ON THE OPTIMAL BLOCKING FACTOR FOR BLOCKED, NON-OVERLAPPED SCHEDULES

by

Praveen Murthy and Edward A. Lee

Memorandum No. UCB/ERL M94/46

10 June 1994

ON THE OPTIMAL BLOCKING FACTOR FOR BLOCKED, NON-OVERLAPPED SCHEDULES

by

Praveen Murthy and Edward A. Lee

Memorandum No. UCB/ERL M94/46

10 June 1994

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

ON THE OPTIMAL BLOCKING FACTOR FOR BLOCKED, NON-OVERLAPPED SCHEDULES

by

Praveen Murthy and Edward A. Lee

Memorandum No. UCB/ERL M94/46

10 June 1994

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

On the Optimal Blocking Factor for Blocked, Non-Overlapped Schedules[1]

Praveen Murthy
Edward.A.Lee

Department of Electrical Engineering and Computer Science
University of California
Berkeley, California 94720

1 Abstract

This paper addresses the problem of determining the optimal blocking factor for blocked, non-overlapped multiprocessor schedules for signal processing programs expressed as synchronous dataflow (SDF) graphs. One approach to determining a multiprocessor schedule for an SDF graph G is to determine a schedule for the J-unfolded graph of G (defined to be the precedence graph of G over J iterations), where $J \geq 1$, and repeat that schedule forever. This approach allows us to exploit some of the inter-iteration parallelism that is usually present in the SDF graph. A schedule for the J-unfolded graph is called a schedule of blocking factor J. It is of interest to determine the value of J that will allow schedules of optimal throughput to be constructed. It will be shown that the critical path of the J-unfolded graph becomes cyclic as J is increased. It will be shown that it is possible to determine this cyclicity by analyzing the critical graph of a matrix that arises in the model that is used. The cyclicity of the critical path implies that we only have to examine a finite number of blocking factors to determine the optimal one.

[1]This research was supported by ARPA and the United States Air Force, and Star Semiconductor

2 Introduction

Synchronous Dataflow (SDF) [1] is a subset of dataflow [8] that has been used as a model for expressing DSP programs [16][17]. An SDF graph is represented by a directed graph $G = (V, E, f, d)$ where V is the set of computation nodes, E is the set of directed edges (representing communication channels), and $f: E \rightarrow Z \times Z$ is a function on the edges to positive integer tuples where the first element of the tuple represents the number of tokens produced on the arc and the second element represents the number of tokens consumed on the arc, and $d(e)$ is the number of delays (initial tokens) on edge e. In addition, each node v in V has an associated positive integer $t(v)$ representing the execution time of v. An example of an SDF graph is given in figure 1(a). Here, one token is produced and two tokens consumed on edge AB. Edge AC has two delays. The SDF graph can be represented by an $m \times n$ topology matrix Γ where m is the number of edges and n the number of nodes. The entry $\Gamma(i, j)$ represents the number of tokens produced by node j on arc i. This number is negative by convention if node j consumes tokens from arc i. The repetitions vector q is the smallest positive integer vector in the null space of Γ; it satisfies the equation $\Gamma q = \bar{0}$. This vector represents the number of times each node must be invoked in order to return each buffer (on each of the arcs) in the graph to its starting state. Notice that since the number of tokens produced and consumed on self-loops has to be the same and does not affect

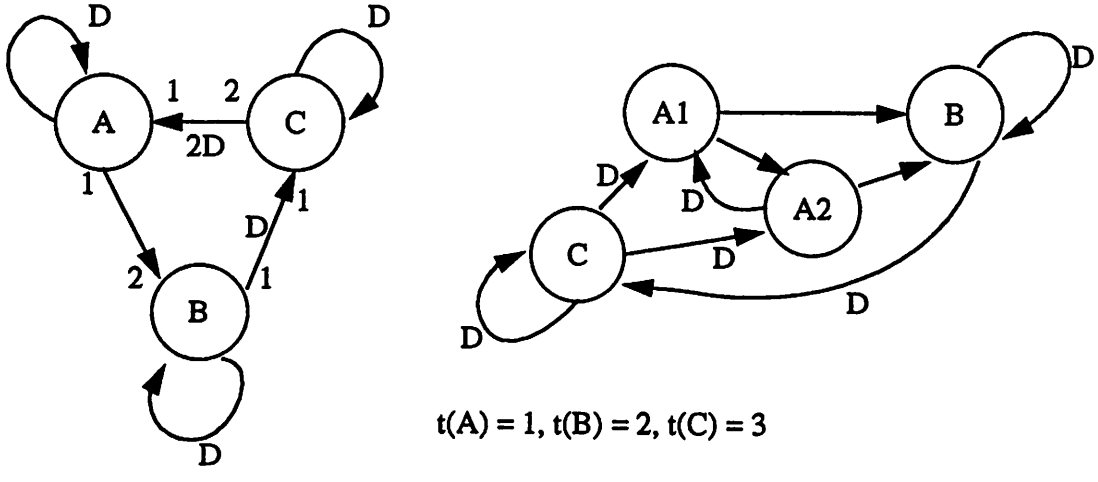


Fig 1. a) An SDF graph. b) The associated homogenous graph

the repetitions vector, these numbers are not shown in figure 1(a). The topology matrix and the repetitions vector for the graph in figure 1(a) are given by

$$\Gamma = \begin{bmatrix} 1 & -2 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 2 \end{bmatrix}$$

and $q = \begin{bmatrix} 2 & 1 & 1 \end{bmatrix}^T$.

A *homogenous* SDF (HSDF) graph is one in which one token is produced and consumed on each arc. Formally, an HSDF graph G is the triple (V, E, d) with V being the set of nodes, E the set of arcs, and d the delay function that denotes the number of initial tokens on the arc. The graph in figure 1(a) is not homogenous because two tokens are produced on edge CA, and two tokens are consumed on edge AB. However, it is possible to systematically construct an HSDF graph from any SDF graph [1]; the resulting graph has $q(i)$ copies of a node i in the original graph. The details of the construction procedure can be found in [1]. The homogenous graph corresponding to the graph in figure 1(a) is shown in figure 1(b).

DSP dataflow programs are non-terminating in nature; they operate on infinite streams of data and produce infinite streams. It is well known that a fundamental upper bound on the throughput achievable in an HSDF graph is given by the inverse of the *maximum cycle mean* [9]:

$$\lambda = MAX_{l \in \Lambda}\{\frac{T(l)}{D(l)}\} \qquad \textbf{(EQ 1)}$$

where Λ is the set of all circuits in the graph, $T(l)$ is the total computation time of circuit l, and $D(l)$ is the delay count of the circuit l. A loop that achieves this maximum is called a *critical loop*. A schedule for an HSDF graph is *rate-optimal* if the iteration-period for the schedule is equal to the maximum cycle mean (also called the *iteration-period bound*). The maximum cycle mean for the graph in figure 1(b) is 3.5. The quantity λ can be found in time $O(|V||E|)$ using Karp's dynamic programming algorithm [15].

In this paper, we are going to assume that a graph contains at least one cycle. If this were not the case, the iteration period bound would be 0 (since there are no loops, the set of loops is empty and equation 1 is defined to be zero over an empty set), and the achievable throughput

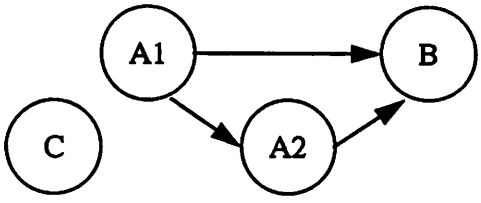


Fig 2. Acyclic precedence graph for graph in figure 1.

would be infinity. We can achieve infinite throughput by scheduling every iteration in parallel using an infinite number of processors. Hence, the problem of finding a schedule that maximizes the throughput is interesting only when the throughput bound is greater than zero (that is, the graph has cycles).

From the HSDF graph, we can construct an *acyclic precedence graph* (APG) *of blocking factor one*. This is the graph obtained by deleting all arcs that have one or more delays on them from the original graph. The APG for the graph in figure 1a is shown in figure 2. Notice that the APG shows only the *intra-iteration* precedences between the nodes.

Define the weight of a path in the APG to be the sum of the execution times of the constituent nodes. Consider now the following strategy for constructing a multiprocessor schedule for an HSDF graph. Instead of using the precedence relations specified by the HSDF graph, we will use only the precedences specified by the APG for constructing the schedule. Each node in APG will be invoked once in the schedule and will be assigned to some processor. Once each processor has finished its tasks, it waits until all other processors have finished their tasks, and then executes its tasks again for the next iteration. This implies that we use some form of barrier synchronization between successive iterations. The first invocation of a node u can only occur if all of its predecessor nodes in the APG have been invoked once. Hence, the total length of the schedule (defined as the maximum of the finishing times for each processor for all of its tasks for one iteration) must be at least equal to the largest-weight path in the APG, the *critical path*. A multiprocessor schedule of this type is called a blocked, non-overlapped schedule of *blocking factor 1*. It is non-overlapped because the n^{th} iteration can only occur after every node has been invoked from the $n-1^{th}$ iteration.

Since a blocked schedule of blocking factor 1 does not exploit *inter-iteration* parallelism, it is useful to schedule the graph over several iterations. A blocked schedule of *blocking factor J* consists of a schedule for the HSDF graph *unfolded J* times. Unfolding a graph J times means considering J successive iterations of the graph. The J-unfolded precedence graph consists of J copies of the APG, and some additional arcs. A node u in the i^{th} copy of the APG, and a node v in the j^{th} copy of the APG, where $j > i$, are connected by an arc in the J-unfolded precedence graph if there is an arc (u, v) in the original graph having $j - i$ delays. The J-unfolded precedence graph can also be referred to as the *APG of blocking factor J*. In this paper, we will use the term "APG" for the APG of blocking factor 1, and the term "J-unfolded graph" interchangeably with the term "APG of blocking factor J" for the J-unfolded precedence graph. Once a J-blocking factor schedule is constructed, it is repeated forever to get a J-periodic schedule. Again, barrier synchronization is assumed between successive blocks of the J-blocking factor schedule.

It is of interest to determine what the optimal value of J should be in order to construct rate-optimal schedules. For example, the critical path in the graph in figure 1(b) is the path $A1 \rightarrow A2 \rightarrow B$, and this path has a weight of 4 (recall that the execution times of nodes A, B, C were 1,2, and 3 respectively). This is evident when we look at the acyclic precedence graph, shown in figure 2. Hence, no schedule (of blocking factor 1) for this graph can have an iteration period of less than 4. The acyclic precedence graph for a blocking factor of 2 (or, equivalently, the 2-unfolded graph) is shown in figure 3(a). It can be verified that the weight of the critical path in this graph is 7. Hence, a schedule can be constructed that has an iteration period of 7. Since two iterations of the original graph occur in 7 time units, the iteration period achieved is 7/2=3.5. Therefore, a blocking factor of 2 is optimal for this graph since it is theoretically possible to construct a blocked, rate-optimal schedule. A rate-optimal schedule using two processors is shown in figure 3(b).

