T H E U N I V E R S I T Y O F M I C H I G A N

SYSTEMS ENGINEERING LABORATORY

Department of Electrical Engineering
College of Engineering

Technical Report SEL-66-3

INITIATION TIMING IN A MODEL FOR PARALLEL COMPUTATION

by

Raymond Reiter

March, 1966

This research was supported by United States Air Force contract AF 30(602)-3546.

Abstract

This paper deals with a particular case of a model for parallel computation as formulated by Karp and Miller. A parallel computation is viewed as a directed graph in which a node represents a sequence of operations to be performed upon data on the node input branches with the results of an operation being placed upon the output branches. An operation associated with a node n may initiate only if there is at least one data item on each input branch to n. Upon initiation, n removes one data item from each input branch and upon termination, places one data item on each output branch. For such a computation graph G necessary and sufficient conditions that a set of real numbers $\{t_i^r \mid r=0,1,\ldots,$ and n_i is a node of G$\}$ represent a sequence of initiation times for the nodes n_i of G are given. A periodic set $\{t_i^r = t_i + r\gamma\}$ is given so that G computes periodically, and the minimum period π is determined in terms of the graph parameters. A maximal computation rate periodic schedule is also given for the case that G is required to compute synchronously, i.e. at integer times. Finally, in the case of a synchronous computation graph G, an analysis is given of the so-called free running execution of G and this is found to yield the maximum computation rate of G.

TABLE OF CONTENTS

1. Introduction

The concept of a computation effected in a parallel fashion entails the notion of a number of processing units with interconnected data channels, with each unit capable of performing some function, provided only that the necessary input data is available to it. A model characterizing such a system has been developed by Karp and Miller of IBM. Briefly this model is as follows:

The computation is represented as a directed graph in which a node n_i denotes an operation to be performed upon data which lie upon the input branches to n_i. The results of the operation by node n_i are placed upon the output branches of n_i. A branch, therefore, represents a data queue. With each branch b_{ij} directed from node n_i to node n_j is associated a quadruple $(A_{ij}, U_{ij}, W_{ij}, T_{ij})$, the elements of which are interpreted as

A_{ij} - the initial number of data items on branch b_{ij}.

U_{ij} - the number of data items placed on b_{ij} upon the termination of the operation associated with n_i.

W_{ij} - the number of data items removed from b_{ij} under a first-in first-out queue discipline, upon the initiation of the operation associated with node n_j.

T_{ij} - a threshold. In order to initiate the operation associated with node n_j, the queue length on b_{ij} must be greater than, or equal to T_{ij} for all input branches b_{ij} to node n_j.

Let G be a computation graph with node set $\{n_i\}$. For every branch b_{ij} let $T_{ij} = U_{ij} = W_{ij} = 1$ and let every directed loop of G contain at least one initial data item. It is known (See [3] or [5].) that such a graph represents a nonterminating computation. We seek an execution of G which is periodic in the sense that if a node n_j first initiates at time t_j, it will initiate thereafter at times $t_j + \gamma$, $t_j + 2\gamma$, ..., where γ, the period, is the same for all nodes of G. Clearly, if the computation is to be controlled by a clock signal, such an execution is desirable. We shall show in Section 2 that such an execution exists for all such computation graphs. Moreover, it is possible to define a parameter π for G such that the above periodic schedule is possible with $\gamma = \pi$ and under which G computes at the maximum possible rate.

In Sections 3 and 4 we consider various problems which arise when the frequency of the clock signal controlling the initiation of the nodes of G is a priori specified. It then turns out that the maximum computation rate periodic schedule of Section 2 is not applicable when π is not an integer. This leads us in Section 3 to define the notion of a synchronous computation graph computing under the so-called free running execution. Under this execution, a node initiates at integer times if and only if each input branch contains at least one data word. The principal result of Section 3 is that a synchronous computation graph under the free running execution computes at the maximum possible rate and that this rate is $1/\pi$. In Section 4 we provide a periodic execution for synchronous computation graphs G. This execution has the following form:

Let $\pi = \lambda/\alpha$ where λ and α are integers, and let n_i be a node of G. Then there are integers $t_i^0 < t_i^1 < \ldots < t_i^{\alpha-1} < t_i^0 + \lambda$ such that n_i initiates only at times

$$
\begin{array}{l}
t_i^0,\ t_i^1,\ \ldots,\ t_i^{\alpha-1}, \\
t_i^0+\lambda,\ t_i^1+\lambda,\ \ldots,\ t_i^{\alpha-1}+\lambda, \\
t_i^0+2\lambda,\ t_i^1+2\lambda,\ \ldots,\ t_i^{\alpha-1}+2\lambda, \\
\ldots\ldots\ldots\ldots\ldots
\end{array}
$$

Clearly, under this execution G computes at the maximum rate $1/\pi$.

2.A. A Periodic Schedule

Throughout this paper, unless stated otherwise, we shall be concerned with computation graphs G such that for each branch $b_{ij} = (n_i, n_j)$ we have $T_{ij} = W_{ij} = U_{ij} = 1$ and such that every directed loop of G contains at least one initial data item. Thus each node of G is eligible for initiation if and only if there is at least one data word on each of its input branches. With each node n_j of G, associate a positive real number τ_j, the <u>execution time</u> of n_j. τ_j is to be interpreted as follows: if n_j initiates at time t, then at time t, one data word is removed from each of the input branches to n_j, and at time $t + \tau_j$, one data word is placed upon each of the output branches of n_j. We say that a directed graph is <u>in-point connected</u> if there exists a node n such that if $n_i \neq n$, then there is a directed path from n_i to n. The node n is called an <u>in-point</u>.

Let G be an in-point connected computation graph with in-point n_1. If there is a branch from n_i to n_j, write $n_i \to n_j$, and let $A_{i,j}$ be the initial number of data words on this branch. By $\pi(n_i)$ for $n_i \neq n_1$, we shall mean a path (which exists by definition)

$$n_{i_k} \to n_{i_{k-1}} \to \dots \to n_{i_2} \to n_{i_1}$$

where $n_{i_1} = n_1$, and $n_{i_k} = n_i$.

Write

$$\sum_{r=1}^{k-1} A_{i_{r+1}, i_r} \quad \text{as} \quad \sum_{\pi(n_i)} A$$

and

$$\sum_{r=i_2}^{i_k} \tau_r \quad \text{as} \quad \sum_{\pi(n_i)} \tau .$$

We shall assume that G contains at least one directed loop.

Let L be a loop of G,

$$n_{i_1} \rightarrow n_{i_2} \rightarrow \ldots \rightarrow n_{i_k} \rightarrow n_{i_{k+1}}$$

where $n_{i_{k+1}} = n_{i_1}$.

Define
$$\pi_L = \frac{\sum_{r=i_1}^{i_k} \tau_r}{\sum_{r=1}^{k} A_{i_r, i_{r+1}}}$$

and let
$$\pi = \max_{\text{loops } L} \{\pi_L\}$$

Let $\gamma \geq \pi$ and consider the following assignment of real numbers t_i to the nodes n_i of G.

$$t_1 = 0$$
$$t_i = \min_{\pi(n_i)} [\gamma \sum_{\pi(n_i)} A - \sum_{\pi(n_i)} \tau] \quad \text{for } n_i \neq n_1 \qquad (1)$$

We shall prove (Corollary 3) that an execution exists for G such that for all nodes n_j of G, n_j initiates only at times t_j, $t_j + \gamma$, $t_j + 2\gamma$, ...,.

In [1], Cuninghame-Green considers a similar problem in which $A_{ij} = 1$ for all branches (n_i, n_j) but for which τ_i is actually a branch (rather than node) parameter a_{ij}. He defines a quantity $\lambda = \max_L \{\sum_L a/\ell\}$ where ℓ is the length of the loop L. Under the identification

$\lambda - a_{ij} = \pi A_{ij} - \tau_i$, our problem becomes formally identical with his. Cuninghame-Green poses the problem of determining a set of initial starting times for the nodes of G such that a periodic execution with period π is possible and he solves this for the special case that G is a loop. In this section we show that the assignment (1) with $\gamma = \pi$ is precisely such a set of initial starting times (Corollary 3).

<u>Lemma 1</u>

If $n_i \to n_j$, then

$$t_i + \tau_i \leq t_j + \gamma\, A_{i,j}.$$

PROOF: (1) Suppose $n_i = n_j$. Then $n_i \to n_i$ is a loop of G whence

$$\frac{\tau_i}{A_{i,i}} \leq \pi \leq \gamma,$$

i.e.

$$t_i + \tau_i \leq t_i + \gamma\, A_{i,i}.$$

(2) Suppose $n_j = n_1$. Then $n_i \to n_j$ is a path $\pi(n_i)$ and we have immediately

$$t_i \leq \gamma\, A_{i,j} - \tau_i = t_j + \gamma\, A_{i,j} - \tau_i$$

(3) Suppose $n_i = n_1$. Let $\pi(n_j)$ be a path for which

$$t_j = \gamma \sum_{\pi(n_j)} A - \sum_{\pi(n_j)} \tau$$

Then

$$t_j = \gamma A_{i,j} + \gamma \sum_{\pi(n_j)} A - \tau_l - \sum_{\pi(n_j)} \tau - \gamma A_{i,j} + \tau_l$$

$$= \gamma \sum_L A - \sum_L \tau - \gamma A_{i,j} + \tau_l$$

where we write L as the loop $n_l \rightarrow \pi(n_j)$, and $\sum_L A$ as the sum around L of the initial branch weights of L, $\sum_L \tau$ as the sum of the execution times of the nodes of L.

