and that, because of this, the obtained solution is always integral. However, while OSBA-LP returns all-integer solutions, the $b(i)$ obtained by dividing the $b'(i)$ results by the period $P$ may be fractional. Ning and Gao solve this by discarding the $b(i)$ results and simply simulating the obtained SPROS (given by the values of the $s(i)$ variables obtained from OSBA-LP) to compute the actual buffer-capacity distribution function $b$ for $s$.

Since the MCM computation, needed as input for the OSBA-LP problem, the LP problem itself, and the buffer-capacity computation from the obtained SPROS can be solved in polynomial time, the proposed approach has polynomial complexity. Ning and Gao also claim optimality of the approach. However, their reasoning is flawed. OSBA-IP and OSBA-LP are truly different problems: while OSBA-IP requires the $b(i)$ to be integer, OSBA-LP requires instead the products $P \cdot b(i)$ to be integer. The next section shows that there are cases where the values of $s(i)$ for an optimal sum of integer $P \cdot b(i)$ (or, in other words, fractional $b(i)$) are different from the values of $s(i)$ for an optimal sum of integer $b(i)$.

## V. Counter-example for Ning and Gao's Optimality Claim

We first provide the counter-example, and then proceed to show where the reasoning of Ning and Gao is flawed.

### A. The Counter-example

OSBA-LP contains two sets of constraints, buffer constraints (a) and precedence constraints (b), both corresponding to the set of edges. The $b'(i)$ that should be minimized are only constrained by the set of buffer constraints (a). A crucial aspect of OSBA-LP is that it tries to maximize $s(i) - s(j)$ for all edges $(i, j)$ under the set of buffer constraints (a), because this leads to the minimal $b'(i)$, while the set of precedence constraints (b) constrains $s(i) - s(j)$ from above (which is clear when rewriting (b) into $s(i) - s(j) \leq P \cdot d(i, j) - e(i)$).

Consider again the example of Figure 1. This is a graph for which Ning and Gao's algorithm does not compute the optimal buffer sizes for rate-optimal scheduling. Table II summarizes the properties of the SRDF graph, the schedule $s_0$ resulting from OSBA-LP with corresponding buffer sizes $b_0$, a buffer-optimal schedule $s_1$ with buffer sizes $b_1$, and the minimum values of the free variables $b'_0$ and $b'_1$ for the two schedules computed from the buffer constraints of OSBA-LP, where the corresponding constraint is given as well. Note that only actor $A$ has more than one output edge. However, each of these edges results in the same constraint, which is only mentioned once in the table (in the row with first entry $(A, j)$). As a result, every actor has exactly one entry in the bottom part of the table. The bottom line gives the values for the OSBA-LP objective function and the total buffer requirements for the schedule. The example shows that the OSBA-LP schedule has a lower value for the objective function than the optimal schedule, but a higher buffer requirement. Differences between the OSBA-LP and optimal solutions are highlighted in italics. The problem that the OSBA-LP approximation has with this counter-example is caused by the scheduling freedom for actor $D$ within the SPROS, as the reasoning below clarifies.

[2]Subsequent papers, also by one of the authors of the original paper (see, e.g., [6], [7]), consider also a variant of OSBA in which every edge in an SRDF graph has its own buffer. This does not significantly change the IP formulation, and the counter-example for the optimality claim given in the next section applies to both versions.

TABLE II
COUNTER-EXAMPLE FOR NING AND GAO'S OPTIMALITY CLAIM.

| | | OSBA-LP | | Optimal | |
|---|---|---|---|---|---|
| $i$ | $e(i)$ | $s_0(i), P = 5$ | | $s_1(i), P = 5$ | |
| $A$ | 1 | 0 | | 0 | |
| $B$ | 1 | 1 | | 1 | |
| $C$ | 1 | 1 | | 1 | |
| $D$ | 1 | *2* | | *4* | |
| $E$ | 2 | 3 | | 3 | |
| $F$ | 4 | 1 | | 1 | |
| $G$ | 3 | 0 | | 0 | |
| $(i,j)$ | $d(i,j)$ | $b'_0(i)$ | $b_0(i)$ | $b'_1(i)$ | $b_1(i)$ |
| $(A,j)$ | 0 | $5-1 \geq 4$ | 1 | $5-1 \geq 4$ | 1 |
| $(B,D)$ | 0 | *$5-1 \geq 4$* | 1 | *$7-3 \geq 4$* | 1 |
| $(C,D)$ | 0 | *$5-1 \geq 4$* | 1 | *$7-3 \geq 4$* | 1 |
| $(D,E)$ | 1 | *$10-1 \geq 9$* | *2* | *$8+1 \geq 9$* | *1* |
| $(E,A)$ | 1 | $6+3 \geq 9$ | 1 | $6+3 \geq 9$ | 1 |
| $(F,G)$ | 1 | $8+1 \geq 9$ | 1 | $8+1 \geq 9$ | 1 |
| $(G,E)$ | 0 | $7-3 \geq 4$ | 1 | $7-3 \geq 4$ | 1 |
| | sum | *46* | *8* | *48* | *7* |

First, two observations imply that schedule $s_1$ of Table II is indeed a solution to the OSBA optimization problem.

1) The example graph has an MCM of 5, resulting from the $(A,F),(F,G),(G,E),(E,A)$ cycle. Since schedule $s_1$ has a period of 5, it is rate optimal.
2) It is straightforward to verify the buffer sizes for $s_1$. Consider for example actor $D$. Schedule $s_1$ results in a buffer size of 1. Initially, edge $(D,E)$ has one token, which is consumed by the first firing of $E$ at time 3. Furthermore, space for a token produced by $D$ is needed from the start of firing $k$ of $D$ at time $4+5k$, till the start of firing $k+1$ of $E$ at time $8+5k$ that consumes that token. So space for one token is needed in interval $[0,3]$ and in intervals $[4+5k, 8+5k]$, which do not overlap for different $k$. Hence, buffer size 1 suffices for actor $D$. Since buffer-capacity distribution function $b_1$ yields buffer sizes of 1 for all actors and size 1 is always minimal[3], $b_1$ gives the minimal buffer sizes among all rate-optimal schedules.

Second, it can be argued that OSBA-LP yields schedule $s_0$ as a solution.

3) Schedule $s_0$ is the schedule already given in Section II-D, where it was already mentioned that all actors fire as soon as they are enabled. This follows in a straightforward way from the precedence constraints in the example SRDF graph, corresponding to constraint set (b) in the OSBA-LP formulation.
4) An important consequence of the previous observation is that actor firings can only be delayed with respect to schedule $s_0$. Let us consider the effect of delaying an actor firing in $s_0$ on the values of $b'(i)$ in the OSBA-LP formulation. Consider for example a delay of 2 time units of firings of actor $D$, which in fact turns schedule $s_0$ into schedule $s_1$. Since $D$ has two input edges $(B,D)$ and $(C,D)$, the effect of this delay on the sum of the $b'(i)$ values is on the one hand a decrease of 2 for the $(D,E)$ edge and on the other hand an increase of $2 \cdot 2$ for the $(B,D)$ and $(C,D)$ edges, as illustrated by the italics constraints in the bottom part of Table II.
5) Reconsidering the example graph shows that there is no freedom in scheduling actors $A$, $F$, $G$, and $E$ without affecting the throughput (excluding a shift in time of the entire schedule, which is not meaningful); $B$ and/or $C$ can be delayed but only if $D$ is also delayed. A crucial observation now is that any group of actors from $B$, $C$, and $D$ whose firings might be delayed thus has only one outgoing edge, whereas it has two incoming edges. This means that the net effect of any delay of actor firings on the $b'(i)$ in OSBA-LP is always an increase, just as in the example given for delaying $D$ only. But this means that no delay of actor firings in $s_0$ can result in a better value of the objective function in OSBA-LP, and hence that $s_0$ is the optimal schedule according to OSBA-LP.