When the blocked schedule is implemented, it is not necessary that we actually use barrier synchronization; the assumption is necessary only for analytical tractability. It has been shown in [18] that if the blocked schedule is implemented in a self-timed manner (where the interprocessor communication points (sends and receives) are the only points of synchronization), then some improvement in throughput can result as the schedule unfolds. However, this improvement

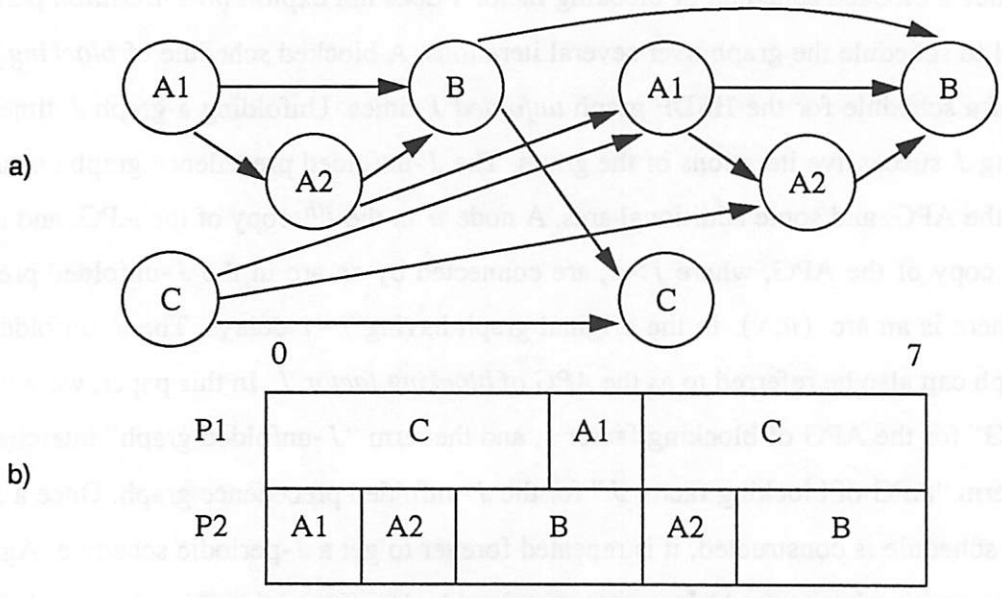


Fig 3. **a)** Acyclic precedence graph of blocking factor 2 for SDF graph in fig. 1. **b)** A blocked, rate-optimal 2-processor schedule.

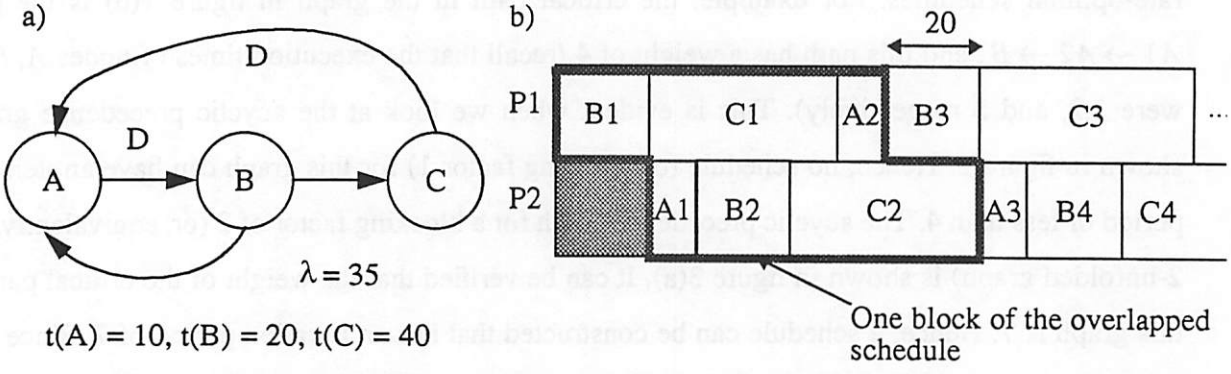


Fig 4. **a)** An HSDF graph that does not have a rate-optimal blocking factor. **b)** A rate-optimal overlapped schedule of blocking factor 2.

depends on the particular schedule that is constructed, and there may not be an improvement for some schedules. Of course, there is no improvement if the original blocked schedule (assuming barrier synchronization) is rate-optimal. Assuming barrier synchronization to determine the throughput achievable (by calculating the critical path) gives a worst-case estimate for the actual performance of any blocked schedule.

Unfortunately, it is not always possible to find a blocking factor that will allow a rate-optimal blocked schedule to be constructed. The graph in figure 4 is an example. As will be shown

later on, no blocking factor for this graph is optimal. However, it is always true that increasing the blocking factor by a finite number m will reduce the achievable iteration period, and in the limit as J goes to infinity, the iteration period will converge to the iteration bound.

There has been considerable work on rate-optimal scheduling in the last five years. Much of this work has been concerned with blocked, overlapped schedules [2][3][13]. A blocked, overlapped schedule is one where not only is the graph unfolded J times before it is scheduled but also where successive blocks are overlapped with each other. This enables inter-iteration parallelism to be exploited to the fullest extent, something blocked, non-overlapped schedules only do to a limited extent. Parhi [2] has shown that it is always possible to unfold a graph a certain number of times to get a perfect-rate graph (a graph where each circuit has only one delay) and schedule the resulting perfect-rate graph rate-optimally using an overlapped schedule. The unfolding factor given by Parhi is the least common multiple (lcm) of all the critical-loop delay counts. Figure 4(b) shows a rate-optimal overlapped schedule of unfolding factor 2 for the graph in figure 4(a). This schedule is overlapped because the third invocation of node B begins before the second invocation of node C has been completed. The rate-optimality of the schedule comes from the fact that in any time window of 70 units, where the window begins at the start of some invocation of node x, there will be two invocations of node x in that time window. Hence, any node x is executed once per 35 time units, meeting the iteration throughput bound.

Much of the work on overlapped scheduling assumes that rate-optimal overlapped schedules can be constructed if a large number of processors are available. Parhi gives an upper bound on the number of processors required, but this number can be quite large. At this time, few heuristics are known for constructing overlapped schedules when the number of processors is fixed and known beforehand. In contrast, there is rich body of work on such heuristics for blocked schedules [10][11][12]. In addition, there has been some recent work on taking interprocessor communication (IPC) into account when constructing blocked multiprocessor schedules [7][14]. Therefore, it is of interest to know theoretical lower bounds on the iteration period achievable for blocked schedules. We cannot hope to know the optimal blocking factor when the number of processors is restricted (or when IPC is taken into account) if we do not know it when the number of processors is unbounded. Therefore, the hope is that our work can be eventually extended (to pro-

vide achievable bounds) to the finite-processor (with IPC) case so that we can use many of the heuristics present for constructing blocked schedules to construct schedules with optimal blocking factors.

3 Max-Plus Algebra Formulation

Max-plus algebra is an algebra where maximization is the addition operation and addition is the multiplication operation [5]. Max-plus addition will be denoted by $\oplus$ and max-plus multiplication will be denoted by $\otimes$. Thus, $5 \oplus 2 = 5$ and $5 \otimes 2 = 7$. The additive identity in this algebra is $-\infty$, denoted by ε, and the multiplicative identity is 0. There is a multiplicative inverse (subtraction in normal algebra) but no additive inverse: the equation $a \oplus x = b$ has no solution if $b < a$. Hence, the algebra does not constitute a ring.

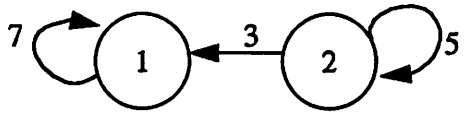
The reason that this algebra is attractive is that it provides an elegant way of describing paths in graphs; this is why it is sometimes referred to as "path algebra". Because of this shorthand and elegant way of formulating paths, certain properties about paths become clearer. It should be emphasized that all of the results derived in this paper have traditional graph-theoretic proofs, but the ideas have been inspired by the max-algebra formulation.

To see how the algebra is relevant to graphs, consider matrices in max-plus. Let A be an $N \times N$ matrix with entries in $\Re \cup \{-\infty\}$. There is an associated graph with N nodes, called the graph of A, where $A(i,j)$ is the weight of the edge (i,j) in the graph. If $A(i,j) = \varepsilon$, then there is no arc between i and j. Conversely, any weighted, directed graph with real-valued weights can be represented by a matrix in max-plus. We use the notation G_A to denote the graph of a matrix A.

An entry in the matrix A^2, where the matrix multiplication is done using max-plus operators, represents the *longest two-arc* path between the corresponding nodes. For example, let

$$A = \begin{bmatrix} 7 & \varepsilon \\ 3 & 5 \end{bmatrix}$$

This represents the graph



We have

$$A^2 = \begin{bmatrix} 7 & \varepsilon \\ 3 & 5 \end{bmatrix} \times \begin{bmatrix} 7 & \varepsilon \\ 3 & 5 \end{bmatrix} = \begin{bmatrix} 14 & \varepsilon \\ 10 & 10 \end{bmatrix}.$$

The longest two-arc path from node 1 to node 1 has weight $max\,(7+7, 5+3) = 14$. The matrix notation A^k allows us to compactly write down the maximum weight paths of length k between any two nodes in G_A. We can write down an expression for the matrix with maximum weight paths between any pair of nodes. It is given by the matrix

$$A^+ \equiv A \oplus A^2 \oplus A^3 \oplus A^4 \oplus \ldots \oplus A^{N-1} \oplus \ldots \qquad \textbf{(EQ 2)}$$

This series converges only if there are no positive weight cycles in the graph (if there were, we would get paths of arbitrarily large weight by traversing positive weight cycles). Hence, if there are no positive weight cycles, the series can be truncated at $N-1$, where N is the number of nodes, since any path of length greater than N has to traverse a cycle, and the cycle cannot increase the weight. Notice that the implied computation in equation 2 is actually the dynamic programming algorithm for finding the all-pairs longest paths in a graph.

It can be shown that the single eigenvalue of A is the maximum cycle mean of the corresponding graph if the graph is strongly connected [5]. If G_A is not strongly connected, then there could be more than one eigenvalue. However, for the A above, there is only one eigenvalue, given by $\lambda = 7$, and $\begin{bmatrix} 4 & 0 \end{bmatrix}^T$ is an eigenvector.

3.1 Description of HSDF graphs in max-plus

HSDF graphs can have arcs with delays; hence we need a way of modeling this. Recall that the delay on an arc represents initial tokens on that arc. Also, the nodes in an HSDF graph have weights that represent execution times; we need a way of modelling this also. The delays can be modeled as follows: let the edge set E in a homogenous graph $G = (V, E)$ be partitioned as $E = \bigcup_{i \in \{0, \ldots, M\}} E_i$ where E_i is the set of edges having i delays, and M is the maximum number of

delays on any arc. Let $M+1$ matrices $A_0, \ldots, A_M$ be defined where a matrix A_i represents the HSDF graph (V, E_i). Notice that (V, E_0), the graph corresponding to the matrix A_0, is the APG defined earlier.

The arcs in the graph can be weighted by assigning to each outgoing arc from a node u the weight $t(u)$, the execution time of node u. This will fail to model the execution time of sink nodes; hence, we introduce a dummy sink node S that has an execution time of zero. An edge (u, S), having zero delays, is added to the graph for every node $u \in V$. The number of vertices in the HSDF graph, N, now includes the dummy sink node.