But

$$\frac{\sum_L \tau}{\sum_L A} \leq \pi \leq \gamma$$

i.e.

$$\gamma \sum_L A - \sum_L \tau \geq 0.$$

Hence

$$t_j \geq \tau_l - \gamma A_{i,j} = t_l + \tau_l - \gamma A_{i,j}.$$

(4) Suppose $n_i \neq n_j$, $n_i \neq n_l$, $n_j \neq n_l$. Let $\pi(n_j)$ be a path for which

$$t_j = \gamma \sum_{\pi(n_j)} A - \sum_{\pi(n_j)} \tau.$$

If $\pi(n_j)$ does not contain n_i, then $\pi(n_i): n_i \rightarrow \pi(n_j)$, is a path from n_i to n_l whence

$$t_i \leq \gamma \sum_{\pi(n_i)} A - \sum_{\pi(n_i)} \tau$$

$$= \gamma A_{i,j} + \gamma \sum_{\pi(n_j)} A - \tau_i - \sum_{\pi(n_i)} \tau$$

$$= \gamma A_{i,j} - \tau_i + t_j .$$

Otherwise $\pi(n_j)$ contains n_i. Then $\pi(n_j)$ has the form

$$n_{i_s} \to n_{i_{s-1}} \to \ldots \to n_{i_r} \to \ldots \to n_{i_1}$$

where $n_{i_s} = n_j$, $n_{i_r} = n_i$, and $n_{i_1} = n_1$. Moreover $n_{i_r} \to n_{i_s}$. Then

$$t_j = \gamma \sum_{\pi(n_j)} A - \sum_{\pi(n_j)} \tau$$

$$= \gamma A_{i,j} + \gamma \sum_{\pi(n_j)} A - \tau_i - \sum_{\pi(n_j)} \tau + \tau_i - \gamma A_{i,j}$$

$$= \gamma \sum_L A - \sum_L \tau + \gamma \sum_{\pi(n_i)} A - \sum_{\pi(n_i)} \tau + \tau_i - \gamma A_{i,j}$$

where we write L as the loop

$$n_{i_r} \to n_{i_s} \to n_{i_{s-1}} \to \ldots \to n_{i_r}$$

and $\pi(n_i)$ is the path from n_{i_r} $(=n_i)$ to n_1

$$n_{i_r} \to n_{i_{r-1}} \to \ldots \to n_1 .$$

But

$$\gamma \sum_{\pi(n_i)} A - \sum_{\pi(n_i)} \tau \geq t_i$$

and

$$\frac{\sum_L \tau}{\sum_L A} \leq \pi \leq \gamma$$

i.e.

$$\gamma \sum_L A - \sum_L \tau \geq 0$$

Hence

$$t_j \geq t_i + \tau_i - \gamma A_{i,j}.$$

Define $x(n,t) = 0$ if and only if node n initiates for the first time at some time $t' \geq t$. For $k=1,2,\ldots,$ define $x(n,t) = k$ if and only if node n initiates for the k-th time at some time $t' < t$ but has not initiated for the (k+1)-th time at some time $t'' < t$. Thus $x(n,t)$ is the number of initiations of node n up to, but not including, time t. If $n_i \rightarrow n_j$, define

$$b_{ij}(t) = A_{i,j} + x[n_i,(t-\tau_i)^+] - x(n_j,t)$$

where

$$x[n_i,(t-\tau_i)^+] = x(n_i,t-\tau_i) \text{ if } n_i \text{ does not initiate at time } t-\tau_i.$$
$$= x(n_i,t-\tau_i) + 1 \text{ if } n_i \text{ does initiate at time } t-\tau_i.$$

$b_{i,j}(t)$ is interpreted as the number of data words on branch (n_i,n_j) at time t.

Theorem 2

Let G be a computation graph. With each node n_i of G associate a set of real numbers $\{t_i^r \mid t_i^r < t_i^{r+1}, \quad r=0,1,\ldots,\}$.

There exists an execution of G such that n_j initiates only at times t_j^0, t_j^1, ..., t_j^r, ..., if and only if for all branches (n_i,n_j) of G we have

$$t_i^r + \tau_i \leq t_j^{r+A_{ij}}.$$

PROOF:

$\Longrightarrow$

Let $n_i \to n_j$ and suppose there exists r such that $t_i^r + \tau_i > t_j^{r+A_{ij}}$. Then

$$t_i^r + \tau_i = t_j^{r+A_{ij}} + \epsilon_{ij}$$

where $\epsilon_{ij} > 0$. Therefore

$$\begin{aligned} b_{ij}(t_j^{r+A_{ij}}) &= A_{ij} + x[n_i,(t_j^{r+A_{ij}} - \tau_i)^+] - x(n_j,t_j^{r+A_{ij}}) \\ &= A_{ij} + x[n_i,(t_i^r - \epsilon_{ij})^+] - (r+A_{ij}) \\ &\leq A_{ij} + r - (r+A_{ij}) \\ &= 0 \end{aligned}$$

which contradicts the condition that n_j initiates at time $t_j^{r+A_{ij}}$.

$\Longleftarrow$

We shall prove, by induction on n, that for all nodes n_j of G, n_j initiates at time t_j^n. Let the distinct numbers of the set $\{t_i^0\}$ be $T_1 < T_2 < \ldots < T_m$. For the case n=0, we apply induction on the subscript r of T_r.

Suppose r=1 and let $t_j^0 = T_1$. Then if $n_i \to n_j$, we have $t_i^0 + \tau_i \leq t_j^{A_{ij}}$. If $A_{ij} = 0$, then $t_i^0 + \tau_i \leq t_j^0 = T_1$ and since $\tau_i > 0$, $t_i^0 < T_1$ a contradiction. Hence $A_{ij} \geq 1$ for all branches (n_i,n_j) so that n_j can initiate at time t_j^0.

Assume, for all nodes n_i for which $t_i \leq T_r$, $r < m$, that n_i initiates at time t_i^0. Suppose $t_j^0 = T_{r+1}$. If $A_{ij} > 0$ for all branches (n_i,n_j) of G then n_j can initiate at time t_j^0. Otherwise $A_{ij} = 0$ for some branch (n_i,n_j) so that

$$t_i^0 + \tau_i \leq t_j^{A_{ij}} = t_j^0 = T_{r+1}.$$

i.e. $t_i < T_{r+1}$ so that $t_i = T_q$ for some $q \leq r$. By the induction hypothesis on r, n_i initiates at time t_i^0. But then n_i terminates at time $t_i^0 + \tau_i \leq t_j^0$ so that at time t_j^0 the branch (n_i,n_j) contains a data word. This is true for all input branches to n_j so that n_j can initiate at time t_j^0. This completes the induction on r and hence establishes the result for n=0.

Now assume, for all nodes n_j of G, that n_j initiates at times $t_j^0,t_j^1,\ldots,t_j^n$. We prove that n_j initiates at time t_j^{n+1}.

Let $T_1 < T_2 < \ldots < T_m$ be the distinct members of the set $\{t_i^{n+1}\}$. Again we apply induction on the subscript r of T_r.

Suppose r=1 and let $t_j^{n+1} = T_1$. We prove that $A_{ij} > 0$ for all branches (n_i,n_j). For if not, there exists a branch (n_i,n_j) such that $A_{ij} = 0$. Then

$$t_i^{n+1} + \tau_i \leq t_j^{n+1+A_{ij}} = t_j^{n+1} = T_1.$$

i.e. $t_i^{n+1} < T_1$ a contradiction.

Now for any branch (n_i,n_j)

$$b_{ij}(t_j^{n+1}) = A_{ij} + x[n_i,(t_j^{n+1} - \tau_i)^+] - x(n_j, t_j^{n+1}).$$

Suppose $n+1 < A_{ij}$. Then

$$b_{ij}(t_j^{n+1}) \geq A_{ij} - x(n_j, t_j^{n+1})$$

$$= A_{ij} - (n+1) \text{ by the induction hypothesis on } n$$

$$> 0.$$

Otherwise $n+1 \geq A_{ij}$ and $t_i^{n+1-A_{ij}}$ is defined, so that

$$t_i^{n+1-A_{ij}} + \tau_i \leq t_j^{n+1}.$$

Therefore

$$b_{ij}(t_j^{n+1}) \geq A_{ij} + x[n_i,(t_i^{n+1-A_{ij}})^+] - x(n_j, t_j^{n+1}).$$

But $A_{ij} \geq 1$ so that $n+1 - A_{ij} \leq n$. Then by the induction hypothesis on n,

$$x[n_i,(t_i^{n+1-A_{ij}})^+] = n+1 - A_{ij} + 1.$$

Also, by the induction hypothesis on n, $x(n_j, t_j^{n+1}) = n+1$. Hence $b_{ij}(t_j^{n+1}) \geq 1$ and n_j can initiate at time t_j^{n+1}.

Assume, then, for all nodes n_i for which $t_i^{n+1} \leq T_r$, $r < m$, that n_i initiates at times t_i^0, t_i^1, ..., t_i^{n+1}. Let n_j be a node for which $t_j = T_{r+1}$. If $A_{ij} > 0$ for all input branches (n_i,n_j) of n_j, then by the same argument as above for r=1,

$$b_{ij}(t_j^{n+1}) \geq 1.$$

and n_j can initiate at time t_j^{n+1}. Otherwise $A_{ij} = 0$ for some branch (n_i,n_j). Then

$$t_i^{n+1} + \tau_i \leq t_j^{n+1} = T_{r+1}$$

i.e. $t_i^{n+1} < T_{r+1}$ so that $t_i = T_q$ for some $q \leq r$. By the induction hypothesis on r, n_i initiates at times $t_i^0, t_i^1, \ldots, t_i^{n+1}$. Then

$$b_{ij}(t_j^{n+1}) = x[n_i, (t_j^{n+1} - \tau_i)^+] - x(n_j, t_j^{n+1}).$$

But $t_j^{n+1} - \tau_i \geq t_i^{n+1}$. Therefore

$$b_{ij}(t_j^{n+1}) \geq x[n_i, (t_i^{n+1})^+] - x(n_j, t_j^{n+1}) = (n+2) - (n+1)$$

by the induction hypothesis on n. Hence all input branches to n_j contain at least one data word at time t_j^{n+1} so that n_j can initiate at times t_j^{n+1}.