Finally, we compare the buffer requirements of $s_0$ and $s_1$.

6) Observation 2) above argues that the buffer sizes defined by $b_1$ are correct. In Section II-D and Figure 2, the buffer-capacity distribution function $b_0$ was already explained, and in particular it was already argued that the buffer space needed by schedule $s_0$ to store the data on edge $(D,E)$ is 2. This shows that $s_0$ needs a larger buffer size for actor $D$ than $s_1$ and, hence, has larger total buffering requirements than $s_1$.

The above reasoning shows that schedule $s_1$ is a solution to OSBA for the SRDF graph of Figure 1. It also shows that $s_0$ is the solution to OSBA for this graph according to the solution method proposed by Ning and Gao in [1], but that it is in fact not a solution to OSBA.

### B. Analysis of the Counter-example

It is interesting to consider potential causes for the sub-optimality of Ning and Gao's approach to solve OSBA. A first observation is that Ning and Gao observe in their paper that the buffer constraints $b(i) \geq (s(j) - s(i) - 1)/P + d(i,j) + 1$ obtained from Equation (8) are conservative approximations, in the sense that buffer capacities that satisfy those constraints are guaranteed to be sufficiently large. This suggests that it might be possible that an obtained SPROS uses less buffer capacity than the constraints specify. Schedules $s_0$ and $s_1$ with their buffer-capacity distribution functions $b_0$ and $b_1$ show that this is indeed possible. Consider for example edge $(F,G)$ of the SRDF graph. The start times of $F$ and $G$ and the buffer capacity assigned to $F$ are equal in both schedules. However, for these values, the above constraint on the buffer capacity yields $1 \geq 1\frac{3}{5}$, which is obviously not true. Thus, the buffer constraints of OSBA-IP *exclude* the solution to the OSBA optimization problem for the example graph of Figure 1, as well as the solution obtained via the LP approach of Ning and Gao themselves. In fact, also the obtained result for the running example that Ning and Gao use in [1] to illustrate their approach is excluded by the IP formulation of OSBA.

These observations may suggest that the IP formulation is the root cause of the sub-optimality of the approach of Ning and Gao. However, this is not the case. Ning and Gao do not further comment on the fact that their buffer constraints

[3]This is, in fact, only true for actors with non-zero execution times.

are not exact but only conservative, nor on the consequences for their approach. However, it is not difficult to derive exact constraints for the required buffer capacities, and study the consequences of using exact constraints.

Consider an actor $i$ of an SRDF graph with buffer capacity $b(i)$ and output edge $(i,j)$ with $d(i,j)$ initial tokens. The buffer has then, at most, $b(i)-d(i,j)$ empty places. (Note that, in line with [1], we do not assume that all output edges of an actor have the same number of initial tokens; some space in the buffer may be occupied by tokens still needed by other actors than $j$ consuming data produced by $i$.) Consider firing $k+1$ of actor $i$, which in an SPS with period $P$ starts at time $s(i)+P\cdot k$. Due to the limited buffer capacity, this firing can only take place if actor $j$ has started sufficiently many firings to provide space, i.e., actor $j$ should start firing $(k+1)-(b(i)-d(i,j))$, which occurs at time $s(j)+P\cdot(k-(b(i)-d(i,j)))$, no later than the start of firing $k+1$ of actor $i$. Thus,

$$\begin{aligned} & s(j)+P\cdot(k-(b(i)-d(i,j)))\leq s(i)+P\cdot k \\ \Leftrightarrow\ & P\cdot b(i)+s(i)-s(j)\geq P\cdot d(i,j). \end{aligned} \tag{10}$$

This last inequality differs in its right-hand side from the constraint of Equation (8). This right-hand side is never greater than the right-hand side in Equation (8). Thus, the following IP formulation of OSBA, combining the precedence constraints of OSBA-IP with the exact buffer constraints, allows strictly more solutions than OSBA-IP.

**OSBA-eIP**

Let $G=(V,E,e,d)$ and $P=\mu(G)$.

| | |
|---|---|
| Minimize | $\sum_{i\in V} b(i)$ |
| subject to | |
| $\forall(i,j)\in E,$ | $P\cdot b(i)+s(i)-s(j)\geq P\cdot d(i,j),$ |
| $\forall(i,j)\in E,$ | $s(j)-s(i)\geq e(i)-P\cdot d(i,j),$ |
| $\forall i\in V,$ | $s(i), b(i)$ integers. |

We can now investigate what happens if we follow the approach of [1], turning OSBA-eIP into an LP via the variable substitution of Equation (9), and computing buffer capacities from the SPROS obtained from that LP. Consider the example of Table II. The crucial observation is that, when using the exact buffer constraints, the right-hand sides of the constraints in the bottom half of the table are all reduced by four. This implies that also the $b'_0(i)$ and $b'_1(i)$ are all reduced by four. Thus, essentially, nothing has changed. SPROS $s_0$, with buffer capacities $b_0$, is still the solution found by the LP-based approach, whereas the optimal solution is $s_1$, with buffer capacities $b_1$ (which now satisfy the IP constraints). This leads to the already mentioned conclusion that the reasoning of Ning and Gao is flawed in the assumption that the schedule corresponding to an optimal sum of integer $P\cdot b(i)$ always results in an optimal sum of integer $b(i)$. This assumption is not correct, neither for the original approximate IP formulation nor for the exact IP formulation given in this subsection.

## VI. The Generalized OSBA Problem

This section introduces a generalized version of OSBA, referred to as gOSBA, which is not only interesting in itself, but also forms the basis for the NP-completeness proof for the original OSBA formulation. After defining gOSBA and showing that it always has a solution, we proceed with proving NP-completeness of gOSBA.

### A. Problem Definition

Our generalized OSBA problem makes more conservative buffering assumptions than the original formulation. The following definition of a buffer-capacity distribution function differs from the one given in Equation (5) in that it assigns a separate buffer to each edge, and that it reserves space in each buffer for an active firing of the consuming actor. This definition captures the most conservative buffering requirements that are possible.

$$\begin{aligned} B(i,j) = \max_{t\in\mathbb{N}_0\cup\{-1\}} & \\ & (c(i,j,t) \\ & +|\{k\in\mathbb{N}_0 \mid s(i,k)\leq t<f(i,k)\}| \\ & +|\{k\in\mathbb{N}_0 \mid s(j,k)\leq t<f(j,k)\}|). \end{aligned} \tag{11}$$

The generalized OSBA problem now differs from the original OSBA problem in the sense that it assumes these most conservative buffer requirements, but also in the sense that it requests a total buffer size that is minimal among all rate-optimal schedules, not necessarily only among periodic ones. This version of OSBA is interesting in, for example, multiprocessor applications in which sharing of buffers among multiple data edges in an SRDF graph may not be possible and in situations where an actor produces different data for different successor actors. This makes it a relevant problem for, for example, modern multimedia applications implemented on multiprocessor systems-on-chip (see, e.g., [3]).

**gOSBA**

Given a live SRDF graph $G=(V,E,e,d)$, construct an SPROS for $G$ that has a buffer-capacity distribution function $B$ such that $\Sigma_{(i,j)\in E}B(i,j)$ is minimal among all buffer-capacity distribution functions for all rate-optimal (not necessarily periodic) schedules.

Schedule $s_1$ of Table II turns out to be a solution to gOSBA for the running example. It requires a buffer of size one for each of the edges in the graph, except for edge $(F,G)$ which needs a size of 2, leading to a total buffer requirement of 10. From the execution times of 4 and 3 for actors $F$ and $G$ respectively, it follows that in any rate-optimal schedule with period $\mu(G)=5$, firings of $F$ and $G$ must necessarily overlap. This means that the conservative assumptions captured in Equation (11) result in a minimal size of 2.