Matrix products can be used to find largest weight paths of various types. The entry (i, j) in the matrix product $A_0 A_1$, for instance, represents the largest weight two-arc path between i and j with the first arc being a zero-delay arc and the second being a one-delay arc. This is evident when we write down the expression for the entry:

$$(A_0 A_1)(i,j) = max_{1 \le k \le N}(A_0(i,k) + A_1(k,j)) .$$

Define

$$A_{N0}^* = E \oplus A_0 \oplus A_0^2 \oplus A_0^3 \oplus A_0^4 \oplus \ldots \oplus A_0^{N-1} \qquad \textbf{(EQ 3)}$$

where E is the max-plus identity matrix with zeros along the diagonal and ε elsewhere. An entry (i, j), $i \neq j$, in A_{N0}^* corresponds to the weight of the largest weight path between nodes i and j in G_{A_0}. To see this, recall that the series in equation 2 could be truncated at $N-1$ if there were no cycles in G_A. Here, A_0 represents the APG, which is acyclic by definition. Since an entry (i, j) in A_0^k represents the maximum weight over all paths of length k between i and j in the APG, the maximum over $1 \le k \le N-1$ of the A_0^k gives the maximum weight over all paths between i and j. It follows that the largest element of the matrix A_{N0}^* is the critical path in the APG. The inclusion of the identity matrix E in equation 3 means that $A_{N0}^*(i, i) = 0 \ \forall i$. This is done to ensure that paths where the first arc is a delay arc will be representable by appropriate matrix products. For example, the matrix product $A_{N0}^* A_1$ represents maximum weight paths where the last arc in the path is an arc with one delay. We could have a case where the maximum weight path between

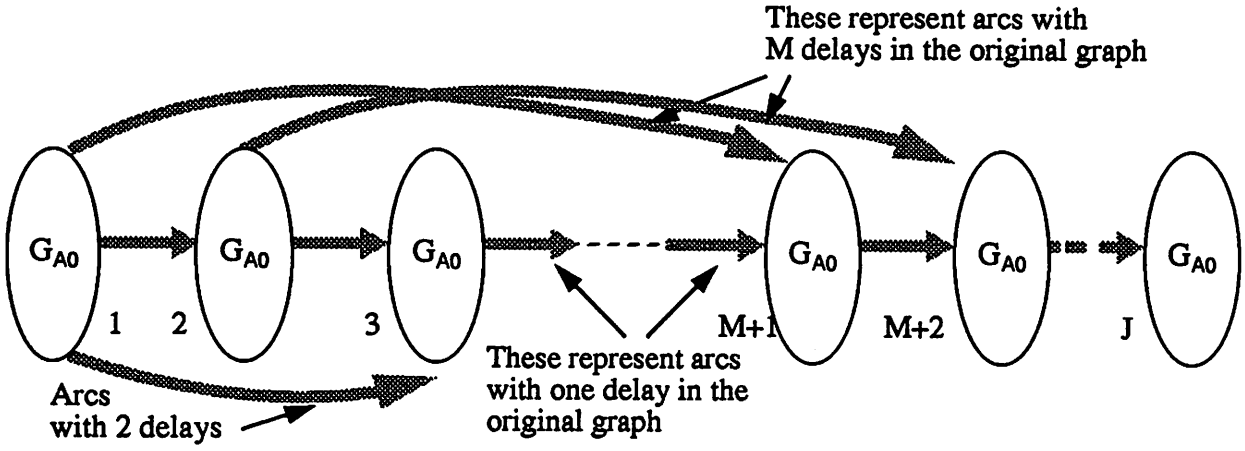


Fig 5. The J-unfolded graph of an HSDF graph G

a pair of nodes consists of only one arc, and that arc is a delay arc. If equation 3 did not have E in the maximization, we would not be able to represent this case.

Consider now the J-unfolded graph of G, denoted $G^{(J)}$. Recall that $G^{(J)}$ consists of J copies of the graph $G_{A_0} = (V, E_0)$ and some additional arcs. We will refer to the i^{th} copy of a node $u \in G_{A_0}$ in $G^{(J)}$ as u^i. There is an arc (u^i, v^j) in $G^{(J)}$, where $i<j$, if and only if $(u, v) \in E_{j-i}$. Figure 5 makes this notion clearer. The grey arcs between different copies of G_{A_0} represent delay arcs having more than one delay in the original graph.

Fact 1: A critical path in $G^{(J)}$ must have $S^K, K \leq J$ as the terminal node, where S^K is the dummy sink node in the K^{th} copy of G_{A_0}. It must have a node from the first copy of G_{A_0} as the initial node.

Indeed, it is clear that the terminal node has to be one of the S^K since if it were not the case, we could always extend the path by adding an S^K, thus increasing the weight of the path (recall that every node in G_{A_0} is connected to S, and every node has positive execution time).

Definition 1: The term $max\{A\}$, where A is a matrix, is defined to be the maximum element of A.

Define $C(J)$ to be an $N \times N$ matrix, where N is the number of nodes in the original HSDF graph (including the dummy sink node), containing the largest weight paths in $G^{(J)}$. This

means that an entry (i, j) in $C(J)$ is the maximum weight of all paths between nodes i and j having $J-1$ delays or fewer in the original HSDF graph.

Fact 2: The maximum weight path p in $G^{(J)}$ between a node u^i and a node v^j, $J \geq j > i$, depends only on $j-i$.

Proof: This is because the existence of arcs between the i^{th} and k^{th} copies of G_{A_0} depends only on $k-i$ since such an arc exists if there is an arc with $k-i$ delays in the original HSDF graph.

Theorem 1:

$$C(J) = C(J-1) \oplus A_{N0}^{*} \bigoplus_{k=1}^{MIN(J-1, M)} A_k C(J-k) \quad \forall J \geq 2 \qquad \textbf{(EQ 4)}$$

where $C(1) = A_{N0}^{*}$, and $M > 0$, where M is the maximum number of delays on any arc in the original HSDF graph. (If $M = 0$, then $C(J) = A_{N0}^{*}$ $\forall J$)

Proof: The proof is by induction on J. We want to prove that $C(J)$ has the property that it is the matrix of maximum weight paths (as defined) if the $C(j)$ have the property for $j \leq J-1$. Assume $J = 2$. For any two nodes, consider the largest weight path between them in $G^{(2)}$. This path can have at most one delay. Hence the path either consists of a subpath in the first copy of G_{A_0}, a delay arc to enter the second copy of G_{A_0}, and a subpath in the second copy of G_{A_0}, or just consists of a path in the first copy of G_{A_0} (i.e, has no delay arcs at all). This can be expressed as $A_{N0}^{*} \oplus A_{N0}^{*} A_1 A_{N0}^{*}$. This satisfies the equation given in the theorem.

Now suppose that the equation holds for all $j \leq J-1 < M+1$. Consider the maximum weight path between any two nodes u, v in $G^{(J)}$. If the maximum weight path between u and v goes through fewer than $J-1$ delays, then this weight will be reflected in $C(J-1)$ because of fact 2 and the induction hypothesis. If the maximum-weight path does have $J-1$ delays, then the first delay arc on the path is an i-delay arc for some $1 \leq i \leq J-1$. This i-delay arc is between a node in the first copy of G_{A_0} and a node in the $i+1^{th}$ copy of G_{A_0}. From here, we want to reach the J^{th} copy of G_{A_0}. An entry (p, q) in the matrix $C(J-i)$ corresponds to the maximum weight $J-1-i$-delay path between p and q (by the induction hypothesis). Hence, the matrix product $A_{N0}^{*} A_i C(J-i)$ will give us the maximum weight path containing an i-delay arc fol-

lowed by a path with $J-1-i$ delays. The maximum over $1 \le i \le J$ of all such matrix products, along with the matrix $C(J-1)$ gives us the maximum-weight $J-1$ -delay path.

The argument is similar if $J \ge M+1$. The difference is that now, the i-delay arc used to leave the first copy of G_{A_0} can only be in the range $1 \le i \le M$.

Corollary 1: If $M = 1$, then

$$C(J) = \bigoplus_{k=0}^{J-1} \left(A_{N0}^{*} A_1\right)^k A_{N0}^{*} \qquad J \ge 1 \qquad \textbf{(EQ 5)}$$

Proof: First off, we have $MIN(J-1, 1) = 1$, for $J \ge 2$. Equation 4 can now be written as

$$C(J) = C(J-1) \oplus A_{N0}^{*} A_1 C(J-1), J \ge 2$$

We can continue by substituting for $C(J-1)$:

$$C(J) = C(J-2) \oplus A_{N0}^{*} A_1 C(J-2) \oplus A_{N0}^{*} A_1 C(J-2) \oplus A_{N0}^{*} A_1 A_{N0}^{*} A_1 C(J-2)$$

$$= C(J-2) \oplus A_{N0}^{*} A_1 C(J-2) \oplus \left(A_{N0}^{*} A_1\right)^2 C(J-2) .$$

The second equality follows from the idempotency of the addition operator in max algebra (i.e, $a \oplus a = a$). We continue the process of substitution to get equation 5.

Unfortunately, equation 4 is difficult to analyze any further (despite its similarity to convolution). Equation 5 is easier with a few restrictions that will be described below. Therefore, we would like to modify the original HSDF graph to have arcs with one delay at most. This can be done using the following technique of graph expansion: for an arc (u, v) that has $m > 1$ delays, create $m-1$ dummy nodes, $u^1, \ldots, u^{m-1}$, with zero execution times, and replace the arc (u, v) with the path $\left(u, u^1, \ldots, u^{m-1}, v\right)$. Each edge in the path has one delay. By this technique, we can represent an arbitrary graph as a graph containing arcs having at most one delay.

We are interested in critical paths in the J-unfolded graph. The maximal element of $C(J)$ in equation 5 (the critical path in the J-unfolded graph), for J large enough, is going to traverse a cycle in the dataflow graph many times. In other words, the critical path is going to be the one that can reach the J^{th} copy of G_{A_0}; a path that does not reach the J^{th} copy can only be critical for the first few J. This is because a path that does not reach the J^{th} copy must be an acyclic path in the

original graph. More formally, consider the graph obtained by deleting all of the strongly connected components from the HSDF graph. Let the maximum weight path be a in the resulting graph. Then, the path a can be critical for at most $weight(a)/\lambda$ unfoldings. Therefore, the weight of any cycle unfolded enough times will be greater than the weight of any acyclic path.

If the original graph (without the dummy sink node) is strongly connected, then there is always a critical path that reaches the J^{th} copy of G_{A_0} for all values of J. To see this, suppose that no critical path reaches the J^{th} copy of G_{A_0}. Let S^K be the dummy sink node that terminates some such path for some $K < J$ in $G^{(J)}$ (by fact 1). Consider the node immediately before S^K in the path, say v^K. Since the original graph G is strongly connected, v cannot be a sink node in G. Hence, v must be the predecessor to another node u, $u \neq S$, and the arc (v, u) has either 1 or 0 delays. In either case, we can extend the path in $G^{(J)}$ by including either u^{K+1} or u^K, thus increasing the weight (or keeping it the same). We will be unable to extend the path only when $K = J$ and every arc leaving v is a delay arc.