This completes the induction on r and hence on n.

<u>Corollary 3</u>

Let G be in-point connected and $\gamma \geq \pi$. Then there exists an execution of G under which, for all nodes n_j of G, n_j initiates only at times $t_j, t_j + \gamma, t_j + 2\gamma, \ldots,$.

PROOF:

For each node n_k of G put $t_k^r = t_k + r\gamma \quad r=0,1,\ldots,$ where t_j is obtained from the assignment (1). Then by Lemma 1, $t_i^r + \tau_i \leq t_j^{r+A_{ij}}$ for every branch (n_i, n_j). The result now follows from Theorem 2.

For an arbitrary nonterminating computation graph G, let us define the <u>computation rate of a node</u> n_j of G to be

$$\rho_j = \lim_{t \to \infty} \frac{x(n_j, t)}{t}$$

if this limit exists. In the case of a computation graph G under a periodic execution with period p, we have $\rho_j = 1/p$ independent of n_j. It thus makes sense in this case to define the <u>computation rate of G</u> to be $\rho_G = 1/p$.

Theorem 4

Let G be in-point connected. With $\gamma = \pi$, the assignment (1) yields the maximum computation rate of G under any periodic execution.

PROOF: Consider a periodic execution under which, for each node n_j of G, n_j initiates at times t_j, t_j+p, t_j+2p, ...,. Let L be a loop of G for which $\pi_L = \pi$, and let A be the sum of the number of data words initially on the branches of L. Then for any node n_j of L,

$$x(n_j, t_j + Ap) - x(n_j, t_j) = A$$

Thus, in time Ap, a data word on L passes once around the loop. But the minimum time in which this can be done is $\sum_L \tau_j$; i.e. $Ap \geq \sum_L \tau_j$ i.e. $p \geq \pi_L = \pi$.

Note that the proof of Theorem 4 reveals that, with $\gamma = \pi$, the assignment (1) yields a periodic execution of G such that if a node n_j lies on a loop L with $\pi_L = \pi$, then ρ_j is the maximum computation rate of the node n_j under any (periodic or not) proper execution of G.

In the actual calculation of times t_j for a given graph G, in the case $\gamma = \pi$, the following observation will be found to be useful: If $n_i \to n_j$, and n_i, n_j are on a loop L for which $\pi_L = \pi$, then

$$t_j = t_i + \tau_i - \pi A_{i,j}.$$

PROOF: Let L be

$$n_i \to n_j \to n_k \to \ldots \to n_r \to n_i.$$

Then by Lemma 1

$$t_j \geq t_i + \tau_i - \pi A_{i,j}$$

$$t_k \geq t_j + \tau_j - \pi A_{j,k}$$

$$\cdots$$

$$t_i \geq t_r + \tau_r - \pi A_{r,i}$$

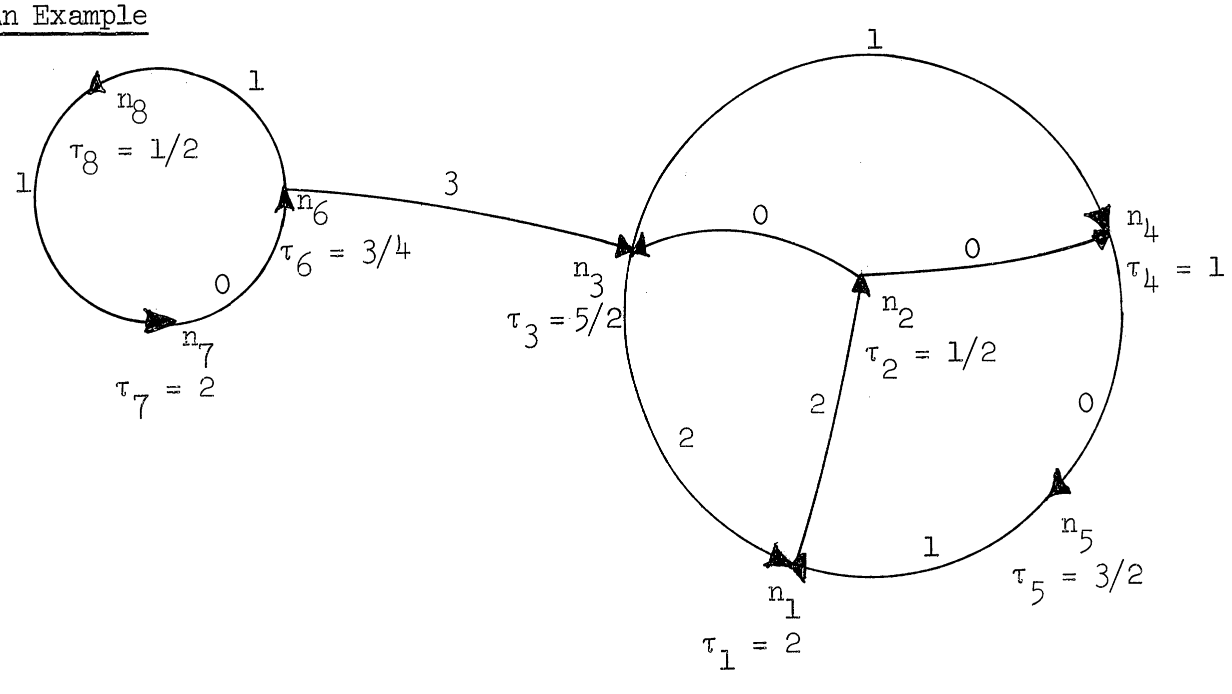
Therefore

$$0 \geq \sum_L \tau - \pi \sum_L A = 0$$

i.e. equality must hold for each of the above inequalities.

Observe that all of the results of this section have been proved under the assumption that a node n_j can initiate at any time that there is sufficient data on all of its input branches, i.e. regardless of whether or not n_j has terminated its previous initiation. If we wish a model in which no node can initiate unless it has terminated its previous execution, we can obtain it by simply adding a branch (n_j, n_j) with $A_{j,j} = 1$ for all nodes n_j. We call such a branch a <u>self-loop</u>.

<u>An Example</u>



The initial numbers of data words is shown on the branches. $\pi = 15/8$ corresponding to the loop

$$n_1 \to n_2 \to n_3 \to n_4 \to n_5 \to n_1.$$

Then we have immediately,

$$\begin{aligned} t_1 &= 0 \\ t_2 &= t_1 + \tau_1 - \pi\, A_{12} = -\,7/4 \\ t_3 &= t_2 + \tau_2 - \pi\, A_{23} = -\,5/4 \\ t_4 &= t_3 + \tau_3 - \pi\, A_{34} = -\,5/8 \\ t_5 &= t_4 + \tau_4 - \pi\, A_{45} = 3/8 \end{aligned}$$

Since (n_6, n_3) is the only path from n_6 to n_3, we have

$$t_6 = \pi\, A_{63} - \tau_6 + t_3 = 29/8.$$

Similarly,

$$\begin{aligned} t_7 &= \pi\, A_{76} - \tau_7 + t_6 = 13/8 \\ t_8 &= \pi\, A_{87} - \tau_8 + t_7 = 3. \end{aligned}$$

We note that $\pi < \tau_1, \tau_3, \tau_7$ so that n_1, n_3 and n_7 initiate at times when they have not yet terminated their previous initiation. If this is undesirable, we can place self-loops on n_1, n_3 and n_7. It then follows that for this new graph,

$$\pi = \max\,(\tau_1, \tau_3, \tau_7) = 5/2.$$

The corresponding t_i are: $t_1 = 0$ $t_2 = -\,1/2$ $t_3 = 0$ $t_4 = 0$

$t_5 = 1$ $t_6 = 27/4$ $t_7 = 19/4$ $t_8 = 27/4.$

B. Extension to Weakly Connected Directed Graphs

A directed graph G is said to be weakly connected if the undirected graph obtained from G by removing the arrows on the branches of G is connected. We shall show how the above periodic schedule for in-point-connected computation graphs can be extended to weakly connected directed graphs. A subgraph G' of G is a maximal in-point connected subgraph if there is no subgraph properly containing G' which is in-point connected. If H is a subgraph of G, by G-H we mean that subgraph of G generated by those nodes of G which are not nodes of H. If H' is another subgraph of G, H+H' is that subgraph of G generated by the nodes of H and H'.

Let G be a weakly connected computation graph and G' a maximal in-point connected subgraph of G. If G' = G, the results of Section 2A are sufficient. Otherwise let G" be a maximal in-point connected subgraph of G-G'. It follows that in G, there is no path from a node of G" to a node of G'. Define a set of ordered node pairs

$$N' = \{(n'_k, n''_\ell) \mid n'_k \text{ and } n''_\ell \text{ are nodes of } G' \text{ and } G'' \text{ respectively, and } n'_k \rightarrow n''_\ell\}.$$

Assign real numbers t_i to the nodes n'_i of G' by (1) of Section 2A, treating G' as a computation graph by itself. Similarly, assign real numbers t''_j to the nodes n''_j of G". Then, by Lemma 1, for all nodes n'_i, n'_j of G' such that $n'_i \rightarrow n'_j$

$$t_i + \tau_i \leq t_j + \gamma A_{ij}$$

Similarly, for all nodes n''_k, n''_ℓ of G" such that $n''_k \rightarrow n''_\ell$

$$t''_k + \tau_k \leq t''_\ell + \gamma A_{k\ell}$$

Define a real number α'' as follows:

If $N' = \emptyset$, $\alpha'' = 0$.