### B. gOSBA always has a Solution

It is not entirely trivial that gOSBA always has a solution, i.e., that it is always possible to construct an SPROS with buffer sizes that are minimal *among all rate-optimal schedules*. In other words, is it possible to achieve periodicity and buffer minimality simultaneously? Using the so-called

capacity-constrained SRDF model of an SRDF graph with a given buffer-capacity distribution function (see, e.g., [11]), it can be proven that gOSBA always has a solution.

The capacity-constrained model $G_{\text{ccm}}$ of an SRDF graph $G$ with buffer-capacity distribution function $B$ is itself an SRDF graph that is obtained from $G$ by adding, for every edge $(i, j)$, a reverse edge $(j, i)$ with $B(i, j) - d(i, j)$ initial tokens. (This may in fact turn the graph into a multi-graph. All the results in this paper generalize to multi-graphs in a trivial way.) Given an edge $(i, j)$, a reverse edge $(j, i)$ in $G_{\text{ccm}}$ captures precisely the remaining buffer space for edge $(i, j)$. The start of a firing of $i$ claims space by consuming a token from edge $(j, i)$; the end of a firing of actor $j$ releases space in the buffer of $(i, j)$ by producing a token on $(j, i)$. This is in line with the conservative buffer assumptions captured by buffer-capacity distribution function $B$. As a result, any schedule of $G$ that never uses more buffer space on any edge than allowed by $B$, is also a schedule of $G_{\text{ccm}}$. Theorem 1 then yields the desired result that gOSBA always has a solution.

*Theorem 2:* Consider a live SRDF graph $G = (V, E, e, d)$ with a buffer-capacity distribution function $B$ such that $\Sigma_{(i,j)\in E} B(i, j)$ is minimal among all buffer-capacity distribution functions for all rate-optimal schedules. Then, $G$ has an SPROS with buffer-capacity distribution function $B$.

*Proof:* Since $G$ allows a rate-optimal schedule with buffer-capacity distribution function $B$, by Theorem 1, the capacity-constrained model $G_{\text{ccm}}$ derived from $G$ and $B$ has an SPROS. By the construction of $G_{\text{ccm}}$, this SPROS respects the buffer capacities specified by $B$. Since $B$ is such that $\Sigma_{(i,j)\in E} B(i, j)$ is minimal among all buffer-capacity distribution functions for all rate-optimal schedules, it follows that the SPROS has buffer-capacity distribution function $B$. ■

*Corollary 2:* gOSBA always has a solution.

The buffer-sizing techniques of [11] and [12], that are efficient in practice, can be used to compute buffer sizes that are minimal under the conservative assumptions of Equation (11) while allowing a rate-optimal schedule of a given SRDF graph. The capacity-constrained model can then be used to construct an SPROS with these buffer sizes in the way explained in Section II-E.

### C. The Constraint Graph

Our NP-completeness proof for gOSBA uses a constraint representation of gOSBA, which we introduce in this subsection. Corollary 2 shows that it is allowed to limit our attention to SPROSs when solving gOSBA, making it in this respect similar to OSBA. As we have seen, the OSBA problem can be captured by two types of linear constraints, the data precedence constraints introduced in Equation (7), and the buffer constraints of Equation (10). (The buffer constraints of Equation (8) cannot be used, because these are not exact.) Also gOSBA can be captured via two sets of constraints, that are very similar to the mentioned constraints. The precedence constraints, labeled (b) below, are actually identical. The buffer-capacity constraints can be derived in the same way as those in Section V-B, while taking into account the separate buffers per output edge of an actor and the fact that input buffer space is only released at the end of a firing. These two aspects explain the occurrence of the $B(i, j)$ and the $e(j)$ in the right-hand side of the buffer constraints given as (a) below. For any given SRDF graph $G = (V, E, e, d)$, given period $P$, and buffer-capacity distribution function $B$, the following system of constraints captures gOSBA. For all $(i, j) \in E$,

$$\begin{aligned} &\text{(a)} \quad s(j) - s(i) \leq P \cdot (B(i, j) - d(i, j)) - e(j), \\ &\text{(b)} \quad s(i) - s(j) \leq P \cdot d(i, j) - e(i). \end{aligned} \tag{12}$$

This system is a system of *difference* constraints [17], i.e., a system of linear constraints where each constraint is a maximum difference between two variables. As shown in [17], a system of difference constraints can be turned into a constraint graph that has a node for every variable and an edge $(u, v)$ with edge weight $m$ for any constraint of the form $v - u \leq m$.[4] As a consequence, it is possible to create a constraint graph for any given SRDF graph $G$, period $P$, and buffer-capacity distribution function $B$. Figure 3 shows the constraint graph for our example SRDF graph of Figure 1, period $P = 5$, and the optimal buffer-capacity distribution function resulting from schedule $s_1$ of Table II, as discussed at the end of Section VI-A.

Before formally defining the constraint graph, it should be noted that the set of constraints of Equation (12) may contain redundant constraints. If actors $i$ and $j$ of the SRDF graph are connected by edges $(i, j)$ and $(j, i)$, there are two constraints of the form $s(i) - s(j) \leq m$, one precedence constraint and one buffer-capacity constraint, and two constraints of the form $s(j) - s(i) \leq n$, as well. In both cases, the constraint with the largest right-hand side is redundant. The constraint graph is constructed after removal of these redundant constraints. Let $c_p(i, j)$ refer to any precedence constraint corresponding to edge $(i, j)$ in $E$, i.e., a constraint of type (b) above, and let $c_b(i, j)$ refer to the buffer constraint (a) corresponding to edge $(i, j)$. We use $w_p(i, j)$ resp. $w_b(i, j)$ to refer to the right-hand sides of $c_p(i, j)$ and $c_b(i, j)$, and assume that $w_p(i, j)$ and $w_b(i, j)$ are $\infty$ if the corresponding constraint is not present. Constraint graph $C(G, P, B)$ is then the weighted graph $(V_C, E_C, w_C)$, with $w_C : E_C \to \mathbb{Z}$, defined as follows:

$$\begin{aligned} V_C &= \{s(i) \mid i \in V\}, \\ E_C &= \{(s(i), s(j)) \mid (i, j) \in E \vee (j, i) \in E\}, \\ w_C &= \{(e, w_b(i, j) \min w_p(j, i)) \mid e = (s(i), s(j)) \in E_C\}. \end{aligned} \tag{13}$$

The following proposition is important later.

*Proposition 1 ([17]):* A set of difference constraints has a solution if and only if the corresponding constraint graph has no cycles with negative accumulative weight.

The constraint graph of Figure 3 does not have cycles with negative accumulative weight. This conforms to the fact that the system of difference constraints of Equation (12) has solutions, schedule $s_1$ of Table II being one of them.

[4]The constraint graph as defined in [17] has one extra auxiliary source node and some extra edges originating from this source node, which are not relevant for our purposes, and hence ignored in the remainder.

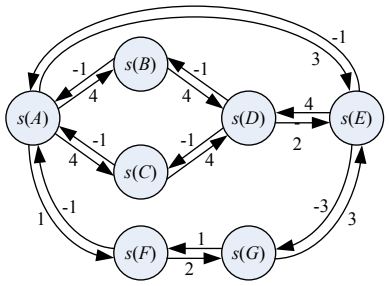


Fig. 3. The constraint graph for our example SRDF graph, period $P = 5$, and gOSBA-optimal buffer-capacity distribution function.

### D. Minimal Buffering for Live Execution

In his dissertation [20], Murthy proves NP-completeness of another buffer optimization problem for dataflow graphs. Murthy's graphs are equivalent to our SRDF graphs, except that our actors have a time valuation. We re-formulate the problem of [20] as a decision problem in the setting of this paper, and refer to it as the Minimal Buffering for Live Execution (MBLE) problem. We use MBLE to prove that gOSBA is NP-complete. The NP-completeness proof for Murthy's MBLE formulation carries over trivially to the current setting. A schedule is sequential if no two actor firings overlap in time.