Hence, we can ignore all of the lower order terms in the summation in equation 5 if J is large enough (for a non-strongly connected graph), or if the graph is strongly connected. To simplify things, we will assume that the original HSDF graph is strongly connected. The results of the analysis for the strongly connected case can be used to analyze the general case. With this simplification, we get that

$$C(J) = \left(A_{N0}^{*} A_1\right)^{J-1} A_{N0}^{*} \qquad \textbf{(EQ 6)}$$

Define

$$B = A_{N0}^{*} A_1 \qquad \textbf{(EQ 7)}$$

We are interested in the asymptotic behaviour of B^J as J goes to infinity because we are interested in the maximum element of $C(J)$, the critical path. Note that the matrix B, defined in equation 7, corresponds to a graph $G_B = (V, E')$ where V is the set of nodes from the original graph, and $(u, v) \in E'$ iff there is a path in the original graph G from u to v with the last arc in the path being a unit delay arc. The weight of the edge is the maximum of the weights of all such paths.

If the original graph is strongly connected, then we do not need to add the dummy sink node since there are no sink nodes in the graph. The following lemma shows that if the original graph is strongly connected, then it suffices to analyze B^J and not $B^{J-1}A^*_{N0}$.

Lemma 1: Suppose that the original HSDF graph is strongly connected. Suppose also that we have not added the dummy sink node to the graph. Then, $max\{B^J\} = max\{B^{J-1}A^*_{N0}\}$.

Proof: Consider the critical path in the $G^{(J)}$ graph (without the dummy sink). It must be the case that the terminal node of the critical path has only outgoing unit-delay arcs as pointed out before. Therefore, if we multiply the right hand side of equation 6 with A_1, then the maximum element of the right hand side will be the weight of a path where the last arc is a delay arc. The weight of the last delay arc is equal to the execution time of the last node. Hence, this is equal to the weight of the critical path.

Therefore the results of analyzing the asymptotic properties of B^J can be used directly without having to consider the multiplication by A^*_{N0} in equation 6.

3.2 Properties of the G_B graph

We denote the maximum cycle mean in a graph $G_B = (V, E)$ (or the eigenvalue its associated matrix representation B) by λ_G (or λ_B). The symbol " $\Rightarrow$" is used to denote paths and the symbol "$\rightarrow$" denotes arcs.

Definition 2: We define the maximum cycle mean for G_B as

$$\lambda_B = MAX_{l \in \Lambda}\{\frac{T(l)}{|l|}\} \qquad \textbf{(EQ 8)}$$

where Λ is the set of circuits, $T(l)$ is the sum of the weights of the arcs on circuit l, and $|l|$ is the number of nodes in circuit l.

Definition 3: A maximal strongly connected subgraph (m.s.c.s) S of G is a strongly connected subgraph of G such that there is no other strongly connected component that properly contains S.

Lemma 2: If G is strongly connected, then G_B is connected.

Proof: Let u, v be any two nodes in G. Then u can reach v and vice-versa. Since the cycle $u \Rightarrow v \Rightarrow u$ must have at least one delay, there is at least one node reachable by both u and v in G_B. Hence, u, v are in the same connected component.

Lemma 3: Let $V_d \subseteq V_G$ be the set of nodes having at least one input delay arc. If G is strongly connected, then G_B has only one m.s.c.s, and the node set of the m.s.c.s is V_d.

Proof: Let $u, v \in V_d$. Then $\exists u', v' \in V_G$ s.t. (u', u), (v', v) are delay arcs by the definition of V_d. Since G is strongly connected, there is a path in G of the form $u \Rightarrow v' \rightarrow v \Rightarrow u' \rightarrow u$. Therefore, $u \Rightarrow v \Rightarrow u$ is a path in G_B. A node not in V_d cannot be reachable in G_B and hence cannot be any m.s.c.s. Since G_B is connected (from lemma 2), the only m.s.c.s has the node set V_d.

Lemma 4: $\lambda_B = \lambda_G$, where G is the HSDF graph.

Proof: Let $C = u_1 \rightarrow u_2 \rightarrow \ldots \rightarrow u_n \rightarrow u_1$ be a critical cycle in G. Then, $\exists u_{i1}, \ldots, u_{im} \in V_G$ in the cycle C such that the incoming arc to each node u_{ij} is a delay arc in the cycle. Therefore, $C' = u_1 \rightarrow u_{i1} \rightarrow u_{i2} \rightarrow \ldots \rightarrow u_1$ is a cycle in G_B with $weight(C') \geq weight(C)$ because the weight of an arc $u_{ij} \rightarrow u_{i(j+1)}$ in G_B is at least as big as the weight of the corresponding path in G. Also, the number of nodes in C' is equal to the number of delays in C. Therefore, $\lambda_B \geq \lambda_G$.

Let us now prove that $\lambda_B \leq \lambda_G$. Let the C above be a critical cycle in G_B. Then $\exists v_1, v_2, \ldots, v_n \in V_G$ such that $P = u_1 \Rightarrow v_1 \rightarrow u_2 \Rightarrow v_2 \rightarrow u_3 \Rightarrow \ldots \Rightarrow v_n \rightarrow u_1$ is a path in G. Each of the arcs (v_i, u_{i+1}) in P is a delay arc. If P is a simple cycle, then we are done since its weight is the same as the weight of the cycle in G_B. If P is not a simple cycle, then it contains many sub-cycles. Consider the following algorithm for "pruning" P of its sub-cycles. We scan P from the left until we find a node s_1 that repeats. The section of the path in P between the two instances of s_1 is removed from P. This is equivalent to "merging" the two instances of s_1 and deleting the path in between. This process is continued until P is free of subcycles. Let n' be the number of delays in the simple cycle P' obtained by pruning P. Then there are $n - n'$ delays in

the deleted sub-cycles $s_i \Rightarrow s_i$. Assume that each of these is a simple cycle. Denoting the number of these sub-cycles by S we have $S \le n - n'$ because each cycle must have at least one delay. If m_i is the number of delays in cycle $s_i \Rightarrow s_i$, we get that

$$weight(C) = weight(P) = weight(P') + \sum_{i=1}^{S} weight(s_i \Rightarrow s_i) \le n'\lambda_G + \sum_{i=1}^{S} m_i\lambda_G = n\lambda_G$$

We also have $weight(C) = n\lambda_B$. Hence, $\lambda_B \le \lambda_G$. If some of the cycles $s_i \Rightarrow s_i$ are not simple, then they too can be pruned of sub-cycles and we can use similar arguments as above to bound the sums.

Lemma 5: If C is a critical cycle in G, then the cycle constructed from C in G_B, as shown in the first part of the proof in lemma 4, is also critical.

Proof: Letting $C = u_1 \to u_2 \to \ldots \to u_n \to u_1$ as before, we know that the arcs in $C' = u_1 \to u_{i1} \to u_{i2} \to \ldots \to u_1$ have weights larger than or equal to the weights of the corresponding paths in C. If the weights were larger, C' would have a larger weight. By lemma 4, we know that this cannot be the case; hence C' is a critical cycle in G_B.

4 Cyclicity of B

We are interested in the asymptotic properties of B^J as J goes to infinity. Since B has positive weight cycles, this will be ill-defined since every entry goes to infinity in the limit. Therefore, we normalize B by subtracting λ from each entry. In max-plus notation, this is represented as $\lambda^{-1}B$ (recall that max-algebra has a multiplicative inverse, namely subtraction). The actual weight in B^J can be gotten from the formula $\lambda^J\left(\lambda^{-1}B\right)^J$. Hence, every cycle in $\lambda^{-1}B$ has non-positive weight with the critical cycles having 0 weights.

Since we are interested in the critical path of the J-unfolded graph, we are interested in the quantity

$$CP(J) = max\{C(J)\} = max\{\lambda^J\left(\lambda^{-1}B\right)^J\} = J\lambda + max\{\left(\lambda^{-1}B\right)^J\}, \qquad \textbf{(EQ 9)}$$

where we have abused notation by combining normal algebra and max-algebra; the term $J\lambda$ uses normal multiplication while the second term uses max-algebraic operations. Define

$$M_B(J) = max\{\left(\lambda^{-1}B\right)^J\} \quad \textbf{(EQ 10)}$$

If $M_B(J) \neq 0$, then J is not a rate-optimal blocking factor since $CP(J)/J > \lambda$.

For graphs that are not strongly connected, we have to use equation 5 to compute $C(J)$ (assuming that J is large enough so that rest of the terms in the summation drop out). In this case, the critical path is given by

$$CP(J) = max\{C(J)\} = max\{\lambda^{J-1}\left(\lambda^{-1}B\right)^{J-1}A^*_{N0}\} \quad \textbf{(EQ 11)}$$
$$= J\lambda - \lambda + max\{\left(\lambda^{-1}B\right)^{J-1}A^*_{N0}\}$$

For non strongly connected graphs, redefine

$$M_B(J) = max\{\left(\lambda^{-1}B\right)^{J-1}A^*_{N0}\} \quad \textbf{(EQ 12)}$$

If $M_B(J) \neq \lambda$ in equation 12, then J is not a rate-optimal blocking factor since $CP(J)/J > \lambda$.

The following definitions apply to generic, weighted digraphs:

Definition 4: The critical graph of a graph G, denoted G^C, is a graph consisting of those nodes and arcs of G that belong to some critical circuit of G. This graph pays a key role in the asymptotic analysis of B^J.

Definition 5: Similarly, the non-critical graph of a graph G, denoted G^{NC}, is a graph consisting of those nodes and arcs of G that belong to some non-critical circuit of G.

For example, figure 6 shows a graph, its critical graph, and its non-critical graph.

Definition 6: The cyclicity of an m.s.c.s is the greatest common divisor (gcd) of the lengths of all its circuits. The cyclicity of a graph G is the least common multiple (lcm) of the cyclicities of all its m.s.c.s's.

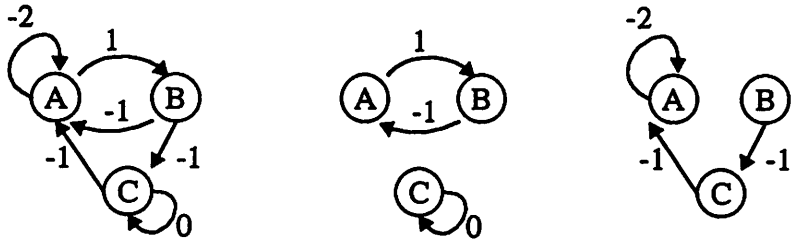


Fig 6. A graph, its critical graph, and its non-critical graph

The cyclicity of the m.s.c.s containing nodes A and B in the critical graph of figure 6 is 2, and the cyclicity for the m.s.c.s containing node C is 1. The cyclicity of the critical graph is lcm(1,2)=2.

Definition 7: [6] A matrix A is said to be cyclic if there exist d and T such that $\forall m > T, A^{m+d} = A^m$. The least such d is called the cyclicity of matrix A and A is said to be d-cyclic.

For the graph in figure 6, the matrix A is given by $A = \begin{bmatrix} -2 & 1 & \varepsilon \\ -1 & \varepsilon & -1 \\ -1 & \varepsilon & 0 \end{bmatrix}$, and by calculating the first few powers of A, we get

$$A^2 = \begin{bmatrix} 0 & -1 & 0 \\ -2 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix}, A^3 = \begin{bmatrix} -1 & 1 & 0 \\ -1 & -1 & -1 \\ -1 & 0 & 0 \end{bmatrix}, A^4 = \begin{bmatrix} 0 & 0 & 0 \\ -2 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix}, A^5 = \begin{bmatrix} -1 & 1 & 0 \\ -1 & -1 & -1 \\ -1 & 0 & 0 \end{bmatrix}$$

and we see that $\forall m > 2, A^{m+2} = A^m$. Of course, it is also true that $\forall m > 2, A^{m+d} = A^m$ where $d = 2k$, $k = 1, 2, 3, \ldots$; we pick the least such d, which is 2, and A is 2-cyclic.