Otherwise $\alpha'' = \max_{(n'_k, n''_\ell) \in N'} \{t_k + \tau_k - t''_\ell - \gamma A_{k\ell}\}$

For all n''_ℓ of G'' define

$$t_\ell = t''_\ell + \alpha''.$$

Then, clearly, for all nodes n_i, n_j of $G' + G''$

$$t_i + \tau_i \leq t_j + \gamma A_{ij}.$$

In general, suppose $\mathcal{G} = G' + G'' + \ldots + G^{(r)}$ where $G^{(i)}$ is a maximal in-point connected subgraph of $G - (G' + G'' + \ldots + G^{(i-1)})$, $i=1,2,\ldots,r$, and that for all nodes n_i, n_j of $\mathcal{G}$ such that $n_i \to n_j$ there exist real numbers t_i, t_j for which

$$t_i + \tau_i \leq t_j + \gamma A_{ij}.$$

If $\mathcal{G} = G$, then we are through. Otherwise let $G^{(r+1)}$ be a maximal in-point connected subgraph of $G - \mathcal{G}$. Then in G there is no path from a node of $G^{(r+1)}$ to a node of $\mathcal{G}$. Define

$N^{(r)} = \{(n_k, n_\ell^{(r+1)}) \mid n_k$ and $n_\ell^{(r+1)}$ are nodes of $\mathcal{G}$ and $G^{(r+1)}$ respectively, and $n_k \to n_\ell^{(r+1)}\}$.

Assign real numbers $t_i^{(r+1)}$ to the nodes $n_i^{(r+1)}$ of $G^{(r+1)}$ by (1) of Section 2A. Define a real number $\alpha^{(r+1)}$ as follows:

If $N^{(r)} = \emptyset$, $\alpha^{(r+1)} = 0$

Otherwise

$$\alpha^{(r+1)} = \max_{(n_k, n_\ell^{(r+1)}) \in N^{(r)}} \{t_k + \tau_k - t_\ell^{(r+1)} - \gamma A_{k\ell}\}$$

Finally, for all $n_\ell^{(r+1)}$ of $G^{(r+1)}$ define

$$t_\ell = t_\ell^{(r+1)} + \alpha^{(r+1)}$$

Then by the same reasoning as above, for all nodes n_i, n_j of $\mathscr{G} + G^{(r+1)}$ such that $n_i \to n_j$,

$$t_i + \tau_i \leq t_j + \gamma A_{ij}$$

Since G has finitely many nodes, this process must terminate with $G = G' + \ldots + G^{(k)}$ for some k. We then have

$$t_i + \tau_i \leq t_j + \gamma A_{ij}$$

for all n_i, n_j of G such that $n_i \to n_j$. By Theorem 2, G then has a sequence of initiation times $t_i + r\gamma$, $r=0,1,\ldots$, for each node n_i of G.

C. <u>A Dual Assignment</u>

Let G be a computation graph with the property that there exists a node n_1 such that for every node $n_i \neq n_1$, there is a path $\pi(n_i)$ from n_1 to n_i. G is said to be <u>out-point connected</u>, and n_1 is called an <u>out-point</u>. Let $\pi(n_i)$ be $n_{i_1} \to n_{i_2} \to \ldots \to n_{i_{k-1}} \to n_{i_k}$ where $n_{i_1} = n_1$ and $n_{i_k} = n_i$.

Write $$\sum_{r=1}^{k-1} A_{i_r, i_{r+1}} \text{ as } \sum_{\pi(n_i)} A$$

and $$\sum_{r=i_1}^{i_{k-1}} \tau_r \text{ as } \sum_{\pi(n_i)} \tau$$

For $\gamma \geq \pi$, consider the following assignment of real numbers T_i to the nodes n_i of G:

$$T_1 = 0$$

$$T_i = \max_{\pi(n_i)} \left[\sum_{\pi(n_i)} \tau - \gamma \sum_{\pi(n_i)} A \right] \text{ for } n_i \neq n_1$$

Then it can be proved, in a manner similar to that of Section 2A, that an execution exists for G such that for all nodes n_j of G, n_j initiates only at times $T_j + r\gamma$, $r=0,1,\ldots,$.

An extension of this result to weakly connected directed graphs is possible in essentially the same fashion as in B above except that $N^{(r)}$ is taken to be

$$N^{(r)} = \{(n_\ell^{(r+1)}, n_k) \mid n_k \text{ and } n_\ell^{(r+1)} \text{ are nodes of } \mathcal{G} \text{ and } G^{(r+1)} \text{ respectively, and } n_\ell^{(r+1)} \to n_k\}.$$

Suppose G is strongly connected. Then we may choose a node n_1 to be the in-point of the assignment (1) and the out-point of the above dual assignment. It then follows that for all nodes n_i of G we have $T_i \leq t_i$. Moreover, if $\{\theta_i\}$ is a set of numbers which satisfy

$$\theta_i + \tau_i \leq \theta_j + \gamma A_{ij}, \qquad \theta_1 = 0$$

for all nodes n_i, n_j of G such that $n_i \to n_j$, i.e. such that n_k initiates only at times $\theta_k + r\gamma$, $r=0,1,\ldots,$ then

$$T_k \leq \theta_k \leq t_k$$

for all nodes n_k of G.

D. Computational Algorithms

For computation graphs G with a large number of nodes the calculation of π by inspection of all the loops of G, and of t_i by inspection of all of the paths from n_i to n_1 is a prohibitive task. Accordingly, the following more efficient algorithms are given.

(i) Calculating π

This algorithm is due to Lawler [4]. Define a recursion variable $u_{ij}^{(n)}(m)$ as follows, where the indices i,j correspond to nodes n_i, n_j of G, and m,n are recursion indices.

For m=0,1,...,

$$u_{ij}^{(0)}(m) = \tau_i \text{ if } n_i \rightarrow n_j \text{ and } m = A_{ij}$$

$$= -\infty \quad \text{otherwise.}$$

For m=0,1,..., n=1,2,...,

$$u_{ij}^{(n)}(m) = \tau_i + \max\{\max_k\{u_{kj}^{(n-1)}(m) \mid A_{ik} = 0\}, \max_k\{u_{kj}(m-A_{ik}) \mid A_{ik} > 0\}\}.$$

Here $u_{kj}(m) = u_{kj}^{(N)}(m)$ where N is such that $u_{kj}^{(N-1)}(m) = u_{kj}^{(N)}(m)$.

Let P be an upper bound on the number of consecutive branches b_{ij} of G for which $A_{ij} = 0$. Then $N \leq P$.

For all i,j, compute the quantities $u_{ij}(0)$, $u_{ij}(1)$, ..., $u_{ij}(M)$ where M is an upper bound on the sum of the A's around any loop. Then $\pi = \max_i \{u_{ii}(1), u_{ii}(2)/2, u_{ii}(3)/3, \ldots, u_{ii}(M)/M\}$. The computation in this case grows as MPN^2 where N is the number of nodes of G.

(ii) Calculating $\{t_i\}$ and $\{T_i\}$

If $n_i \to n_j$, define $a_{ij} = \gamma A_{ij} - \tau_i$. Then

$$t_i = \min_{\pi(n_i)} \{ \sum_{\pi(n_i)} a \}$$

where
$$\sum_{\pi(n_i)} a = \gamma \sum_{\pi(n_i)} A - \sum_{\pi(n_i)} \tau .$$

Moreover, for any loop L, $\sum_L a \geq 0$ since

$$\gamma \sum_L A - \sum_L \tau \geq \pi \sum_L A - \sum_L \tau \geq 0.$$

This condition guarantees the convergence of the following algorithm which is a slight modification of one given in [2].

1) Start by assigning all nodes n_i labels of the form $[-, \theta(n_i)]$ where $\theta(n_1) = 0$ (n_1 is the in-point of G), $\theta(n_i) = \infty$, $n_i \neq n_1$.

2) Search for a branch (n_i, n_j) such that $\theta(n_j) + a_{ij} < \theta(n_i)$. Here $\infty + a = \infty$). If such a branch is found, change the label on node n_i to $[n_j, \theta(n_j) + a_{ij}]$ and repeat. (That is, the new $\theta(n_i)$ is $\theta(n_j) + a_{ij}$.) If no such branch is found, terminate.

The value of $\theta(n_k)$ upon termination is t_k and the shortest path from n_k to n_1 is obtained by tracing the sequence of first co-ordinate nodes from n_k to n_1.

Suppose that G is not in-point connected and let n_1 be a node of G. Then this algorithm has the virtue of yielding minimal paths from all nodes n_i of G to n_1 whenever they exist. Thus the set of all nodes n_k for which upon termination $\theta(n_k) < \infty$ represents a maximal in-point connected subgraph of G. We can combine this observation with the results of Section 2B to yield the following procedure for computing $\{t_i\}$ for an arbitrary graph G.

1) Choose a node n_1 of G and apply the above algorithm to yield G', a maximal in-point connected subgraph of G, and a set $\{t_i \mid n_i'$ is a node of G'$\}$.

2) Determine G - G' and repeat (1) applied to G - G', yielding a maximal in-point connected subgraph G" of G - G', and a set $\{t_i'' \mid n_i''$ is a node of G"$\}$.

3) Determine the set $\{t_i \mid n_i''$ is a node of G"$\}$ by the method of Section 2B.

4) Continue in the obvious fashion until all of the nodes of G are exhausted.

The dual computation for the set $\{T_i\}$ is as follows. If $n_i \rightarrow n_j$, define $a_{ij} = \gamma A_{ij} - \tau_j$. Then for any loop L, $\sum_L a \geq 0$.

1) Start by assigning all nodes n_i labels of the form $[-, \theta(n_i)]$ where $\theta(n_1) = 0$ (n_1 is the out-point of G), $\theta(n_i) = \infty$, $n_i \neq n_1$.

2) Search for a branch (n_i, n_j) such that $\theta(n_i) + a_{ij} < \theta(n_j)$. (Here $\infty + a = \infty$). If such a branch is found, change the label on node n_j to $[n_i, \theta(n_i) + a_{ij}]$ and repeat. If no such branch is found, terminate.