**MBLE**

Given a live SRDF graph $G = (V, E, e, d)$ with $e(i) = 1$ for all $i \in V$ and a positive integer $K$, does $G$ have a sequential SPS that has a buffer-capacity distribution function $B$ such that $\Sigma_{(i,j)\in E} B(i,j) \leq K$?

*Theorem 3 ([20]):* MBLE is NP-complete.

Murthy gives a reduction from the feedback-arc-set (FAS) problem [19], which for a given directed graph essentially asks for a subset of edges of a given size that breaks all cycles in the graph. Given a FAS graph, the reduction creates an SRDF graph by, for the sake of reasoning, reversing the edges in the FAS graph and by adding one initial token to each of these edges. The crucial observation is then that a sequential SPS for an SRDF graph with precisely one token on every edge has a buffer size of 2 for any given edge if the source actor of the edge is scheduled before the sink and 1 otherwise.

The following property is needed in our complexity proof.

*Lemma 1:* For any live SRDF graph $G = (V, E, e, d)$ with $e(i) = 1$ for all $i \in V$, MCM $\mu(G) \leq |V|$.

*Proof:* The sum of actor execution times is $\sum_{i\in V} e(i) = |V|$. Thus, no cycle has a sum of execution times greater than $|V|$. Since $G$ is live, the minimal amount of tokens in any cycle is one. The maximum possible cycle mean is from a cycle with maximum execution time and minimum number of initial tokens. This is thus bounded by $|V|/1$. ■

### E. Complexity of gOSBA

To reason about the complexity of gOSBA, we first formulate it as a decision problem.

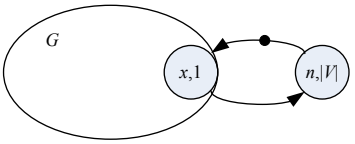


Fig. 4. The reduction from MBLE to gOSBA-D.

**gOSBA-D**

Given a live SRDF graph $G = (V, E, e, d)$ and a positive integer $K$, does $G$ have an SPROS with buffer-capacity distribution function $B$ such that $\Sigma_{(i,j)\in E} B(i,j) \leq K$?

To prove NP-completeness of gOSBA-D, we create an equivalent instance of gOSBA-D from any instance of MBLE. Assume an MBLE instance with $G = (V, E, e, d)$ and positive integer $K$. The equivalent gOSBA-D instance $G_g = (V_g, E_g, e_g, d_g)$ with positive integer $K_g = K + 2$ is created by taking $G$, choosing an arbitrary node $x \in V$, and adding a new node $n \notin V$ as illustrated in Figure 4 and formalized as follows:

$$
\begin{aligned}
V_g &= V \cup \{n\};\\
E_g &= E \cup \{(x,n),(n,x)\};\\
e_g &= e \cup \{(n,|V|)\};\\
d_g &= d \cup \{((n,x),1),((x,n),0)\}.
\end{aligned}
$$

The idea behind the transformation is that the new actor creates a sufficiently long critical cycle, implying (1) that an MBLE solution extended for the new actor is an SPROS of the resulting gOSBA graph with, except for the new edges, the same buffer capacities, and (2) that any SPROS of the gOSBA graph can be sequentialized to form a solution to the original MBLE instance. The reduction from MBLE to gOSBA-D cannot be applied directly to the original OSBA formulation because sequentializing an SPS may adversely affect the required buffer capacities under the buffering assumptions made by Ning and Gao, whereas it does not under the more conservative gOSBA assumptions.

The fact that $K_g = K+2$ in the gOSBA-D instance follows from the observation that the buffer capacity for edges $(n, x)$ and $(x, n)$ is one in any admissible schedule of $G_g$.

*Corollary 3:* If $B_g$ is a buffer-capacity distribution function for an admissible schedule of gOSBA-D instance $G_g$, then $B_g(n, x) = B_g(x, n) = 1$.

We proceed with two lemmas needed in the NP-completeness proof.

*Lemma 2:* The MCM $\mu(G_g)$ of $G_g$ is $|V| + 1$.

*Proof:* Lemma 1 applies to MBLE graph $G$, implying that $\mu(G) \leq |V|$. By construction, the critical cycle in $G_g$ is therefore the cycle $(n, x), (x, n)$ with cycle mean $|V| + 1$. ■

*Lemma 3:* If buffer-capacity distribution $B$ results from a solution of MBLE instance $G$, then constraint graph $C(G, P, B)$ does not have negative cycles for any $P \geq |V|$.

*Proof:* By definition of MBLE, $G$ has a sequential SPS that respects buffer capacities $B$. Since all actors of $G$ have an execution time of one, their sequential execution takes $|V|$ time units. Thus, an SPS respecting $B$ with period $|V|$ exists, and therefore there exists also an SPS respecting $B$ for any $P \geq |V|$. It follows that the system of difference constraints of Equation (12) has a solution for such a $P$ and $B$, which in turn implies the desired result based on Proposition 1. ■

*Theorem 4:* gOSBA-D is NP-complete.

*Proof:* It is straightforward to see that gOSBA-D is in NP: If we have a proposed solution $B$ for gOSBA-D instance $G$, we can build the constraint graph $C(G, \mu(G), B)$ of $G$ with period $\mu(G)$ and buffer-capacity distribution $B$. A search for negative cycles in $C(G, \mu(G), B)$ can be done polynomially [17]. If no negative cycles are found, then, according to Proposition 1, the set of constraints of Equation (12) has a solution, showing that $B$ is a solution of gOSBA-D.

For the proof of NP-hardness, we show that MBLE is reducible to gOSBA-D. Let $G = (V, E, e, d)$, with positive integer $K$ be an MBLE instance. Let $G_g$ with $K_g = K + 2$ be the corresponding gOSBA-D instance as defined above.

First, if buffer-capacity distribution function $B$ with $\Sigma_{(i,j)\in E} B(i,j) \leq K$ is a solution of the MBLE instance, then buffer-capacity distribution $B_g = B \cup \{((n,x),1),((x,n),1)\}$ is a solution of the corresponding gOSBA-D instance, as the following shows. By Lemma 3, constraint graph $C(G, |V|+1, B)$ does not have negative cycles. Thus, by Proposition 1, there is an SPS $s$ for $G$ with period $\mu(G_g) = |V| + 1$ that respects $B$. As a consequence, $s_g = s \cup \{(n, s(x) + 1)\}$ is an SPS for $G_g$, that by Corollary 3 and Lemma 2 has buffer-capacity distribution $B_g$ and is rate-optimal, showing that $s_g$ and $B_g$ form a solution of the gOSBA-D instance $G_g$.

Second, if schedule $s$ with buffer-capacity distribution function $B_g$ with $\Sigma_{(i,j)\in E_g} B_g(i,j) \leq K_g$ is a solution of the gOSBA-D instance, then buffer-capacity distribution function $B = B_g \setminus \{((n,x), B_g(n,x)), ((x,n), B_g(x,n))\}$ is a solution of the corresponding MBLE instance, as the following shows. Observe that $B$ respects the bound $K$ of the MBLE instance by Corollary 3. Since $s$ and $B_g$ are a solution of the gOSBA-D instance, the system of constraints of Equation (12) has $s$ as a solution and by Proposition 1 constraint graph $C(G_g, \mu(G_g), B_g)$ has no negative cycles. If we remove node $s(n)$ with its input and output edge from this graph, we obtain constraint graph $C(G, \mu(G_g), B)$. Since we only removed a node and two edges from $C(G_g, \mu(G_g), B_g)$, no new cycles have been created, and so $C(G, \mu(G_g), B)$ does not have negative cycles. Again using Proposition 1, this means that $G$ has an SPS with period $\mu(G_g)$ that respects buffer capacities $B$. Since by Lemma 2 $\mu(G_g) = |V| + 1$, and since all actor execution times in $G$ are one, this SPS can be sequentialized without affecting the period and without negatively affecting the required buffer capacities. The latter follows from a simple inductive reasoning. Thus, the resulting sequential SPS is a solution of the MBLE instance.