Now we prove the main result in the paper, namely, the cyclicity of B. First we need the following number-theoretic lemma, stated without proof:

Lemma 6: [21] Given n integers $0 < a_1 < a_2 < \ldots < a_n$, suppose that $Q = k \cdot gcd\{a_1, \ldots, a_n\}$ Then, for each $k \geq k^*$, where $k^* = (a_1 - 1)(a_n - 1)$, the Diophantine equation

$$Q = \sum_{i=1}^{n} a_i x_i$$

always has a solution in non-negative integers x_i. The value given for k^* is not tight for $n > 2$; the problem of finding the least k^* such that the lemma holds is still open [20]. Better bounds are

given by Erdös and Graham in [23], and an algorithm for computing the bound in time $O(na_n^2)$ is given by Wilf in [22].

Theorem 2: If G is strongly connected, then B is ρ-cyclic, where ρ is the cyclicity of $G^C(B)$.

Proof: We wish to show that for any J large enough, $B^J = B^{J+\rho}$. That is, the maximum weight over J length paths between any pair of nodes u, v, in the graph G_B, is equal to the maximum weight over $J+\rho$ length paths between the same pair u, v in the graph G_B. Recall that G_B is connected and has one m.s.c.s, denoted $G^S(B)$. There are two cases to consider:

Case 1: One of the nodes u, v belongs to $G^C(B)$.

Case 2: Neither of the nodes belongs to $G^C(B)$.

For case 1, assume without loss in generality that $v \in G^C(B)$. Consider a path between u and v. In general, this path consists of an acyclic path between u and v, denoted $p(u, v)$, some number of non-critical circuits, and some number of critical circuits. Consider an acyclic path $p^*(u, v)$ of maximum weight between u and v. Consider values of J of the form $J = |p^*| + k\rho$, for k large enough, where the notation $|p|$ denotes the length of the path p. By lemma 6, we know that there is a critical circuit of length $k\rho$ if k is large enough, in the m.s.c.s of $G^C(B)$ that v belongs to (note that the critical graph can have more than one m.s.c.s). Actually, we know that there are critical circuits of length $k\rho'$ in the m.s.c.s of $G^C(B)$ that v belongs to, where ρ' is the cyclicity of that m.s.c.s (defined as the gcd of the lengths of all circuits in the m.s.c.s). However, we do not want our results to depend on the particular m.s.c.s of $G^C(B)$ that v belongs to (since we want to prove cyclicity for paths between all pairs of nodes); hence, it suffices for critical circuits of length $k\rho$ to exist, where ρ is the lcm of the cyclicities of each m.s.c.s in $G^C(B)$ (as per definition 6). Therefore, a path of length $J = |p^*| + k\rho$ has the same weight as the path p^*, and this is the maximum possible weight. A path of length $J+\rho = |p^*| + (k+1)\rho$ also has the same maximum weight. Hence, for J length paths of the form $J = |p^*| + k\rho$, the maximum weight between u and v is ρ cyclic.

Consider now the possibility that the J length path is not expressible as $J = |p^*| + k\rho$ in case 1. In general, if a J length path exists at all, it can be expressed as

$$J = |p| + k_1\rho_{NC} + k_2\rho, \qquad \textbf{(EQ 13)}$$

where ρ_{NC} is the cyclicity of the non-critical graph of B, k_1 and k_2 are large integers, and p is some acyclic path between u and v. We claim that for any J expressible as in equation 13, the integer k_1 lies in the range $k_1^* \le k_1 \le k_1^* + \rho - 1$, where k_1^* is the smallest integer such that for all $k_1 \ge k_1^*$, there is a non-critical circuit of length $k_1\rho_{NC}$ (in lemma 6). The implication of this claim is that as J increases, even if we have to use non-critical circuits in the J length path, we need to only use a bounded number of them. To prove the claim, suppose that there is a J length path where $J = |p| + k_1\rho_{NC} + k_2\rho$, with k_1 and k_2 arbitrary but large enough (for lemma 6 to apply). We want to show that we can construct another J length path with fewer non-critical cycles and a larger number of critical cycles. That is, we want to replace some of the non-critical circuits with critical circuits. If we are able to do this, then we will produce a J length path of larger weight. Note that we have the same acyclic path in both paths. We want to solve the equation $k_1\rho_{NC} = k'_1\rho_{NC} + k'_2\rho$ where $k_1^* \le k'_1 \le k_1^* + \rho - 1$. This equation represents the partitioning of a non-critical circuit of length $k_1\rho_{NC}$ into a non-critical circuit of length $k'_1\rho_{NC}$, and a critical circuit of length $k'_2\rho$. It can be verified that if we set

$$k'_2 = \frac{\rho_{NC}}{\rho}(k_1 - k_1^* - c),$$

where $c = (k_1 - k_1^*) \bmod \rho$, then $k'_1 = k_1^* + c$ and we get $k_1^* \le k'_1 \le k_1^* + \rho - 1$. This proves the claim. Thus, if we have a J length path with $J = |p| + k_1\rho_{NC} + k_2\rho$, then we can construct a $J + \rho$ length path by traversing a critical circuit of length $(k_2 + 1)\rho$, and this path has the same weight as the J length path. Moreover, we claim that this is the maximum weight $J + \rho$ length path. Suppose, to the contrary, that this is not the case, and the $J + \rho$ length path has larger weight. If the maximum weight $J + \rho$ length path consists of a non-critical cycle of length $k_1\rho_{NC}$, where $k_1 \le k^*_1 + \rho - 1$ (this has to be the case by the above claim), and a critical cycle of length $k_2\rho$, then we can construct a J length path that has the same weight by traversing a critical cycle of length $(k_2 - 1)\rho$. This contradicts the premise that the J length path had smaller weight. Hence, for J as in equation 13, the maximum weight path is ρ cyclic.

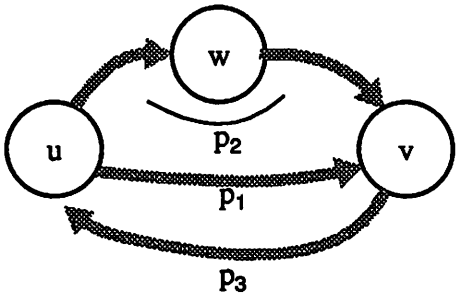


Fig 7. The case where two nodes are not on any critical cycle.

The last possibility for case 1 is if there is no path of length J between u and v. If it turns out that a $J + k\rho$ length path does exist for some k, then the situation of not having a J length path is transitory since we have already shown that if a path of some length $J + k\rho$ exists, then so does a path of length $J + (k + 1)\rho$. If there is no path of length $J + k\rho$ for any value of k, then every $J + k\rho$ length path has a value of ε. This completes the proof that the maximum weight path between nodes u and v is cyclic.

For case 2, there is the possibility that for certain values of J, every path between the two nodes consists of nodes not on any critical cycle. In this case, we cannot apply our arguments above since when we increase J by ρ, we cannot traverse a critical circuit of a larger length (by ρ) since in order to do so, we have to have a node from some critical circuit in the path. Hence, for these values of J, we might be able to traverse only non-critical circuits, and this will drive the weight of the path to ε. However, we show that this situation cannot exist. Let u and v be two nodes not in $G^C(B)$. However, assume that they do belong to $G^S(B)$. Since we are concerned about the case where these nodes can reach each other via arbitrarily long paths having no nodes from $G^C(B)$, assume that there is a non-critical cycle between u and v having no nodes from $G^C(B)$. Figure 7 shows the situation. The paths p_1 and p_3 form the non-critical cycle without any nodes from $G^C(B)$. The node w is some node on some critical circuit, and because $G^S(B)$ is strongly connected, there is a path from u to w and from w to v. These two paths are collectively denoted p_2. Therefore, there are paths of length $k_1(|p_1| + |p_3|) + |p_1|$ between u and v for all positive k_1, and the weight of these paths goes to ε as k_1 increases. We want to show that we can construct another path, that goes through w, of the same length as these paths. Let

$\rho' = gcd(|p_1| + |p_3|, |p_2| + |p_3|)$. Then there is a path of the same length through w if the following equation has a non-trivial solution in k_2 and k_3 (that is, $k_2 > 0$):

$$k_1(|p_1| + |p_3|) + |p_1| = k_2\rho + k_3\rho' + |p_2|. \qquad \textbf{(EQ 14)}$$

The right hand side of the equation represents the length of a path that goes through some number of critical cycles (from w), some number of the two non-critical cycles, and the acyclic path p_2. Since ρ' divides $|p_1| + |p_3|$ and $|p_2| + |p_3|$, it divides their difference, $||p_2| - |p_1||$. Therefore, equation 14 can be written as $k_2\rho = (k_1 l_1 - l_2 - k_3)\rho'$, where $l_1 = (|p_1| + |p_3|)/\rho'$, and $l_2 = (|p_2| - |p_1|)/\rho'$. We can choose k_3 such that k_2 is an integer. As in the previous case, we can only generate paths of this kind if k_3 and k_2 are big enough for lemma 6 to apply; therefore, if we set $k_3 = k_3^* + c$, $c = \left(k_1 l_1 - l_2 - k_3^*\right) mod\ \rho$, then $k_3^* \le k_3 \le k_3^* + \rho$ where k_3^* is the smallest integer such that non-critical circuits of length $k_3\rho'$ can be generated for all $k_3 \ge k_3^*$. Ofcourse, all this requires that k_1 be big enough.

Therefore, for two nodes not on any critical cycle, we can still construct paths between them (if they exist) that include nodes on critical cycles. This means that a $J + \rho$ length path will have the same weight since we can traverse a critical circuit of larger length (by ρ).

A final possibility in the case where neither of the nodes is on any critical cycle is if one of the nodes is not in $G^S(B)$. This possibility is similar to the above since there is always a path from such a node to some node in $G^S(B)$. If both of the nodes are not in $G^S(B)$, then neither node is reachable and there is no path between them of any length. **QED**.

Theorem 2 tells us that it is enough to consider $T + \rho$ blocking factors in order to determine whether there is a rate-optimal blocking factor, where T is the length of the transient before B becomes cyclic. A blocking factor J greater than $T + \rho$ will result in $M_B(J) = M_B(J - \rho)$, allowing us to determine $M_B(J)$ for all J.

Let us now consider the case where G is not strongly connected. In this case, B may not be connected and may have more than one m.s.c.s, with some m.s.c.s's having strictly negative weight cycles. If these m.s.c.s's have no access to critical cycles, then certain weights in B^J will go to $-\infty$ as $J \to \infty$. Therefore B will not be cyclic in general. However, the critical path is

always going to be cyclic because the critical path is eventually going to consist of many traversals of a critical circuit. Hence, the starting node in any critical path is going to be a node that can reach an m.s.c.s in B having at least one critical cycle. Formally, we have the following theorem:

Theorem 3: The critical path is ρ-cyclic, where ρ is the cyclicity of $G^C(B)$.