Then if n_k has value $\theta(n_k)$ upon termination, $T_k = -\theta(n_k)$. Again, if G is not out-point connected, this algorithm yields maximal paths from n_1 to all nodes n_i of G whenever they exist. $\{T_i\}$ for an arbitrary graph G can be determined in a fashion parallel to that given above.

3. Synchronous Computation Graphs

Suppose that the node initiation times of a computation graph G are to be controlled by a clock signal. Then node initiation times are constrained to be integer multiples of the clock period. But if π is not an integer the maximal rate periodic schedule of Section 2 is not suitable. Indeed, no maximal rate periodic execution of G, with integer initiation times, is possible in the case that π is not an integer. This remark follows from the proof of Theorem 4 where it is shown that if p is the period of a proper execution of G, then $p \geq \pi$, so that if G must have integer initiation times, we must have $p \geq \lceil \pi \rceil$ (the greatest integer containing π). Then, clearly, if π is not an integer, $1/p < 1/\pi$, the maximal computation rate.

In view of this contingency, we consider computation graphs for which the node execution times $\{\tau_i\}$ are all positive integers, and in which a node n_j is permitted to initiate only at integer times. We call such a computation graph a synchronous computation graph.

The principal result of this section is the determination of an execution (the free running execution), for a class of synchronous computation graphs, under which G computes at the maximal rate $1/\pi$.

Theorem 5 considers the most general form of a computation graph, as defined in [3].

Theorem 5

Let G be a synchronous computation graph. Let $\mathcal{E}$ be an execution under which, for all j, node n_j initiates at time t if and only if for all branches b_{ij} into n_j, the number of data words on b_{ij} is greater than or equal to T_{ij}. Let $\mathcal{E}'$ be any other execution.

Then $x(n_j,t) \geq x'(n_j,t)$.

PROOF: The proof is by induction on t. For t=0 the result is immediate. Assume, for $t \leq n$, that for all nodes n_j of G, $x(n_j,t) \geq x'(n_j,t)$. Let $t = n+1$ and consider node n_j.

(1) Suppose n_j initiates at t=n under $\mathcal{E}$. Then

$x(n_j,n+1) = 1 + x(n_j,n) \geq 1 + x'(n_j,n) \geq x'(n_j,n+1)$

and the result holds.

(2) Suppose n_j does not initiate at t=n under both $\mathcal{E}$ and $\mathcal{E}'$. Then $x(n_j,n+1) = x(n_j,n) \geq x'(n_j,n) = x'(n_j,n+1)$ and again the theorem holds.

(3) Suppose n_j does not initiate at t=n under $\mathcal{E}$ but n_j does initiate at t=n under $\mathcal{E}'$. Then there are two possibilities:

(a) $x(n_j,n) > x'(n_j,n)$ in which case

$x(n_j,n+1) = x(n_j,n) \geq x'(n_j,n) + 1 = x'(n_j,n+1)$.

(b) $x(n_j,n) = x'(n_j,n)$. Since n_j does not initiate at t=n under $\mathcal{E}$ but does initiate at t=n under $\mathcal{E}'$, there is a branch $b_{ij} = (n_i,n_j)$ such that $b_{ij}(n+1) < T_{ij}$ and $b'_{ij}(n+1) \geq T_{ij}$ where the prime indicates queue length under $\mathcal{E}'$. Then, in particular, $b_{ij}(n+1) < b'_{ij}(n+1)$, i.e.

$$A_{ij} + U_{ij}\ x(\bar{n}_i,n+1) - W_{ij}\ x(n_j,n) < A_{ij} + U_{ij}\ x'(\bar{n}_i,n+1) - W_{ij}\ x'(n_j,n)$$

where we write $x(\bar{n}_i,n+1)$ as the number of terminations of node n_i in time n+1 under $\mathcal{E}$, and primes correspond to $\mathcal{E}'$. Hence $x(\bar{n}_i,n+1) - x'(\bar{n}_i,n+1)$ which contradicts the induction hypothesis $x(n_i,t) \geq x'(n_i,t)$ for $t=0,1,\ldots,n$.

We call the execution $\mathcal{E}$ in the statement of Theorem 5 the <u>free running execution</u>. Theorem 5 then establishes that a synchronous computation graph executes at the fastest possible rate under the free running execution.

Note that Theorem 5 has been proved under the assumption that a node n_j can initiate at any time that there is sufficient data on all of its input branches; i.e. regardless of whether or not n_j has terminated its previous initiation. If we wish a model in which no node can initiate unless it has terminated its previous execution, we can obtain it by simply adding a self-loop to n_j for all n_j of G.

3.1 Synchronous Graphs with $\tau=1$, U=W=T=1

We consider synchronous computation graphs with $\tau_i = 1$ for all nodes n_i, and for all branches $b_{ij} = (n_i,n_j)$ we have $U_{ij} = W_{ij} = T_{ij} = 1$. Later we shall see how the results obtained with $\tau_i=1$ may be applied to a graph in which n_i has associated with it an arbitrary positive integer execution time.

Let n be a node of G and for $k \geq 1$, $S_k(n)$ a sequence of (not necessarily distinct) nodes, $\{n_0, n_1, \ldots, n_{p+1}\}$, such that

(1) $n_{p+1} \rightarrow n_p \rightarrow \ldots \rightarrow n_1 \rightarrow n_0$ with $n_0 = n$.

(2) $$\sum_{j=1}^{p+1} A_{j,j-1} \geq k.$$

(3) $$\sum_{j=1}^{p} A_{j,j-1} < k.$$

We call such a sequence $S_k(n)$ a <u>k sequence</u>.

Put
$$A_0 = A_{10}$$
$$A_1 = A_{21}$$
$$\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$
$$A_{p-1} = A_{p,p-1}$$
$$A_p = k - \sum_{i=0}^{p-1} A_i.$$

Define $A'_0 = A_0$ and for $1 \leq j \leq p$ let

$$A'_j = A_j + A'_{j-1} + K_j$$

where $K_j = -1$ if $A'_{j-1} \geq 1$

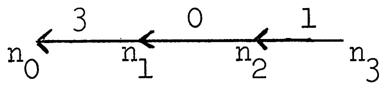
$= 0$ if $A'_{j-1} = 0$.

Define $d_k(S_k(n)) = p + A'_p - 1$.

Intuitively, the significance of $d_k(S_k(n))$ is as follows: Let w_k be the k-th data word "backed up" from node n along the node sequence

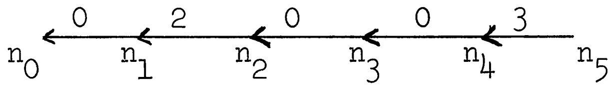
$S_k(n)$. Then $d_k(S_k(n))$ is a measure of "how far away" w_k is from the node n.

Example



$S_4(n_0)$

$d_4(S_4(n_0)) = 3$



$S_4(n_0)$

$d_4(S_4(n_0)) = 5$

Define $T_k(n) = \max\limits_{S_k(n)} \{d_k(S_k(n))\}$.

We shall be interested in tracing the progress of the k-th data word through time along a k sequence $S_k(n)$. To that end we make the following definitions:

Let $A_0, A_1, \ldots, A_p$ be a sequence of nonnegative integers.

Let $\epsilon_0, \epsilon_1, \ldots, \epsilon_p$ and $\delta_0, \delta_1, \ldots, \delta_p$ be sequences such that

(1) $\epsilon_i = 0$ or 1 $\quad i=0,1,\ldots,p-1, \quad \epsilon_p = 0$

(2) $\delta_i = 0$ or 1 $\quad i=0,1,\ldots,p$

(3) $\epsilon_i = 1 \Longleftrightarrow \delta_{i+1} = -1 \quad i=0,1,\ldots,p-1.$

A sequence $A_0(1), A_1(1), \ldots, A_p(1)$ is said to be <u>1-admissible</u> if

$$A_i(1) = A_i + \epsilon_i + \delta_i \qquad i=0,1,\ldots,p.$$

In general, if $A_0(t-1), A_1(t-1), \ldots, A_p(t-1)$ is $(t-1)$-admissible, then the sequence $A_0(t), A_1(t), \ldots, A_p(t)$ is said to be <u>t-admissible</u> if $A_i(t) = A_i(t-1) + \epsilon_i + \delta_i$ where ϵ_i and δ_i satisfy the conditions (1) - (3) above.

Let $S_k(n)$ be the k sequence $n_{p+1} \rightarrow n_p \rightarrow \dots \rightarrow n_0$, $n_0=n$, and let

$$A_i = A_{i+1,i}, \qquad i=0,1,\dots,p-1$$

$$A_p = k - \sum_{i=0}^{p-1} A_i.$$

Then the k-th data word w_k "backed up" from node n lies on branch $b_{p+1,p}$. Let $A_0(t)$, $A_1(t)$, ..., $A_p(t)$ be a t-admissible sequence derived from A_0, A_1, ..., A_p, and let r be the maximum subscript such that $A_r(t) \neq 0$. Then at time t, w_k lies on branch $b_{r+1,r}$ and the sequence $A_0(t)$, $A_1(t)$, ..., $A_r(t)$ yields the branch distribution of data words "between" node n_0 and w_k.

Define $A_0'(t) = A_0(t)$ and for $1 \leq j \leq p$, let

$$A_j'(t) = A_j(t) + A_{j-1}'(t) + K_j(t)$$

where

$$K_j(t) = -1 \quad \text{if} \quad A_{j-1}'(t) \geq 1$$
$$= 0 \quad \text{if} \quad A_{j-1}'(t) = 0.$$

Let r be the maximum subscript such that $A_r(t) \neq 0$. Define

$$d(S_k(n),t) = r + A_r'(t) - 1$$

$d(S_k(n),t)$ is the "distance" of w_k from node n at time t.

Define $T_k(n,t) = \max_{S_k(n)} \{d(S_k(n),t)\}$.