This shows that MBLE is reducible to gOSBA-D. Since gOSBA-D is NP-hard and in NP, it is NP-complete. ■

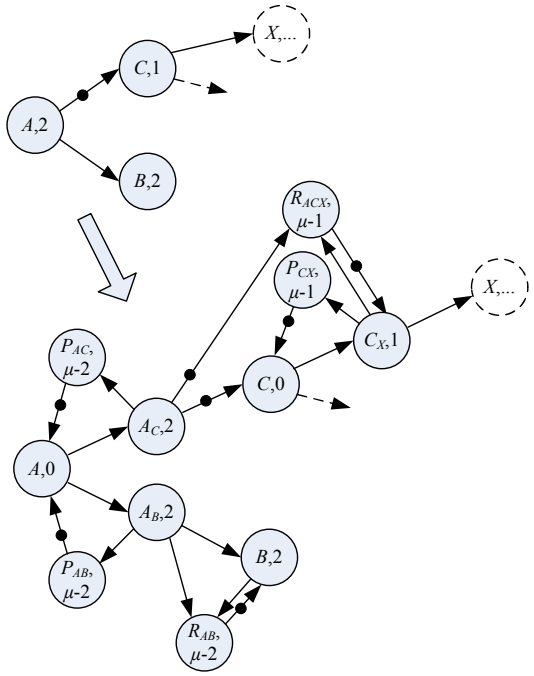


Fig. 5. Reduction from gOSBA to OSBA.

## VII. Complexity of OSBA

To prove the NP-completeness of the original OSBA formulation, we first phrase OSBA as a decision problem.

**OSBA-D**

Given a live SRDF graph $G = (V, E, e, d)$ and a positive integer $K$, does $G$ have an SPROS with buffer-capacity distribution function $b$ such that $\Sigma_{i\in V} b(i) \leq K$?

We show that gOSBA-D can be reduced to OSBA-D, thus showing that OSBA-D is NP-complete. Figure 5 illustrates the reduction.

The basic idea is that any SPROS of the gOSBA-D instance, say $G_g$, has a one-to-one correspondence with an SPROS of the OSBA-D instance, say $G_O$, while every buffer needed by $G_g$ has a one-to-one correspondence with a buffer of $G_O$ and all other buffers in $G_O$ have a fixed size for all SPROSs. Assume that $G_g$ has MCM $\mu$ and consider some SPROS, which thus has period $\mu$.

First, consider actor $A$ in the example of Figure 5. Actor $A$ has multiple output edges, which in gOSBA-D means that the two edges have separate buffers whereas the two edges in OSBA-D would share a buffer. To mimic gOSBA-D, we have to create two separate buffers in the OSBA-D instance $G_O$. To this end, actor $A$ with execution time 2 is replaced by actor $A$ with execution time 0, and any output edge $(A, j)$ (e.g., $(A, B)$ in the figure) is replaced by two actors, actor $A_j$ that inherits the execution time, 2, from $A$, and actor $P_{Aj}$ that gets execution time $\mu - 2$, plus four edges $(A, A_j)$, $(A_j, P_{Aj})$, $(P_{Aj}, A)$, and $(A_j, j)$. In a rate-optimal schedule, which has period $\mu$, the cycle through $A$, $A_j$, and $P_{Aj}$ enforces that the firings of these three actors occur as soon as the firing

of the preceding actor finishes. As a consequence, the buffers of $A$ and $P_{Aj}$ in $G_O$ are 0 and (due to the initial token) 1 respectively[5]. As another consequence, the firing of $A_j$ in $G_O$ completes at the same point in time as the corresponding firing of $A$ completes in $G_g$. Edge $(A_j, j)$ and the buffer of actor $A_j$ now take the role of edge $(A, j)$ with its buffer in $G_g$.

Second, consider actor $B$ with its input edge $(A, B)$. In gOSBA-D, actor $B$ releases input space at the end of its firing. In OSBA-D, $B$ would release input space at the firing start. To mimic the gOSBA-D behavior, actor $B$ is embedded in a cycle of length $\mu$ with actor $R_{AB}$. As in the previous construct, the firings of these two actors occur in a rate-optimal execution as soon as the firing of the preceding actor finishes. Thus, in a rate-optimal schedule, actor $R_{AB}$ starts at the same time as actor $B$ finishes. By adding now an edge $(A_B, R_{AB})$, which in OSBA-D shares its buffer with edge $(A_B, B)$, space in this buffer in $G_O$ is now only released at the *start* of a firing of $R_{AB}$, which corresponds exactly with the release of space in $G_g$ at the *end* of the corresponding firing of $B$.

In summary, the two sketched constructs address precisely the two differences between gOSBA-D and OSBA-D. Of course, they can also occur in combination, and edges may contain initial tokens, as illustrated for actor $C$ and edge $(A, C)$ in the figure. The initial token on $(A, C)$ is copied to both the corresponding $(A_C, C)$ and $(A_C, R_{ACX})$ edges. The remainder of this section gives a formal NP-completeness proof for OSBA-D based on the sketched transformation.

Consider gOSBA-D instance $G_g = (V_g, E_g, e_g, d_g)$ with positive integer $K_g$. OSBA-D instance $G_O = (V_O, E_O, e_O, d_O)$ with positive integer $K_O = K_g + |E_g| + |\{(i,j,k) \mid (i,j), (j,k) \in E_g\}|$ is created as follows. To simplify notations and reasoning, we apply the first construct sketched above to all edges in the graph, not only to those that are an output edge of an actor with multiple output edges. This transformation therefore affects all actors, except those without any output edges. Assume that $V_g$ is partitioned into actors with output edges $V_g^+$ and actors without output edges $V_g^-$.

$$
\begin{aligned}
V_O = {} & V_g \cup \cup_{(i,j)\in E_g} \{i_j, P_{ij}\} \cup \\
& \{R_{ij} \mid (i,j) \in E_g, j \in V_g^-\} \cup \\
& \{R_{ijk} \mid (i,j),(j,k) \in E_g\}; \\
E_O = {} & \cup_{(i,j)\in E_g} \{(i,i_j),(i_j,P_{ij}),(P_{ij},i),(i_j,j)\} \cup \\
& \cup_{(i,j)\in E_g, j\in V_g^-} \{(j,R_{ij}),(R_{ij},j),(i_j,R_{ij})\} \cup \\
& \cup_{(i,j),(j,k)\in E_g} \{(j_k,R_{ijk}),(R_{ijk},j_k),(i_j,R_{ijk})\}; \\
e_O = {} & \{(i,0) \mid i \in V_g^+\} \cup \{(i, e_g(i)) \mid i \in V_g^-\} \cup \\
& \cup_{(i,j)\in E_g} \{(i_j, e_g(i)), (P_{ij}, \mu(G_g) - e_g(i))\} \cup \\
& \{(R_{ij}, \mu(G_g) - e_g(j)) \mid (i,j) \in E_g, j \in V_g^-\} \cup \\
& \{(R_{ijk}, \mu(G_g) - e_g(j)) \mid (i,j),(j,k) \in E_g\}; \\
d_O = {} & \cup_{(i,j)\in E_g} \{((i,i_j),0),((i_j,P_{ij}),0),((P_{ij},i),1), \\
& \qquad ((i_j,j),d_g(i,j))\} \cup \\
& \cup_{(i,j)\in E_g, j\in V_g^-} \{((j,R_{ij}),0),((R_{ij},j),1), \\
& \qquad ((i_j,R_{ij}),d_g(i,j))\} \cup \\
& \cup_{(i,j),(j,k)\in E_g} \{((j_k,R_{ijk}),0),((R_{ijk},j_k),1), \\
& \qquad ((i_j,R_{ijk}),d_g(i,j))\}.
\end{aligned}
$$

[5]Buffer size 0 may seem artificial, but it is caused by the fact that we only consider buffer requirements after all firing starts and ends at a certain moment in time have occurred. The size 0 is in fact convenient in the reduction.