Proof: The previous results on the cyclicity of B have illustrated that a maximum weight path between two nodes that happens to traverse a critical cycle is cyclic. The critical path in a J-unfolded graph has to consist of several traversals of some critical cycle if J is large enough. Hence, the cyclicity of such a path is equal to ρ.

We use the preceding results to find the optimal blocking factors, when they exist, for several graphs in the next section. Unfortunately, it will be shown that the transient T can be quite large sometimes. It will also be shown the cyclicity of $M_B(J)$ is sometimes less than ρ; this occurs if there is more than one m.s.c.s in $G^C(B)$ and the critical path always traverses critical cycles from a subset of the total number of m.s.c.s's in $G^C(B)$. Then, the cyclicity of the critical path will be the lcm of the cyclicities of that subset of the m.s.c.s's and this might be smaller than the cyclicity of B. We note that even if an optimal blocking factor does not exist, we can still choose blocking factors for which $M_B(J)$ is the smallest.

5 Examples

Example 1: Consider the graph in figure 1, reproduced in figure 8. Since the graph has arcs with at most one delay, we do not need to do any graph expansion. The relevant matrices are:

$$A_0 = \begin{bmatrix} \varepsilon & 1 & 1 & \varepsilon & 1 \\ \varepsilon & \varepsilon & 1 & \varepsilon & 1 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & 2 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & 3 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}, \text{ and } A_1 = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 2 & 2 & \varepsilon \\ 3 & 3 & \varepsilon & 3 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}. \text{ We calculate } A_{N0}^* = \begin{bmatrix} 0 & 1 & 2 & \varepsilon & 4 \\ \varepsilon & 0 & 1 & \varepsilon & 3 \\ \varepsilon & \varepsilon & 0 & \varepsilon & 2 \\ \varepsilon & \varepsilon & \varepsilon & 0 & 3 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & 0 \end{bmatrix}.$$

Similarly, we can calculate B. The graph for $\lambda^{-1}B$ is shown in figure 9(a) and its critical graph is shown in figure 9(b). From the critical graph, it is seen that the cyclicity is 2. By simulation (i.e, by evaluating successive powers of B and stopping when we see the periodic regime), we find

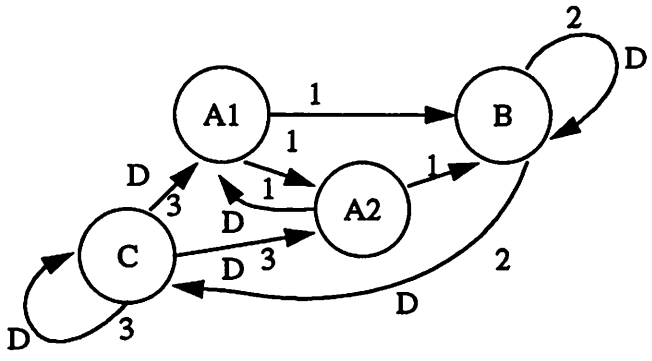


Fig 8. HSDF graph from figure 1.

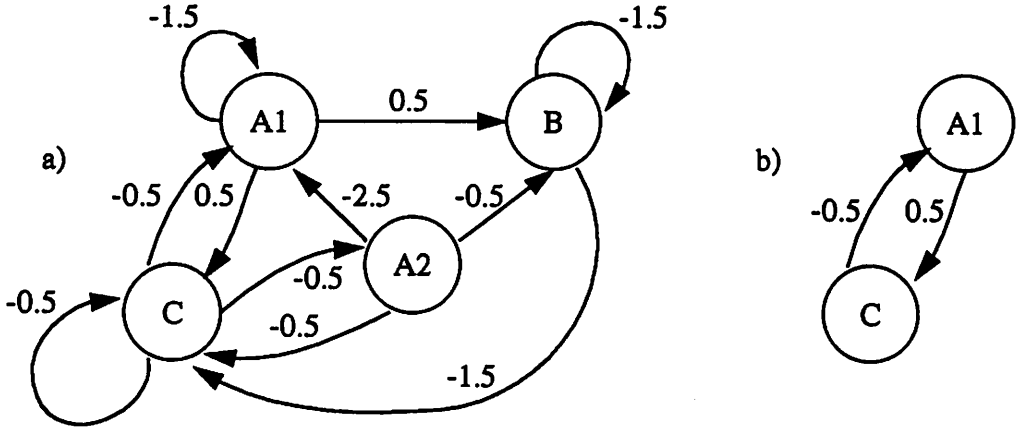


Fig 9. a) The graph of the normalized B-matrix. b) The critical graph of graph in a)

that $T = 3$. Hence, $B^{J+2} = B^J \ \forall J \geq 3$. By calculating the first four powers of B, we get the following values for $M_B(J)$:

$M_B(1) = 0.5, M_B(2) = 0.0, M_B(3) = 0.5, M_B(4) = 0.0.$

Therefore, for this graph, even blocking factors are rate-optimal while odd blocking factors are not. The iteration period, as a function of J, is given by

$$T(J) = CP(J)/J = (\lambda J + M_B(J))/J = \lambda + M_B(J)/J, \qquad \textbf{(EQ 15)}$$

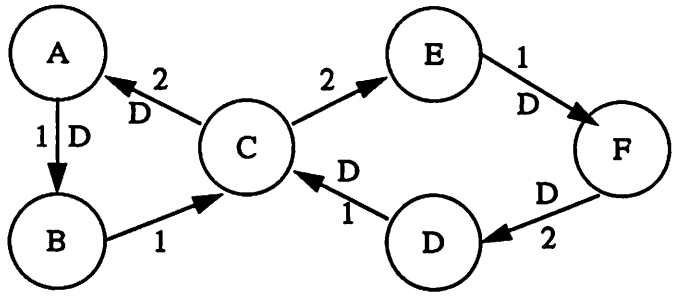


Fig 10. A graph with a 2-delay and 3-delay critical cycles

$$T(J) = \begin{cases} \lambda + \frac{0.5}{J} & J = 1, 3, 5, \ldots \\ \lambda + \frac{0.0}{J} = \lambda & J = 2, 4, 6, \ldots \end{cases}$$

Notice that although $\lambda^{-1}B$ has a transient of two before it becomes periodic, the critical path is cyclic immediately. This will not be true in general. However, we have observed that the transient for the critical path is usually less than the transient for the matrix.

Example 2: This is an interesting example where the cyclicity is one even though there are no critical cycles with one delay in the HSDF graph. The graph for this example appears in figure 10. This graph has $\lambda = 2$. The graph G_B is shown in figure 11(a), and the critical graph in figure 11(b). There are two critical cycles in the HSDF graph and three critical cycles in G_B of lengths 2,3, and 5. Hence the cyclicity is $gcd(2, 3, 5) = 1$. By simulation, we find that $T = 8$. However, there is no transient for $M_B(J)$ and it is equal to 2 for every J. The iteration period is given by $T(J) = \lambda + \frac{2}{J}$.

Example 3: This example illustrates a case where the cyclicity of the critical path is smaller than the cyclicity of the matrix. The graph is depicted in figure 12. The maximum cycle mean for this graph is 6. The graph G_B is given in figure 13, and the critical graph in figure 14. From the critical graph, we see that the cyclicity is 6. Through simulation, T turns out to be 8. The critical path has no transient and is cyclic with cyclicity 3 as shown by

$$M_B(1) = 8, M_B(2) = 5, M_B(3) = 4, M_B(4) = 8, M_B(5) = 5, M_B(6) = 4$$

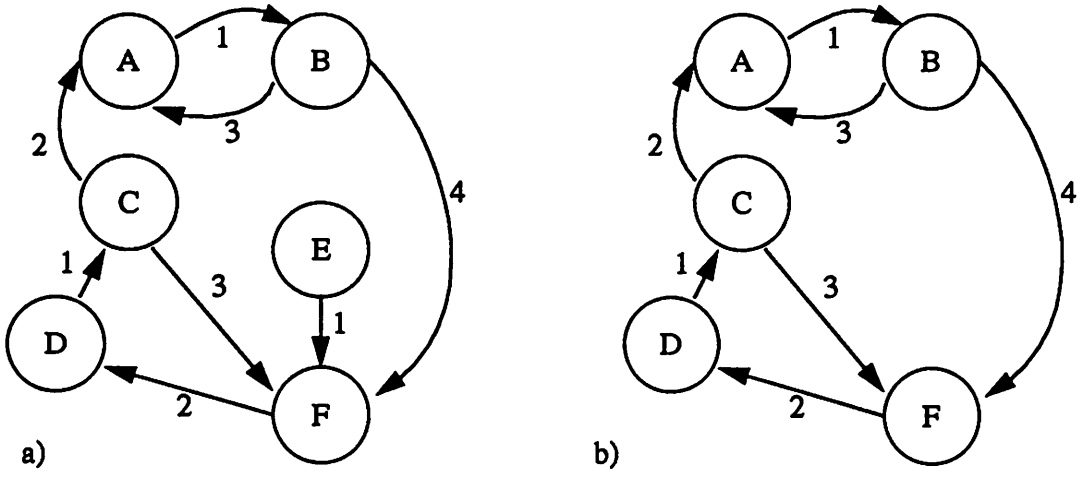


Fig 11. a) Graph of the B-matrix for graph in fig. 10. b) Critical graph of graph in a).

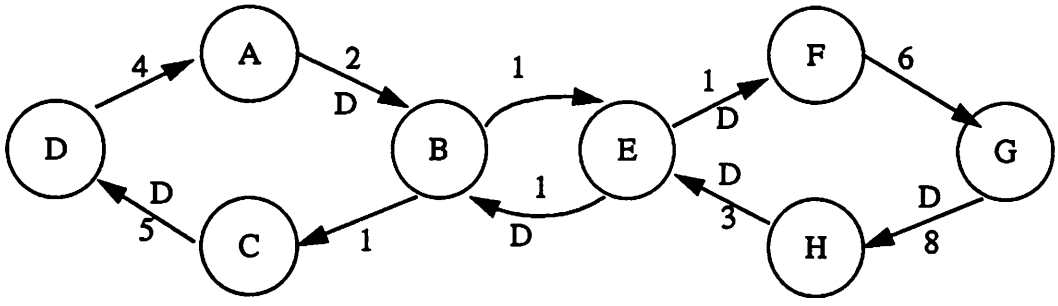


Fig 12. The graph for example 3.