Lemma 6

$$A_r'(1) = A_r' + v, \; v \in \{-1,0,1\}, \qquad r=0,1,\dots,p$$

$$A_r'(1) = A_r' + 1 \Longrightarrow \epsilon_r = 1$$

$$A_r'(1) = A_r' - 1 \Longrightarrow \epsilon_r = 0$$

PROOF: The proof is by induction on r. Take r=0. Then

$$A_0'(1) = A_0(1) = A_0 + \epsilon_0 + \delta_0 = A_0' + \epsilon_0 + \delta_0$$

and $\epsilon_0 + \delta_0 \in \{-1,0,1\}$. Clearly,

$$A_0'(1) = A_0' + 1 \implies \epsilon_0 = 1$$

$$A_0'(1) = A_0' - 1 \implies \epsilon_0 = 0.$$

Assume the result to hold for r=m-1 and consider

$$\begin{aligned} A_m'(1) &= A_m(1) + A_{m-1}'(1) + K_m(1) \\ &= A_m + \epsilon_m + \delta_m + A_{m-1}'(1) + K_m(1) \\ &= A_m' + \epsilon_m + \delta_m + A_{m-1}'(1) - A_{m-1}' + K_m(1) - K_m. \end{aligned}$$

(i) Suppose $A_{m-1}'(1) = A_{m-1}'$

Then $K_m(1) = K_m$ and

$$A_m'(1) = A_m' + \epsilon_m + \delta_m$$

i.e. $A_m'(1) = A_m' + v.$

Clearly $A_m'(1) = A_m' + 1 \implies \epsilon_m = 1$

$$A_m'(1) = A_m' - 1 \implies \epsilon_m = 0.$$

(ii) Suppose $A_{m-1}'(1) = A_{m-1}' + 1$. Then $\epsilon_{m-1} = 1$ by the induction hypothesis so that $\delta_m = -1$. Also, $A_{m-1}'(1) > 1$ so $K_m(1) = -1$.

Therefore $A_m'(1) = A_m' + \epsilon_m - 1 + 1 - 1 - K_m$

$$= A_m' + \epsilon_m - 1 - K_m$$

$$= A_m' + v.$$

Finally $\epsilon_m - 1 - K_m = 1 \implies \epsilon_m = 1$

$$\epsilon_m - 1 - K_m = -1 \implies \epsilon_m = 0.$$

(iii) Suppose $A_{m-1}'(1) = A_{m-1}' - 1$. Then by the induction hypothesis $\epsilon_{m-1} = 0$ so $\delta_m = 0$. Also $A_{m-1}' \geq 1$ so that $K_m = -1$. Therefore

$$\begin{aligned} A_m'(1) &= A_m' + \epsilon_m - 1 + K_m(1) + 1 \\ &= A_m' + \epsilon_m + K_m(1) \end{aligned}$$

and $\epsilon_m + K_m(1) \in \{-1,0,1\}$.

Clearly, $\epsilon_m + K_m(1) = 1 \Longrightarrow \epsilon_m = 1$

$\epsilon_m + K_m(1) = -1 \Longrightarrow \epsilon_m = 0$.

Let $A_0(1), A_1(1), \ldots, A_p(1)$ be a 1-admissible sequence derived from $A_0, A_1, \ldots, A_p$. Let r be the maximum subscript such that $A'_{r-1} = 0$. If no such r exists, take r=0. We say that $A_0(1), A_1(1), \ldots, A_p(1)$ is freely derived from $A_0, A_1, \ldots, A_p$ if

(i) $\delta_r = -1$

(ii) $A'_s = 1 \Longrightarrow \epsilon_s = 1 \qquad r \leq s \leq p-1$.

Lemma 7

Let $A_0(1), A_1(1), \ldots, A_p(1)$ be freely derived from $A_0, A_1, \ldots, A_p$. Then $A'_p(1) = A'_p - 1$.

PROOF: Suppose $r \neq 0$. By Lemma 6 we have $A'_{r-1}(1) \in \{0,1\}$, i.e.

$$A'_{r-1}(1) + K_r(1) = 0$$

Then
$$\begin{aligned}
A'_p(1) &= A_p(1) + A'_{p-1}(1) + K_p(1) \\
&= A_p(1) + [A_{p-1}(1) + A'_{p-2}(1) + K_{p-1}(1)] + K_p(1) \\
&\quad \cdots\cdots\cdots\cdots\cdots \\
&= A_p(1) + \ldots + A_r(1) + A'_{r-1}(1) + K_r(1) + K_{r+1}(1) \\
&\quad + \ldots + K_p(1) \\
&= A_p(1) + \ldots + A_r(1) + K_{r+1}(1) + \ldots + K_p(1) \\
&= \epsilon_p + A_p + \ldots + A_r + \delta_r + K_{r+1}(1) + \ldots + K_p(1) \\
&= A_p + \ldots + A_r + K_p + \ldots + K_{r+1} + (K_p(1) - K_p) \\
&\quad + \ldots + (K_{r+1}(1) - K_{r+1}) + \epsilon_p + \delta_r.
\end{aligned}$$

But since $A'_{r-1} = 0$ we have

$$A'_p = A_p + \ldots + A_r + K_p + \ldots + K_{r+1}.$$

Hence $\quad A'_p(1) = A'_p + (K_p(1) - K_p) + \ldots + (K_{r+1}(1) - K_{r+1}) + \epsilon_p + \delta_r.$

If r=0 we obtain the same expression for $A'_p(1)$:

$$\begin{aligned} A'_p(1) &= A_p(1) + \ldots + A_0(1) + K_p(1) + \ldots + K_1(1) \\ &= A_p + \ldots + A_0 + \epsilon_p + \delta_0 + K_p(1) + \ldots + K_1(1) \\ &= A_p + \ldots + A_0 + K_p + \ldots + K_1 + \epsilon_p + \delta_0 + (K_p(1) - K_p) \\ &\quad + \ldots + (K_1(1) - K_1) \\ &= A'_p + (K_p(1) - K_p) + \ldots + (K_1(1) - K_1) + \epsilon_p + \delta_0 \end{aligned}$$

Now if $A'_s > 1$, $p-1 \geq s \geq r$ then by Lemma 6 $A'_s(1) \geq 1$ so that $K_{s+1}(1) = K_{s+1} = -1$. Suppose $A'_s = 1$. Then $\epsilon_s = 1$ and by Lemma 6 $A'_s(1) \geq A'_s = 1$, i.e. $K_{s+1}(1) = K_{s+1} = -1$.

Since r is the maximum subscript such that $A'_{r-1} = 0$, these are the only possibilities for A'_s. Hence

$$A'_p(1) = A'_p + \epsilon_p + \delta_r.$$

But $\epsilon_p = 0$, $\delta_r = -1$, i.e. $A'_p(1) = A'_p - 1$.

<u>Lemma 8</u>

Let G be a synchronous computation graph under the free running execution. Let $S_k(n)$, $n_{p+1} \to n_p \to \ldots \to n_0$, $n_0=n$, be a k-sequence for which $d_k(S_k(n)) = T_k(n)$. Then for $0 \leq t \leq T_k(n)$

(i) $\quad d(S_k(n),t) = T_k(n) - t$

(ii) $\quad d(S_k(n),t) = T_k(n,t)$.

PROOF: We prove the result for t=1. Then by (ii), the conditions for the lemma to hold are valid at t=1 and we have a basis for iterating the argument which established the result for $t=1$.

If $T_k(n) = 0$, there is nothing to prove. Hence assume $T_k(n) \geq 1$.

(i) Consider the sequence $A_0, A_1, \ldots, A_p$ where $A_i = A_{i+1,i}$, $i=0,1,\ldots,p-1$, and

$$A_p = k - \sum_{i=0}^{p-1} A_i.$$

Define, for $i=0,1,\ldots,p$, $\delta_i = -1$ if and only if node n_i initiates at $t=0$. Otherwise $\delta_i = 0$. For $i=0,1,\ldots,p-1$, put $\epsilon_i = 1$ if and only if $\delta_{i+1} = -1$. Otherwise $\epsilon_i = 0$. Finally, let $\epsilon_p = 0$. We prove that the sequence $A_0(1)$, $A_1(1)$, ..., $A_p(1)$ defined by $A_i(1) = A_i + \epsilon_i + \delta_i$, $i=0,1,\ldots,p$ is a 1-admissible sequence freely derived from $A_0, A_1, \ldots, A_p$.

Let r be the maximum subscript such that $A'_{r-1} = 0$. If no such r exists take $r=0$. We show that n_r initiates at $t=0$. Clearly, $A_r > 0$. Suppose there exists a branch $b_{r'+1,r}$ into n_r such that $A_{r'+1,r} = 0$. Let $S'_k(n)$ be a k-sequence $n_{q'+1} \rightarrow n_{q'} \rightarrow \ldots \rightarrow n_{r'+1} \rightarrow n_{r'} \rightarrow n_0$ where $r'=r$. Then since $A'_{r-1} = 0$, or in the case $r=0$, we must have

$$\begin{aligned} d_k(S'_k(n)) &= q' + A'_{q'} - 1 \\ &= q' + A_{q'} + \ldots + A_{r'} + K_{q'} + \ldots + K_{r'+1} - 1. \end{aligned}$$

But since
$$\begin{aligned} A'_{r'} &= A_{r'} + A'_{r-1} + K_{r'} \\ &= 0, \end{aligned}$$

then
$$K_{r'+1} = 0.$$

Hence
$$\begin{aligned} d_k(S'_k(n)) &= q' + A_{q'} + \ldots + A_{r'} + K_{q'} + \ldots + K_{r'+2} - 1 \\ &= q' + A_p + \ldots + A_r + K_{q'} + \ldots + K_{r'+2} - 1 \\ &\geq q' + A_p + \ldots + A_r - [q' + 1 - (r'+2)] - 1 \\ &= A_p + \ldots + A_r + r \\ &= d_k(S_k(n)) + 1 \end{aligned}$$

a contradiction. Hence $A_{r'+1,r} > 0$ for all branches $b_{r'+1,r}$ into n_r and n_r initiates at t=0. Now let $A'_s = 1$ for $p-1 \geq s \geq r$. Then by a similar argument to that just given we must have that n_{s+1} initiates at t=0.