In the remainder of this subsection, we assume for simplicity that every actor has an output edge. The reasoning for actors without an output edge is simpler and goes along the same lines.

First, we observe that $G_g$ and $G_O$ have the same MCM.

*Lemma 4:* $\mu(G_O) = \mu(G_g)$.

*Proof:* All the cycles of $G_g$ are still present in $G_O$ with the same total execution time and the same number of initial tokens but with for any edge $(i,j) \in E_g$ the extra $i_j$ actors inserted. This means that $\mu(G_O) \geq \mu(G_g)$. The transformation from $G_g$ to $G_O$ adds cycles $(i,i_j),(i_j,P_{ij}),(P_{ij},i)$ with execution time $\mu(G_g)$ and one initial token. It furthermore adds cycles $(j_k,R_{ijk}),(R_{ijk},j_k)$ with execution time $\mu(G_g)$ and one initial token. None of these cycles causes $\mu(G_O)$ to be strictly larger than $\mu(G_g)$. Finally, for every cycle of $G_g$ going through edges $(i,j),(j,k) \in E_g$, the transformation adds a cycle with extra actor $R_{ijk}$ through edges $(i_j,R_{ijk})$ and $(R_{ijk},j_k)$. However, the additional execution time $\mu(G_g) - e_g(j)$ is compensated by one extra token, on the $(R_{ijk},j_k)$ edge, which means that also these extra cycles do not increase the MCM. Hence, $\mu(G_O) = \mu(G_g)$. ■

The following lemma shows that, given the actor firing times of the actors in $V_g$ in any SPROS of $G_O$, the firing times of all the additional actors are fixed. It also shows that, except for the buffers needed for the $i_j$ actors, all buffer sizes are fixed and independent of the particular SPROS.

*Lemma 5:* Let $s_O$ be an SPROS of $G_O$ with buffer-capacity distribution function $b_O$. For edges $(i,j),(j,k) \in E_g$,

1) $s_O(i_j) = s_O(i)$ and $s_O(P_{ij}) = s_O(i) + e_g(i)$;
2) $s_O(j_k) = s_O(j)$ and $s_O(P_{jk}) = s_O(R_{ijk}) = s_O(j) + e_g(j)$;
3) $b_O(i) = b_O(j) = 0$ and $b_O(P_{ij}) = b_O(P_{jk}) = b_O(R_{ijk}) = 1$.

*Proof:* Properties 1 and 2 follow from the actor execution times of $G_O$ and the observations that, by Lemma 4, $s_O$ has period $\mu(G_g)$, that the $i, i_j, P_{ij}$ actors are on a cycle of length $\mu(G_g)$ with an initial token on edge $(P_{ij},i)$, that the $j, j_k, P_{jk}$ actors are on a cycle of length $\mu(G_g)$ with an initial token on edge $(P_{jk},j)$, and that the $j_k$ and $R_{ijk}$ actors are on a cycle of length $\mu(G_g)$ with an initial token on edge $(R_{ijk},j_k)$. Property 3 then follows directly from Equation (5) and the derived relations between the various actor start times. ■

The following two propositions prove a one-to-one correspondence between SPROSs of $G_g$ and $G_O$ and their buffer sizes.

*Proposition 2:* A schedule $s_g$ is an SPROS of $G_g$ if and only if $s_O$ with, for all actors $i \in V_g$, $s_O(i) = s_g(i)$ and the start times of the other actors of $G_O$ as in Lemma 5, is an SPROS of $G_O$.

*Proof:* This result follows immediately from Lemma 5 and the observation that the finishing time $f_g(i)$ for any actor

in $V_g$ is identical to the finishing time $f_O(i_j)$ for the extra actors $i_j$ in $G_O$. As a result, all tokens on edges $(i_j, j)$ in $G_O$ become available at the same moment in time as corresponding tokens on edges $(i, j)$ in $G_g$. ■

*Proposition 3:* Consider two corresponding SPROSs $s_g$ of $G_g$ and $s_O$ of $G_O$. If $s_g$ has buffer-capacity distribution function $B_g$, then $s_O$ has buffer-capacity distribution function $b_O$ with, for any edge $(i, j) \in E_g$, $b_O(i_j) = B_g(i, j)$ and the buffer capacities of the other actors of $G_O$ as in Lemma 5.

*Proof:* Given Lemma 5, the only remaining proof obligation is to show that $b_O(i_j) = B_g(i, j)$, for any edge $(i, j) \in E_g$. First, the number of initial tokens on edge $(i, j)$ in $G_g$ is the same as the maximal number of initial tokens on any of the output edges of actor $i_j$ in $G_O$. Second, because the finishing time $f_g(i)$ for any actor in $V_g$ is identical to the finishing time $f_O(i_j)$ for the $i_j$ actors in $G_O$, tokens on edges $(i_j, j)$ and $(i_j, R_{ijk})$ in $G_O$ become available at the same moment in time as corresponding tokens on edge $(i, j)$ in $G_g$. Third, the latest consumption of any of the tokens produced by a firing of $i_j$ on its output edges to actors $P_{ij}$, $R_{ijk}$, $j$, and any actors $R_{hij}$ for input edges $(h, i)$ is performed by actors $R_{ijk}$. The $P_{ij}$ and $R_{hij}$ actors consume the produced tokens at the same time as they are produced; if actor $j$ consumes the token at time $t$, then all the $R_{ijk}$ actors consume the corresponding tokens at time $t + e_g(j)$. Thus, the tokens occupy space in the buffer $b_O(i_j)$ till time $t + e_g(j)$. Finally, since start times of actors $j$ in $G_g$ and $G_O$ correspond and according to Equation (11) the space for a token in buffer $B_g(i, j)$ is released at time $t + e_g(j)$ if $j$ starts a firing at time $t$, the correspondence $b_O(i_j) = B_g(i, j)$ follows. ■

*Theorem 5:* OSBA-D is NP-complete.

*Proof:* The transformation from $G_g$ to $G_O$ is polynomial. Furthermore, in line with the argument in the proof of Theorem 4, a proposed buffer-capacity distribution function for $G_O$ can be verified in polynomial time by building a constraint graph based on the OSBA precedence and buffer constraints as given in the OSBA-eIP problem formulation. Finally, Propositions 2 and 3 show that gOSBA-D has a solution with total buffer requirements $K_g$ if and only if OSBA-D has a solution with total buffer requirements $K_g + |E_g| + |\{(i, j, k) \mid (i, j), (j, k) \in E_g\}|$. The result then follows from Theorem 4. ■

## VIII. An Exact Solution

### A. The Solution

Our exact solution to OSBA, with exponential complexity, is based on the throughput-buffering trade-off analysis technique of [11], [12] for multi-rate and cyclo-static dataflow graphs. SRDF graphs are a subclass of these graph types. The technique explores the trade-off space between throughput and buffer requirements by iteratively executing a dataflow graph while computing throughput that can be obtained with given buffer sizes. The exploration starts from buffers of size 0 and then recursively increases buffers that potentially prevent a throughput improvement. An actor is said to have a *storage dependency* if its firing depends on the availability of space in some buffer. Such a buffer is then increased and the dataflow graph is re-evaluated with the increased buffer. In this way, the smallest buffers allowing a rate-optimal schedule are found. In order to apply the technique in the current setting, two small adaptations are needed. The technique as presented in [11], [12] assumes separate buffers for all output edges of an actor, whereas OSBA assumes a shared buffer. The notion of a storage dependency and the computation of total buffer requirements from a given execution need to be adapted to reflect this. However, these adaptations do not affect the correctness argument given in [12].

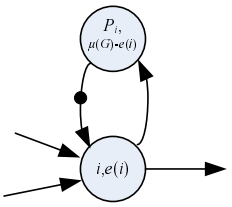


Fig. 6. An SPROS-preserving transformation.