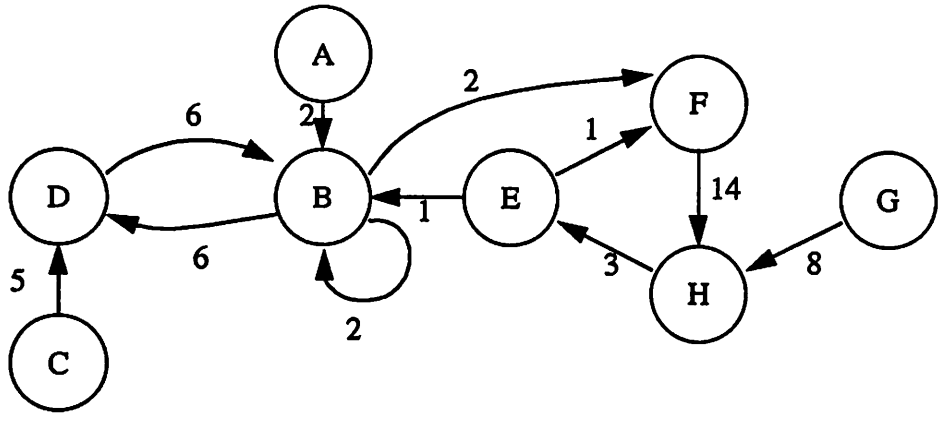


Fig 13. The graph of the B-matrix for graph in fig. 12

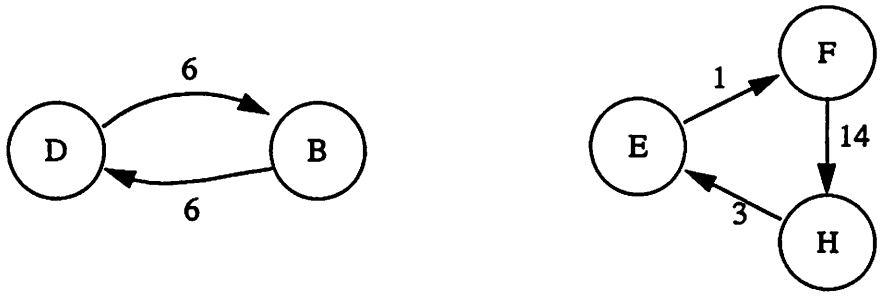


Fig 14. Critical graph for graph in fig. 13.

The iteration period is thus

$$T(J) = \begin{cases} \lambda + \frac{8}{J} & J = 1, 4, 7, \ldots \\ \lambda + \frac{5}{J} & J = 2, 5, 8, \ldots \\ \lambda + \frac{4}{J} & J = 3, 6, 9, \ldots \end{cases}$$

From the above equation, we can see that we will get the best performance if we choose a blocking factor that is a multiple of 3.

Example 4: There has been no transient in the critical path in all of the previous examples. This example shows that the transient can exist and can be quite large. The HSDF graph, the graph of B, and the critical graph appear in figures 15(a,b,c). We have $\lambda = max\left(100, \frac{55}{2}, 99\right) = 100$. The cyclicity is 1. Simulation gives us $T = 146$ but the transient for the critical path turns out to be 50. Thus, blocking factors greater than 50 are rate-optimal. The reason for the long transient is that we have two interacting loops, one of which has a cycle mean very close to the maximum cycle mean. It takes a long time for the effect of the critical cycle to start dominating.

Example 5: The next two examples show the effect of retiming [19] on the blocking factor. Consider the HSDF graphs, their B-graphs, and critical graphs depicted in figure 16(a,b). Both graphs have $\lambda = 35$. The one in figure 16(b) is a retimed version of the graph in figure 16(a). Using the same methods as in the previous examples, we find that the iteration period for each of the graphs is given by

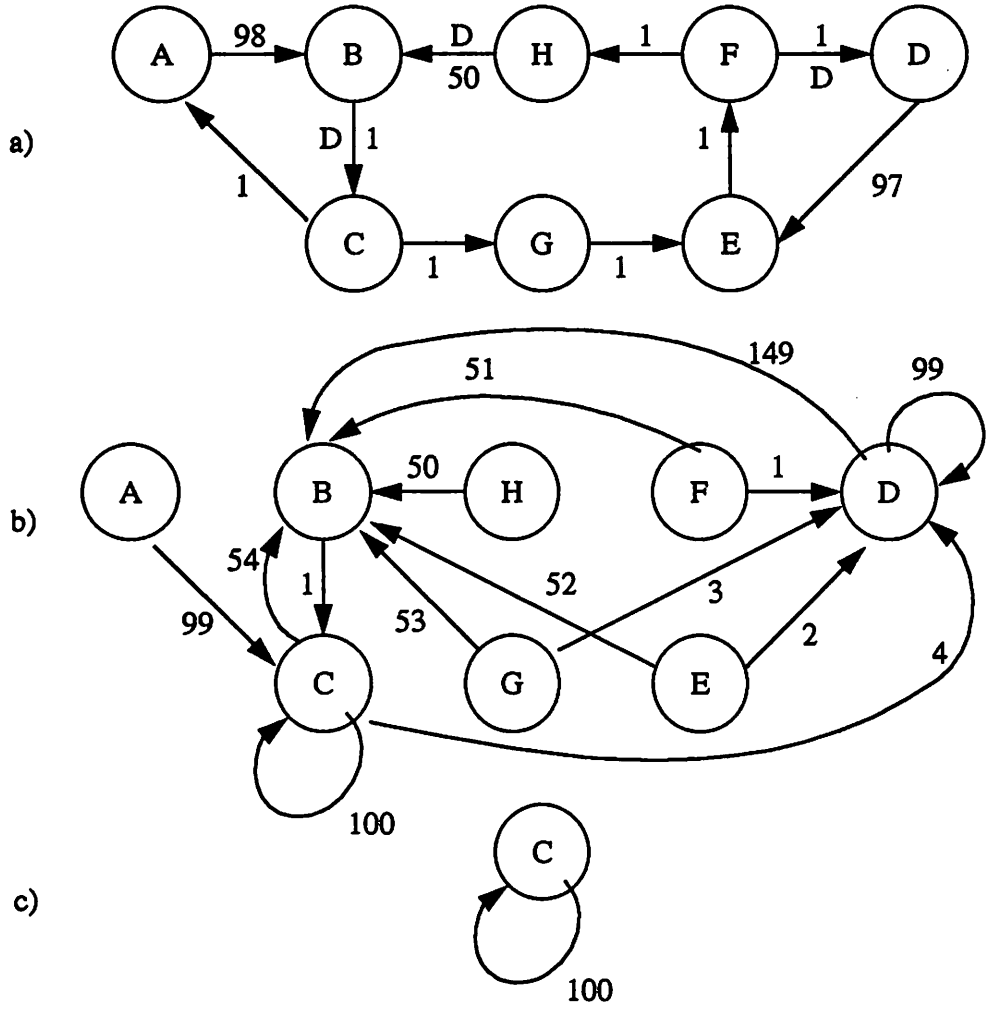


Fig 15. a) An HSDF graph where the transient is large. b) The graph of the B-matrix. c) The critical graph of b)

$$T_a(J) = \begin{cases} \lambda + \frac{20}{J} & J = 2, 4, 6, \ldots \\ \lambda + \frac{25}{J} & J = 1, 3, 5, \ldots \end{cases}$$

$$T_b(J) = \begin{cases} \lambda + \frac{0}{J} & J = 2, 4, 6, \ldots \\ \lambda + \frac{5}{J} & J = 1, 3, 5, \ldots \end{cases}$$

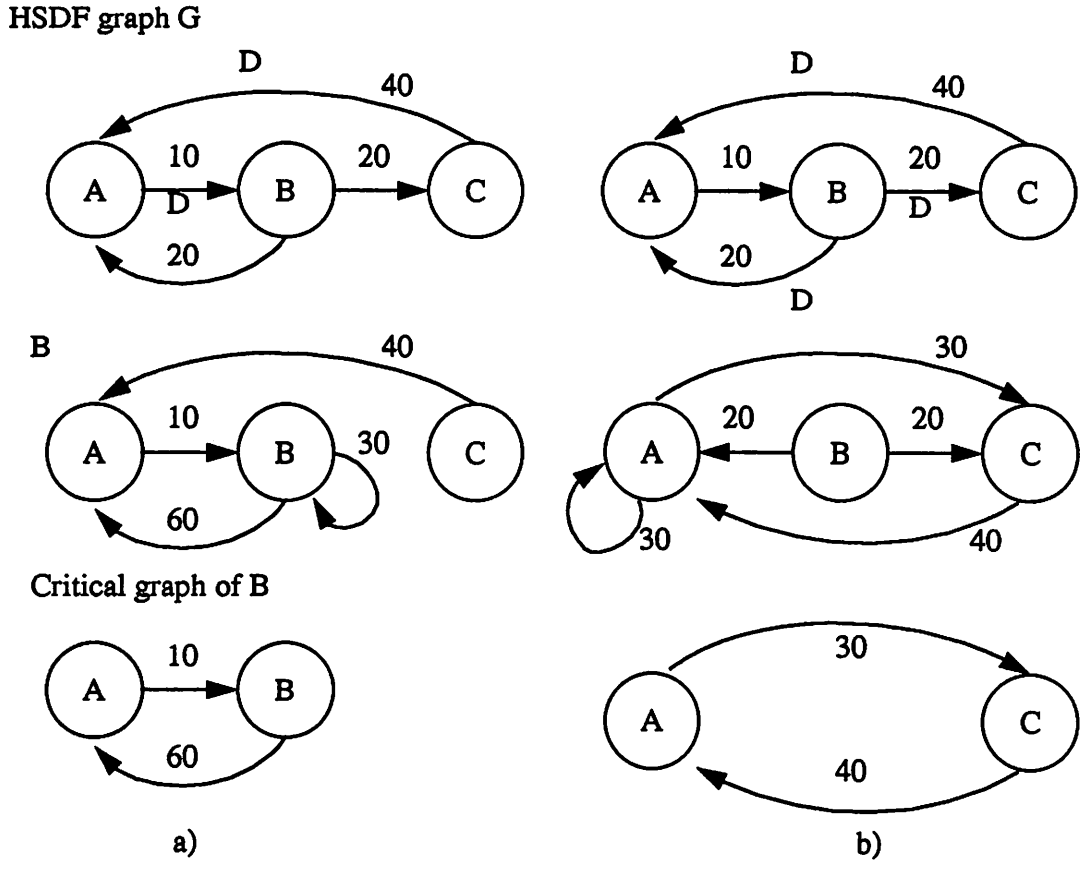


Fig 16. a) An HSDF graph that does not have a rate-optimal blocking factor, b) A retimed version of graph in a) that does have a rate-optimal blocking factor.

It is seen that even blocking factors are rate-optimal in the retimed graph.

Example 6: The example in figure 17, taken from [2], shows that sometimes, no retiming results in a rate-optimal blocking factor. Every retiming of the graph results in an $M_B(J)$ that is greater than the one for the original graph. Since the graph is perfect-rate (every loop has one delay), the cyclicity is one and we find that

$$T(J) = \lambda + \frac{6}{J} \qquad \forall J \geq 1$$

If every node in the graph has unit execution time, then it has been shown in [4] that there is a retiming that gives a rate-optimal blocking factor.

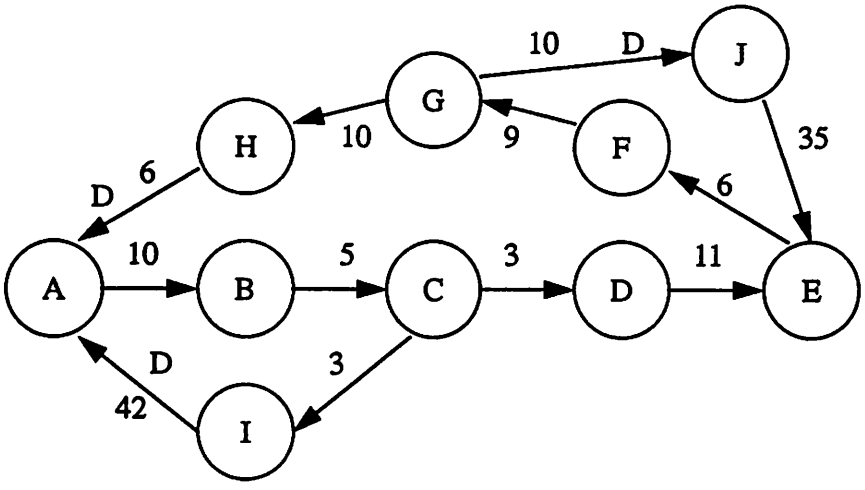


Fig 17. No retiming results in a rate-optimal blocking factor for this graph.