It follows that $A_0(1), A_1(1), \ldots, A_p(1)$ is a 1-admissible sequence freely derived from $A_0, A_1, \ldots, A_p$. By Lemma 7, $A'_p(1) = A'_p - 1$. Then if $A'_p > 1$,

$$\begin{aligned} d(S_k(n),1) &= d_k(S_k(n)) - 1 \\ &= T_k(n) - 1. \end{aligned}$$

Suppose $A'_p = 1$. Then since $T_k(n) \geq 1$ we must have $p \geq 1$. Then $A'_{p-1} \in \{0,1\}$. If $A'_{p-1} = 0$ then since $A_0(1), A_1(1), \ldots, A_p(1)$, is freely derived from $A_0, A_1, \ldots, A_p$, we must have $\epsilon_{p-1} = 1$. Since $A'_{p-1} = 0$ implies $A_{p-1} = 0$ then $\delta_{p-1} = 0$ and $A_{p-1}(1) = 1$. By Lemma 6, $A'_{p-1}(1) = 1$ and

$$\begin{aligned} d(S_k(n),1) &= p-1 + A'_{p-1}(1) - 1 \\ &= (p + A'_p - 1) - 1 \\ &= d_k(S_k(n)) - 1 \\ &= T_k(n) - 1. \end{aligned}$$

Finally, we consider the case $A'_{p-1} = 1$. Let r be the maximum subscript such that $A'_r = 0$. If no such r exists, take r=0. Then by the same argument as in the proof of Lemma 7, we obtain

$$A'_{p-1}(1) = A'_{p-1} + \epsilon_{p-1} + \delta_r.$$

Since $A_0(1), A_1(1), \ldots, A_p(1)$ is freely derived from $A_0, A_1, \ldots, A_p$, we have

$$\epsilon_{p-1} = 1, \qquad \delta_r = -1 \qquad \text{and}$$

$$A'_{p-1}(1) = A'_{p-1} = A'_p$$

so again $$d(S_k(n),1) = T_k(n) - 1.$$

(ii) Let $S_k'(n)$ be a k-sequence and suppose

$$d(S_k(n),1) < d(S_k'(n),1). \qquad (1)$$

Then $\quad d_k(S_k(n)) - 1 < d(S_k'(n),1) \leq d_k(S_k'(n)),$

i.e. $\quad d_k(S_k(n)) \leq d_k(S_k'(n)).$

But $\quad d_k(S_k(n)) = T_k(n)$

so that $\quad d_k(S_k'(n)) = T_k(n).$

By (i), $\quad d(S_k'(n),1) = d_k(S_k'(n)) - 1.$

By (1), $\quad d_k(S_k(n)) < d_k(S_k'(n))$

a contradiction.

Theorem 9

Let G be a synchronous computation graph under the free running execution. Then node n initiates for the k-th time at $t = T_k(n)$.

PROOF: Let $S_k(n)$ be a k-sequence for which $d_k(S_k(n)) = T_k(n)$. Then by Lemma 8, $d(S_k(n),T_k(n)) = 0 = T_k(n,T_k(n))$. Thus at $t = T_k(n)$ every input branch to node n contains a data word so that n initiates at time $T_k(n)$. Finally, if w_k is the k-th data word "backed up" from node n along the sequence $S_k(n)$ at t=0, then at $t=T_k(n)$, w_k is the first data word "backed up" from node n along $S_k(n)$, i.e. $x(n,T_k(n)) = k-1$ and n initiates for the k-th time at $t = T_k(n)$.

Let $\vec{b}(t)$ be the vector with components $b_{ij}(t)$ for all branches (n_i,n_j) of G.

Lemma 10

Let G be a strongly connected synchronous computation graph under the free running execution. Then there exists a nonnegative integer t' and a positive integer λ such that for all $t \geq t'$

$$\vec{b}(t+\lambda) = \vec{b}(t).$$

PROOF: Since G has $\tau_i = 1$ for all nodes n_i of G, $\vec{b}(t+1)$ is uniquely determined by $\vec{b}(t)$, i.e.

$b_{ij}(t+1) = b_{ij}(t)$ if there is (not) a branch b_{ki} with $b_{ki}(t) = 0$ and there is (not) a branch b_{rj} with $b_{rj}(t) = 0$

$b_{ij}(t+1) = b_{ij}(t) + 1$ if for all branches b_{ki}, $b_{ki}(t) \geq 1$ and there is a branch b_{rj} such that $b_{rj}(t) = 0$.

$b_{ij}(t+1) = b_{ij}(t) - 1$ if for all branches b_{rj}, $b_{rj}(t) \geq 1$ and there is a branch b_{ki} such that $b_{ki}(t) = 0$.

Consider the sequence $B = \vec{b}(0), \vec{b}(1), \ldots,$. Since G is strongly connected, the queue lengths of G are uniformly bounded above (See [3.] for a proof of this statement.) Hence there exist finitely many vectors $\vec{b}(t)$ so that some $\vec{b}(t)$ of B must repeat in the sequence. Therefore there exists a positive integer λ and a nonnegative integer t' such that $\vec{b}(t'+\lambda) = \vec{b}(t')$. But, by the above remarks, if $\vec{b}(t'+\lambda) = \vec{b}(t')$, then $\vec{b}(t'+\lambda+1) = \vec{b}(t'+1)$, etc. Hence the Lemma.

Lemma 11

Let G be strongly connected. Then for any pair of nodes n_i and n_j of G,

$$\lim_{t \to \infty} \frac{x(n_i,t)}{x(n_j,t)} = 1.$$

PROOF: Let $n_i \to n_k$. Then

$$b_{ik}(t) = A_{i,k} + x[n_i,(t-\tau_i)^+] - x(n_k,t).$$

Since $U_{ij}= W_{ij}= T_{ij}= 1$ for all branches $b_{ij}= (n_i,n_k)$ of G, and since G is strongly connected, then $b_{ik}(t)$ is uniformaly bounded above and G is nonterminating. (For the proofs of these statements, see [3].) Hence

$$\lim_{t \to \infty} \frac{x[n_i,(t-\tau_i)^+]}{x(n_k,t)} = \lim_{t \to \infty} \frac{x(n_i,t)}{x(n_k,t)} = 1.$$

Since G is strongly connected, there is a path from n_i to n_j so we may iterate the above argument yielding the statement of the lemma.

Lemma 12

Let G be a strongly connected synchronous computation graph, under the free running execution. Then for all $t \geq t'$, $\vec{b}(t+\lambda) = \vec{b}(t)$ if and only if for all $t \geq t'$, $x(n_j,t+\lambda) - x(n_j,t) = \alpha$, a constant independent of n_j and t.

PROOF: $\Longrightarrow$

Let n_j initiate at time t. Since $b_{ij}(t) = b_{ij}(t+\lambda)$ for all branches b_{ij}, n_j must initiate at time $t+\lambda$. Suppose n_j initiates at times $t_1,t_2, \ldots,t_\alpha$ where $t' \leq t_1 < t_2 < \ldots < t_\alpha < t_1+\lambda$. Then by the above remark, n_j initiates at times $t_1 + \lambda$, $t_2 + \lambda$, $t_\alpha + \lambda$, $t_1 + 2\lambda$, ..., i.e. $x(n_j,t+\lambda) - x(n_j,t) = \alpha_j$, independent of t.

Now let n_k be another node of G. Then we obtain, similarly, $x(n_k,t+\lambda) - x(n_k,t) = \alpha_k$ independent of t. By by Lemma 11,

$$\lim_{r \to \infty} \frac{x(n_k,t+r\lambda)}{x(n_j,t+r\lambda)} = 1$$

Therefore $\qquad \alpha_k/\alpha_j = 1.$

$\Longleftarrow$

$$b_{ij}(t+\lambda) = A_{i,j} + x(n_i,(t+\lambda-1)^+) - x(n_j,t+\lambda)$$
$$= A_{i,j} + x(n_i,t+\lambda) - x(n_j,t+\lambda).$$

But $\qquad x(n_i,t+\lambda) - x(n_i,t) = x(n_j,t+\lambda) - x(n_j,t).$

Hence $\qquad b_{ij}(t+\lambda) = A_{i,j} + x(n_i,t) - x(n_j,t)$
$$= b_{ij}(t).$$

It follows from Lemma 12 that $\vec{b}(t)$ has period λ if and only if for all nodes n_i of G, n_i initiates at times

$$t_i^0,\ t_i^1,\ \ldots,\ t_i^{\alpha-1},$$
$$t_i^0 + \lambda,\ t_i^1 + \lambda,\ \ldots,\ t_i^{\alpha-1} + \lambda,$$
$$t_i^0 + 2\lambda,\ t_i^1 + 2\lambda,\ \ldots,\ t_i^{\alpha-1} + 2\lambda,$$
$$\cdots\cdots\cdots\cdots\cdots\cdots$$

where $\qquad t' \le t_i^0 < t_i^1 < \ldots < t_i^{\alpha-1} < t_i^0 + \lambda.$

Lemma 13

Let G be a strongly connected synchronous computation graph under the free running execution. Then for any node n

$$\lim_{k\to\infty} \frac{T_k(n)}{k} = \frac{\lambda}{\alpha}$$

PROOF: By Theorem 9, for any node n of G

$$x(n,T_k(n)) = k-1.$$

Therefore

$$\frac{T_k(n)}{x(n,T_k(n))} = \frac{T_k(n)}{k-1}$$

and

$$\lim_{k \to \infty} \frac{T_k(n)}{x(n,T_k(n))} = \lim_{k \to \infty} \frac{T_k(n)}{k} \qquad (1)$$