Unfortunately, the result of the sketched analysis are the minimal buffer sizes among all rate-optimal schedules, not necessarily SPROSs. To the best of our knowledge, under the buffering assumptions of OSBA, it is an open problem whether the minimal buffer sizes among all rate-optimal schedules can also be realized with an SPROS. Note that Section VI-B, via Theorem 1 and a capacity-constrained model, shows that buffer minimality and static periodicity can be obtained simultaneously for gOSBA. However, we have not been able to develop a similar capacity-constrained model for OSBA. Nevertheless, the technique of [11], [12] forms the basis of an exact solution to OSBA, by applying it on a transformed graph that preserves SPROSs with their buffer requirements.

Given a graph $G$, let $G_{\text{spros}}$ be the SRDF graph obtained by adding for each actor $i$ an actor $P_i$ with execution time $\mu(G) - e(i)$ and two edges $(i, P_i)$ and $(P_i, i)$, the latter with one initial token. The transformation is illustrated in Figure 6. Clearly, the MCM of $G_{\text{spros}}$ is $\mu(G)$.

Consider now an SPROS $s$ of $G$. Schedule $s_{\text{spros}}$ obtained from $s$ by defining $s_{\text{spros}}(i) = s(i)$ and $s_{\text{spros}}(P_i) = s(i) + e(i)$ for all $i$ is an SPROS of $G_{\text{spros}}$, because every actor is forced in a periodic regime with period $\mu(G)$. Conversely, any SPROS of $G_{\text{spros}}$ can be turned into an equivalent SPROS of $G$, by simply omitting the schedule times of actors $P_i$.

If an SPROS $s$ of $G$ has buffer-capacity distribution function $b$, then $s_{\text{spros}}$ has buffer-capacity distribution function $b_{\text{spros}}$ with, for all actors $i$, $b_{\text{spros}}(i) = b(i)$ and $b_{\text{spros}}(P_i) = 1$. The addition of an actor $P_i$ does not affect the required buffer size of actor $i$ because $P_i$ always immediately starts its firing when $i$ finishes its firing. The buffer for $P_i$ is clearly 1 (even when $P_i$s execution time is 0, because of the initial token). Thus, the buffer-optimal SPROSs of $G$ and $G_{\text{spros}}$ coincide, yielding the same buffer sizes for the actors of $G$.

The technique of [11], [12] can now be used to solve OSBA for a given SRDF graph $G$ by applying it to the transformed graph $G_{\text{spros}}$, as the following reasoning shows.

As shown in [21] and [12], the result of the buffer analysis is

a periodic rate-optimal schedule with an initial transient part. By the construction of $G_{\text{spros}}$, with every actor embedded in a cycle of length $\mu(G)$, every actor is forced to fire in a strictly periodic regime eventually in any rate-optimal execution of $G_{\text{spros}}$. As a consequence, the result of the buffer analysis of [11], [12] applied to $G_{\text{spros}}$ is a schedule $s$ such that for some $N \geq 0$, for all actors $i$ in $G_{\text{spros}}$ and all $k \geq N$, $s(i, k+1) = s(i, k) + \mu(G)$.

The technique of [11], [12] computes the buffer sizes from the periodic part of schedule $s$. Assume that those buffer sizes are given by the buffer-capacity distribution function $b_{\text{spros}}$. It turns out that schedule $s$ can be turned into an SPROS of $G_{\text{spros}}$ with the same buffer-capacity distribution function $b_{\text{spros}}$. Since the total buffer size of $b_{\text{spros}}$ is minimal among all rate-optimal schedules of $G_{\text{spros}}$, it is minimal among all SPROSs of $G_{\text{spros}}$, and hence, because of the one-to-one correspondence between SPROSs of $G$ and $G_{\text{spros}}$ and ignoring the fixed-size buffers of the extra $P_i$ actors, among all SPROSs of $G$.

The proof that $s$ can be turned into an SPROS of $G_{\text{spros}}$ with buffer-capacity distribution function $b_{\text{spros}}$ uses an inductive argument on the length of the transient, i.e., the first $N$ firings of all actors. The key point is that, by the construction of $G_{\text{spros}}$, all firings of one actor in $s$ are *at least* $\mu(G)$ time units apart. Maximally *delaying* the first $N$ firings of all actors in $s$ yields the desired SPROS. If $N = 0$, then $s$ is an SPROS. For the inductive step, assume $N > 0$; consider the actor firings in the transient in reverse order, ordered by their finishing times, and assuming some arbitrary order for firings finishing at the same point in time. The first actor in this sequence, which is (one of) the last actor(s) to complete its $N^{\text{th}}$ firing, can now be delayed as follows. Assume the considered actor is actor $i$. The considered firing has index $N-1$ and start time $s(i, N-1)$. By construction of $G_{\text{spros}}$, $s(i, N-1) = s(i, N) - \mu(G) - \delta$ for some non-negative integer $\delta \in \mathbb{N}_0$. The start time $s(i, N-1)$ can now be delayed by $\delta$ to $s(i, N-1)+\delta$, at the same time setting the start time of the extra actor $P_i$ to $s(i, N-1) + \delta + e(i)$. Thus, these two firings are added to the strictly periodic part of the schedule. Since all the required input tokens for firing $N-1$ of $i$ are already available at $s(i, N-1)$, they are also available at $s(i, N-1)+\delta$. Delaying an actor firing delays the production of output tokens. The start time of firing $N-1$ of actor $P_i$ is adapted appropriately. A crucial point now is that no firings with index $N$ or greater are affected. Firing all actors the same number of times, as in one period of the schedule, leaves the token distribution over the edges of the graph unchanged. Since the periodic part of $s$ has period $\mu(G)$, this means that the production of output by firing $N-1$ of actor $i$ at time $s(i, N) - \mu(G)$ is in time to allow the start of all firings with index $N$ of all the actors of $G_{\text{spros}}$. Again using the fact that firing all actors the same number of times does not affect the token distribution, the periodic part of the new schedule still has buffer-capacity distribution $b_{\text{spros}}$. Assuming now that the first $n$ firings in the considered sequence, i.e., the last $n$ firings in the transient of $s$, have been delayed maximally, the same reasoning applies to firing $n+1$ in the sequence, completing the inductive argument.

In summary, given an SRDF graph $G$, OSBA can be solved by transforming $G$ into $G_{\text{spros}}$ and then apply the analysis of [11], [12]. The periodic part of the computed schedule and the derived buffer sizes form a solution to OSBA, when ignoring the additional actors $P_i$. Both the number of investigated buffer sizes and the length of the execution of an SRDF with a given buffer size may be exponential in the size of the graph. Nevertheless, the technique turns out to work well in practice, as illustrated in [11] and [12] and the next subsection.

TABLE III
AVERAGE OVERESTIMATION OF BUFFER SIZES BY THE OSBA SOLUTION OF NING AND GAO FOR TWO SYNTHETIC BENCHMARKS.

| | avg. overest. (%) | subopt. sol. (%) | avg. overest. subopt. (%) |
|---|---|---|---|
| set 1 | 2.7 | 17.2 | 6.0 |
| set 2 | 1.7 | 48.9 | 2.1 |

### B. Error Quantification

To get some insight in the degree of inaccuracy of the OSBA solution of [1], we used the SDF$^3$ [22] toolset to construct a benchmark of two sets of 10,000 synthetic SRDF graphs each. The first set contains graphs with 7 nodes and on average 9.5 edges. The second set contains graphs with 40 nodes and on average 57.6 edges. The graphs have been constructed with an average in/out-degree of 1.5 edges per node and a maximum of 3 input/output edges per node. These characteristics are chosen to reflect the characteristics of medium-sized and large models one may expect in the DSP and multimedia domains. We use synthetic examples to create a sufficiently large test set.