Example 7: The next example illustrates the analysis for a non-strongly connected HSDF graph (figure 18). This graph has a sink node, H, and hence the dummy sink node S is shown explicitly. In graphs that are not strongly connected, we speak of two transients. The first transient is due to the fact that a path that does not intersect any cycle in the graph may be critical for a few unfoldings. As has been discussed before, this situation can only remain true for the first few unfoldings. As the number of unfoldings increases, the critical path will be one that traverses some critical cycles. Hence, the first transient is the number of unfoldings required for some path that does intersect a cycle to become critical. As can be seen from the figure, the path $FGHS$ is not accessible in its entirety by any m.s.c.s of the graph. Since this path has one delay, its length stays constant for all blocking factors greater than two. The length of the transient is therefore the number of unfoldings needed for the weight of $FGHS$ to become non-critical. By evaluating equation 5 for the first two blocking factors, we determine the weight of this path to be 40; hence an upper bound on the transient is $40/\lambda = 40/5 = 8$ unfoldings. The second transient is the same as the transient encountered for strongly-connected graphs; namely, that arising from the interaction of non-critical cycles with critical cycles as in example 4. Of-course, the second transient may overlap the first transient so that the length of the overall transient (i.e., the number of unfoldings before the periodic regime is reached) need not be equal to the sum of the two. In general, we would need to find the strongly connected components of the graph and identify nodes that neither

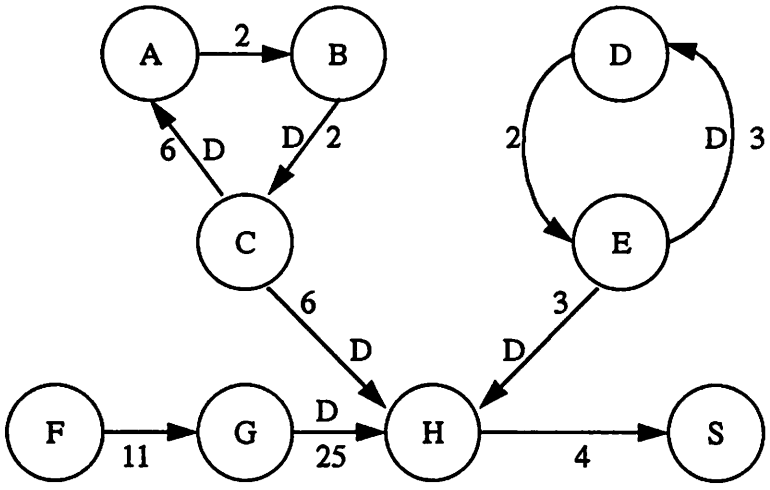


Fig 18. A non-strongly-connected HSDF graph.

reach nor are reachable by any of the strongly connected components. We would then use equation 5 to calculate $C(J)$ for successive unfolding factors. If the maximal element in this matrix (the critical path), at some unfolding factor, is for some node that can reach or is reachable by a strongly connected component, then we know that we have finished the first transient. We then continue until we see cyclic behaviour. The advent of cyclic behaviour in the maximum weight signals the end of the second transient.

Continuing with the example, we determine the B graph (figure 19(a)). B is connected, but the critical graph of B has two strongly connected components (figure 19(b)). These have cyclicities of 1 and 2; hence the cyclicity of the critical path is two. We compute $B^{J-1}A^{*}_{N0}$ for $J \geq 3$ and find that

$$M_B(J) = \begin{cases} 6 & J \text{ odd} \\ 5 & J \text{ even} \end{cases} \qquad \text{(this is } M_B(J) \text{ from equation 12).}$$

Therefore, we conclude that even blocking factors greater than 8 are rate-optimal.

In summary, for a non-strongly connected HSDF graph, we first find the strongly connected components of the graph and delete them. The resulting graph is acyclic, and we compute the path that has the largest weight. Then, an upper bound on the first transient is the weight of this path divided by the maximum cycle mean. We compute $C(J)$ using equation 5 for values of J past the bound we have until we see the maximum element reach the periodic regime.

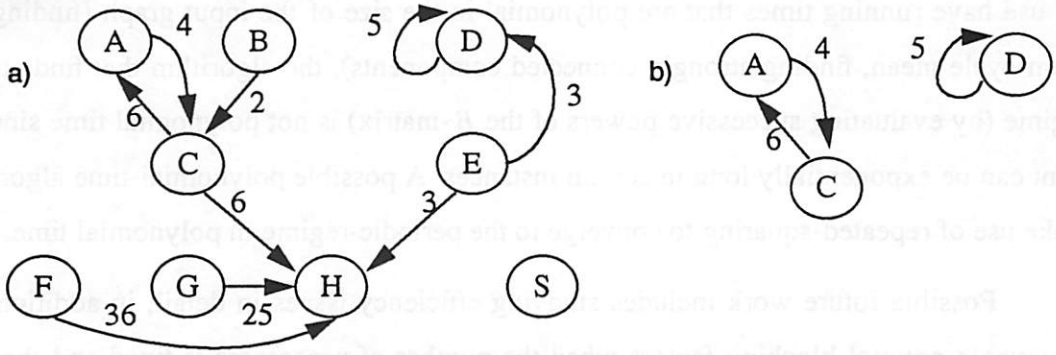


Fig 19. **a)** The graph of the B-matrix for fig. 18. **b)** The critical graph of a).

6 Conclusion

In this paper, we have developed a formulation for the problem of finding rate-optimal blocking factors for blocked, non-overlapped schedules. We have used a max-algebra formulation to show that the critical path in a J-unfolded HSDF graph becomes cyclic as J becomes large enough. We have shown that it is possible to determine this cyclicity by analyzing the critical graph of the B-matrix, a matrix that arises out of the model. The cyclicity of the critical path implies that we have to examine only a finite number of unfoldings to determine whether a rate-optimal unfolding exists. This number is equal to the sum of the cyclicity and a transient. Unfortunately, the transient can be quite large sometimes.

While this paper has contributed to our theoretical understanding of the dynamics of increasing the blocking factor, construction of rate-optimal blocked schedules for multiprocessors is still a difficult problem. In addition, we often have a fixed, finite number of processors available; finding the best blocking factor when we are processor constrained is an open problem of more practical interest. The results in this paper certainly provide us with a upper bound on the performance we can expect with a finite number of processors.

The issue of algorithmic efficiency has not been dealt with in this paper. Clearly, the technique of graph expansion alone can cause an exponential blow-up in the size of the actual graphs that are dealt with if a large number of delays are present on any arc. While some algorithms that

we use have running times that are polynomial in the size of the input graph (finding the maximum cycle mean, finding strongly connected components), the algorithm that finds the periodic regime (by evaluating successive powers of the B-matrix) is not polynomial time since the transient can be exponentially long in certain instances. A possible polynomial-time algorithm might make use of repeated-squaring to converge to the periodic-regime in polynomial time.

Possible future work includes studying efficiency issues in detail, in addition to finding achievable optimal blocking factors when the number of processors is fixed and there is a cost associated with interprocessor communication.

7 Acknowledgments

We are grateful for Sriram's help in pointing out relevant work in this area, and for many useful discussions.

8 References

[1] E. A. Lee, "A Coupled Hardware and Software Architecture for Programmable Digital Signal Processors", Ph.D. Thesis, UC-Berkeley, 1986

[2] K. K. Parhi, D. G. Messerschmitt, "Static Rate-Optimal Scheduling of Iterative Data-Flow Programs via Optimum Unfolding", IEEE Transactions on Computers, February 1991

[3] L. F. Chao, E. M. Sha, "Static Scheduling for Synthesis of DSP Algorithms on Various Models", Technical Report, Princeton University, 1993

[4] L. F. Chao, E. M. Sha, "Unfolding and Retiming Data-Flow DSP Programs for RISC Multiprocessor Scheduling", ICASSP 1992

[5] G. Cohen, D. Dubois, J. P. Quadrat, M. Viot, "A Linear-System-Theoretic View of Discrete-Event Processes and its Use for Performance Evaluation in Manufacturing", IEEE Trans. on Automatic Control, March 1985

[6] F. Baccelli, G. Cohen, G. J. Olsder, J. P. Quadrat, "Synchronization and Linearity", Prentice Hall, 1993

[7] G. Sih, "Multiprocessor Scheduling to Account for Interprocessor Communication", Ph.D. Thesis, UC Berkeley, 1991

[8] J. B. Dennis, "First Version of a Data Flow Procedure Language", MAC Technical Memo. 61, Laboratory for Computer Science, MIT, May 1975

[9] R. Rieter, "Scheduling Parallel Computations", Journal of the ACM, (14), 1968

[10] T. L. Adam, K. M. Chandy, J. R. Dickson, "A Comparison of List Schedules for Parallel Processing Systems", Comm. ACM 17 (12), December 1974

[11] T. C. Hu, "Parallel Sequencing and Assembly Line Problems", Operations Research 9 (6), 1961

[12] W. H. Kohler, "A Preliminary Evaluation of the Critical Path Method for Scheduling Tasks on Multiprocessor Systems", IEEE Trans. on Computers, December 1975

[13] D. A. Schwartz, "Synchronous Multiprocessor Realizations of Shift-Invariant Flow Graphs", Ph.D. Thesis, Georgia Tech, 1985

[14] G. Liao, G. Gao, E. Altman, V. K. Agarwal,"A Comparative Study of DSP Multiprocessor List Scheduling Heuristics", Technical Report, McGill University, 1993

[15] R. Karp, "A Note on the Characterization of the Minimum Cycle Mean in a Digraph", Discrete Mathematics, 23, 1978

[16] E. A. Lee, W.-H. Ho, E. Goei, J. Bier, and S. Bhattacharyya, "Gabriel: A Design Environment for DSP", IEEE Trans. on ASSP, November, 1989.

[17] J. Buck, S. Ha, E. A. Lee, D. G. Messerschmitt, "Ptolemy: a Framework for Simulating and Prototyping Heterogeneous Systems", to appear in *International Journal of Computer Simulation*, special issue on "Simulation Software Development," January, 1994.

[18] S. Sriram, "Modelling Run-time behaviour of Static Multi-processor Schedules", Qualification Exam Proposal, UC Berkeley, June 1993.

[19] C. E. Leiserson, F. Rose, J. Saxe, "Optimizing Synchronous Circuitry by Retiming," in Proc. Third Caltech Conf. VLSI, Pasadena CA, Mar 1983

[20] R. K. Guy, "Unsolved Problems in Number Theory," New York, Springer-Verlag, pp63, 1981

[21] A. Brauer, "On a Problem Of Partitions," American Journal of Mathematics, pp299-312, 1942

[22] H. S. Wilf, "A Circle-Of-Lights Algorithm for the 'Money Changing Problem'," American Mathematical Monthly, pp562-565, August, 1978

[23] P. Erdös, R. L. Graham, "On a Linear Diophantine Problem of Frobenius," Acta Arithmetica, pp399-408, 1972