Experiments were performed on a 3.4 GHz Pentium 4 PC with 4 GB of internal memory. The run-time needed to compute the buffer sizes and the SPROS with the method presented in the previous subsection is on average 2.06 ms for the first set with a maximum of 684 ms. The second set has an average run-time of 156 ms with a maximum of 9786 ms. The experiments never need more than a few MB of memory at a time. These results indicate that the exact solution, despite its theoretical complexity, is efficient in practice.

Table III compares the original and the exact OSBA solutions. It shows the average overestimation in buffer sizes of Ning and Gao's approach for the two sets. This average also takes into account those graphs for which the original algorithm gives an optimal schedule. Table III also shows how often Ning and Gao's algorithm provides a sub-optimal solution, and the average overestimation for those cases.

For the medium-sized graphs, set 1, Ning and Gao's algorithm gives a sub-optimal solution in about one out of six cases. For the large graphs, set 2, the algorithm is sub-optimal in about one out of two cases. This difference is expected because the type of subgraphs for which Ning and Gao's approach is sub-optimal may occur more frequently in larger graphs. If the result is sub-optimal, the overestimation is larger for smaller graphs. Also this is as expected, since smaller graphs typically have smaller absolute total buffer sizes, which means that the relative overestimation is larger. The overestimation is small in all cases, which may explain in part why the sub-optimality of the approach remained undetected up to now.

## IX. Conclusion

This paper revisits buffer sizing for rate-optimal single-rate dataflow scheduling, in particular the Optimal Scheduling and Buffer Allocation (OSBA) problem defined in [1]. We have shown that the linear-programming-based polynomial solution proposed in [1] is sub-optimal, in contrast to the optimality claim made in [1]. We have proven that the problem is in fact NP-complete. Along the way, we have shown that also a generalized version of OSBA with commonly assumed more conservative buffering assumptions is NP-complete as well. We developed exact solutions to both OSBA variants that are efficient in practice. These solutions have been implemented in the tool SDF$^3$ [22], that is available via `http://www.es.ele.tue.nl/sdf3`.

## Acknowledgment

We thank the reviewers for their suggestions to improve this paper. This work was supported in part by the Dutch Science Foundation NWO, project 612.064.206, PROMES.

## References

[1] Q. Ning and G. Gao, "A novel framework of register allocation for software pipelining," in *Proc. ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. ACM, 1993, pp. 29–42.

[2] A. Dasdan, "Experimental analysis of the fastest optimum cycle ratio and mean algorithms," *ACM Transactions on Design Automation of Electronic Systems*, vol. 9, no. 4, pp. 385–418, Oct. 2004.

[3] P. Poplavko, T. Basten, M. Bekooij, J. van Meerbergen, and B. Mesman, "Task-level timing models for guaranteed performance in multiprocessor networks-on-chip," in *Proc. Int'l Conf. on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*. ACM, 2003, pp. 63–72.

[4] S. Stuijk, T. Basten, B. Mesman, and M. Geilen, "Predictable embedding of large data structures in multiprocessor networks-on-chip," in *Proc. Conf. on Digital System Design (DSD)*. IEEE, 2005, pp. 388–395.

[5] O. Moreira, M. Bekooij, and J. Mol, "Online resource management in a multiprocessor with a network-on-chip." in *Proc. Symposium on Applied Computing*. ACM, 2007, pp. 1557–1564.

[6] R. Govindarajan, G. Gao, and P. Desai, "Minimizing memory requirements in rate-optimal schedules," in *Proc. Int'l Conf. on Application-Specific Array Processors*. IEEE, 1994, pp. 75–86.

[7] ——, "Minimizing buffer requirements under rate-optimal schedule in regular dataflow networks," *The Journal of VLSI Signal Processing*, vol. 31, no. 3, pp. 207–229, 2002.

[8] J. Wang, A. Krall, M. A. Ertl, and C. Eisenbeis, "Software pipelining with register allocation and spilling," in *proceedings of the 27th International Symposium and Workshop on Microprogramming and Microarchitecture (MICRO-27)*, December 1994. [Online]. Available: citeseer.ist.psu.edu/wang94software.html

[9] M. S. Lam, "Software pipelining: an effective scheduling technique for VLIW machines," *SIGPLAN Notices*, vol. 39, no. 4, pp. 244–256, 2004.

[10] O. Moreira, J. Mol, M. Bekooij, and J. van Meerbergen, "Multiprocessor resource allocation for hard real-time streaming with a dynamic job mix." in *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2005, pp. 332–341.

[11] S. Stuijk, M. Geilen, and T. Basten, "Exploring trade-offs in buffer requirements and throughput constraints for synchronous dataflow graphs," in *Proc. Design Automation Conf. (DAC)*. ACM, 2006, pp. 899–904.

[12] ——, "Throughput-buffering trade-off exploration for cyclo-static and synchronous dataflow graphs," *IEEE Transactions on Computers*, vol. 57, no. 10, pp. 1331–1345, 2008.

[13] E. Lee and D. Messerschmitt, "Synchronous data flow," *Proceedings of the IEEE*, vol. 75, no. 9, pp. 1235–1245, 1987.

[14] A. Ghamarian, M. Geilen, T. Basten, B. Theelen, M. Mousavi, and S. Stuijk, "Liveness and boundedness of synchronous data flow graphs," in *Proc. Formal Methods in Computer Aided Design (FMCAD)*. IEEE, 2006, pp. 68–75.

[15] S. Sriram and S. Bhattacharyya, *Embedded Multiprocessors: Scheduling and Synchronization*. Marcel Dekker Inc., 2000.

[16] R. Reiter, "Scheduling parallel computations," *Journal of the ACM*, vol. 15, no. 4, pp. 590–599, 1968.

[17] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press/McGraw-Hill, 2001.

[18] R. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*. Plenum, 1972, pp. 85–103.

[19] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[20] P. Murthy, "Scheduling techniques for synchronous and multidimensional synchronous dataflow," Ph.D. dissertation, University of California at Berkeley, Berkeley, USA, 1996.

[21] A. Ghamarian, M. Geilen, S. Stuijk, T. Basten, A. Moonen, M. Bekooij, B. Theelen, and M. Mousavi, "Throughput analysis of synchronous data flow graphs," in *Proc. Int'l Conf. on Application of Concurrency to System Design (ACSD)*, 2006, pp. 25–34.

[22] S. Stuijk, M. Geilen, and T. Basten, "SDF3: SDF For Free," in *Proc. Int'l Conf. on Application of Concurrency to System Design (ACSD)*, 2006, pp. 276–278.

**Orlando Moreira** is a senior scientist at ST-Ericsson. He graduated in Electronics Engineering from the University of Aveiro. Before joining ST-Ericsson, he worked for Philips Research and NXP Semiconductors. In 2007-2008, he led a joint Nokia, NXP and ST-Ericsson team in developing a hard-real-time software architecture for radio. He has published work on reconfigurable computing, real-time multiprocessor scheduling, and data flow analysis.

**Twan Basten** is an associate professor in the Electrical Engineering Department at Eindhoven University of Technology and Research Fellow of the Embedded Systems Institute, both in the Netherlands. He holds an MSc and a PhD in Computing Science from Eindhoven University of Technology. His research interests include the design of resource-constrained embedded systems, multiprocessor systems and computational models.

**Marc Geilen** is an assistant professor in the Department of Electrical Engineering at Eindhoven University of Technology. He holds an MSc in Information Technology and a PhD from the Eindhoven University of Technology. His research interests include modeling, simulation and programming of multimedia systems, multiprocessors and wireless sensor networks, and multi-objective optimization and trade-off analysis.

**Sander Stuijk** received his Master's degree (with honors) in Electrical Engineering in 2002 and his PhD degree in 2007 from the Eindhoven University of Technology. He is currently a postdoc in the Department of Electrical Engineering at the Eindhoven University of Technology. His research focuses on the mapping of streaming multimedia applications on multiprocessor platforms.