By Lemma 12, for all $t \geq t'$

$$x(n,t+r\lambda) - x(n,t) = r\alpha$$

i.e.

$$\lim_{r \to \infty} \frac{x(n,t+r\lambda)}{t+r\lambda} = \lim_{r \to \infty} \frac{r\alpha}{t+r\lambda}$$

$$= \alpha/\lambda$$

Hence by (1)

$$\lim_{k \to \infty} \frac{T_k(n)}{k} = \lambda/\alpha.$$

<u>Theorem 14</u>

Let G be a strongly connected synchronous computation graph. Then

$$\lambda/\alpha = \max(1,\pi).$$

PROOF: Let L be a loop, length ℓ, of G for which $\pi_L = \pi$ and let A be the total number of data words on the branches of L. Then, since a data word requires at least ℓ units of time to pass once around L, we must have

$$x(n,t+\ell) - x(n,t) \leq \min(\ell,A),$$

for any node n of L. By Lemma 12,

$$x(n,t+\ell\lambda) - x(n,t) = \ell\alpha$$

for all $t \geq t'$. By the above remark

$$\ell\alpha \leq \lambda \min(\ell, A)$$

i.e.

$$\lambda/\alpha \geq \max(1,\pi) \tag{1}$$

(i) Suppose $\pi > 1$. Let n be a node of G and let $S_k(n)$, $n_{p+1} \to n_p \to \ldots \to n_{r+1} \to n_r \to \ldots \to n_0$, $n_0=n$, be a k-sequence such that $d_k(S_k(n)) = T_k(n)$. Let r be the maximum node subscript such that if $n_{r+1} \to n_r \to \ldots \to n$ is an m-sequence $S_m(n)$, then

$$d_m(S_m(n)) = r$$

r exists by the definition of $d_1(S_1(n))$. We shall prove that k-m is uniformly bounded above. Consider the (k-m+1)-sequence, $S_{k-m+1}(n_r)$, $n_{p+1} \to n_p \to \ldots \to n_{r+1} \to n_r$. By the manner in which r was chosen, we must have

$$d_{k-m+1}(S_{k-m+1}(n_r)) = k - m \tag{2}$$

and also

$$T_k(n) = d_k(S_k(n)) = d_m(S_m(n)) + d_{k-m+1}(S_{k-m+1}(n_r)) \tag{3}$$

We prove that

$$d_{k-m+1}(S_{k-m+1}(n_r)) = T_{k-m+1}(n_r) \tag{4}$$

For let $S'_{k-m+1}(n_r)$, $n_q \to n_{q-1} \to \ldots \to n_r$ be such that

$$d_{k-m+1}(S'_{k-m+1}(n_r)) > d_{k-m+1}(S_{k-m+1}(n_r)).$$

Then S''_k, $n_q \to n_{q-1} \to \ldots n_r \to n_{r-1} \to \ldots \to n$ is such that

$$\begin{aligned} d_k(S''_k(n)) &= d_m(S_m(n)) + d_{k-m+1}(S'_{k-m+1}(n_r)) \\ &> d_m(S_m(n)) + d_{k-m+1}(S_{k-m+1}(n_r)) \\ &= d_k(S_k(n)) \\ &= T_k(n) \qquad \text{a contradiction.} \end{aligned}$$

Now suppose that k-m is not uniformly bounded above. Put s = k-m. Then there exists some subsequence $\{s_i\}$ of $\{s\}$ such that $s_i \to \infty$. But then

$$\lim_{s_i \to \infty} \frac{T_{s_i+1}(n_r)}{s_i+1} = \lim_{s_i \to \infty} \frac{d_{s_i+1}(S_{s_i+1}(n_r))}{s_i+1} \qquad \text{by (4)}$$

$$= \lim_{s_i \to \infty} \frac{s_i}{s_i+1} \qquad \text{by (2)}$$

$$= 1.$$

But

$$\lim_{s \to \infty} \frac{T_s(n_r)}{s} = \lambda/\alpha \geq \pi > 1 \qquad \text{by Lemma 13 and (1)}$$

a contradiction. Hence there exists M such that $k-m \leq M$ for all k.

Now let P be a path from n to n_{p+1} and let ℓ be the total length of the circuit defined by the sequence $n_{p+1} \to n_p \to \dots \to n$ and then back to n_{p+1} via P. Let $L_1, L_2, \dots, L_t$ be all of the loops making up the above circuit, and for $i=1,2,\dots,t$ let A_i be the number of data words on L_i and ℓ_i the length of L_i. Then by the choice of r,

$$\begin{aligned} d_m(S_m(n)) &= r < \ell \\ &= \ell_1 + \dots + \ell_t \\ &= A_1\, \ell_1/A_1 + \dots + A_t\, \ell_t/A_t \\ &\leq \pi \sum_{i=1}^{t} A_i \end{aligned}$$

i.e.
$$\frac{d_m(S_m(n))}{\sum_{i=1}^{t} A_i} < \pi$$

i.e. by (3)

$$\frac{T_k(n) - d_{k-m+1}(S_{k-m+1}(n_r))}{k} \cdot \frac{k}{\sum_{i=1}^{t} A_i} < \pi$$

Now since $k - m \leq M$,

$$\lim_{k \to \infty} \frac{d_{k-m+1}(S_{k-m+1}(n_r))}{k} = 0$$

Also, since k-$m \leq M$ and since P is a path we have

$$\lim_{k \to \infty} \frac{k}{\sum_{i=1}^{t} A_i} = 1.$$

Hence

$$\lim_{k \to \infty} \frac{T_k(n)}{k} \leq \pi.$$

This coupled with (1) and Lemma 13 yields the desired result.

(ii) Suppose $\pi \leq 1$. If k-m is not uniformly bounded above, we have, as before

$$\lim_{s_i \to \infty} \frac{T_{s_i}(n_r)}{s_i} = 1$$

which is what we wanted to prove.

Otherwise, we obtain, as before

$$\lim_{k \to \infty} \frac{T_k(n)}{k} \leq \pi. \tag{5}$$

If $\pi < 1$, this contradicts (1) so that k-m is not uniformly bounded above. If $\pi = 1$ then (1) and (5) together with Lemma 13 yield the desired result.

Corollary 15

If $\pi \leq 1$, then $\lambda = \alpha = 1$.

PROOF: By Lemma 12, for all $t \geq t'$ and for any node n of G

$$x(n,t+\lambda) - x(n,t) = \alpha$$

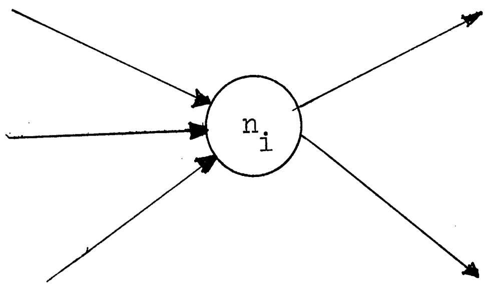
$$= \lambda \quad \text{by Theorem 14.}$$

But $$x(n,t+\lambda) - x(n,t) = \lambda \quad \text{if and only if}$$

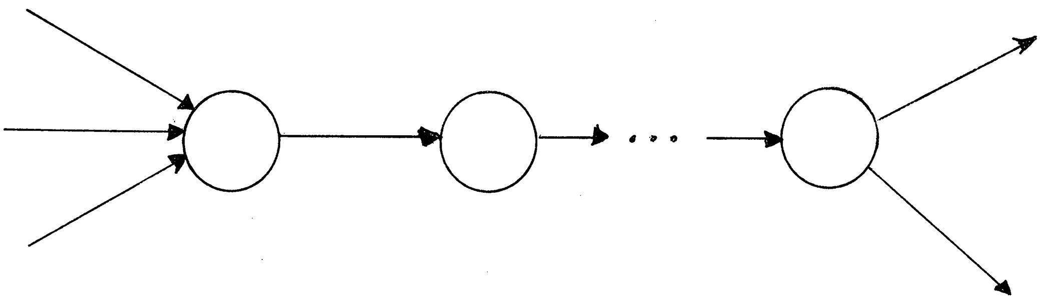
node n initiates at times t, t+1, ..., t+λ-1, i.e. node n initiates at times t', t'+1, t'+2, ...,.

3.2 Computation Graphs with $\tau_i \geq 2$

These may be reduced to the case τ_i=1 for all n_i as follows. Suppose n_i has $\tau_i \geq 2$.



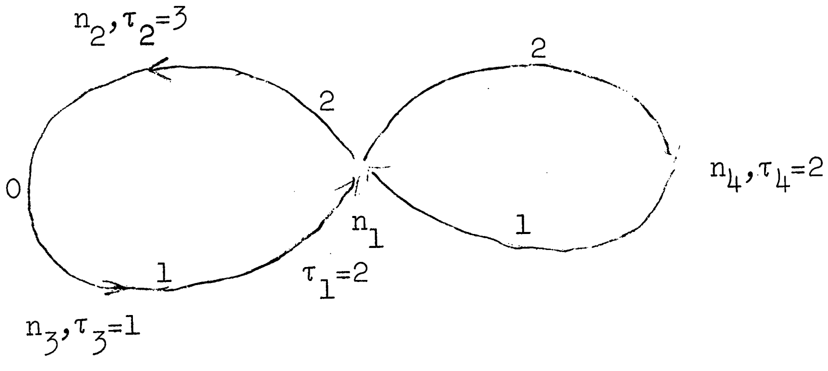
Then replace n_i by the following sequence of τ_i nodes each with the execution time 1.



It should be clear that all of the above results concerning free running executions of synchronous computation graphs can now be translated into similar results for the case $\tau_i \geq 1$, except that n_i can initiate at unit time intervals, so that if $\tau_i \geq 2$, n_i can initiate before its previous execution has terminated. If this situation is undesirable, we may place a self-loop on n_i.

An Example

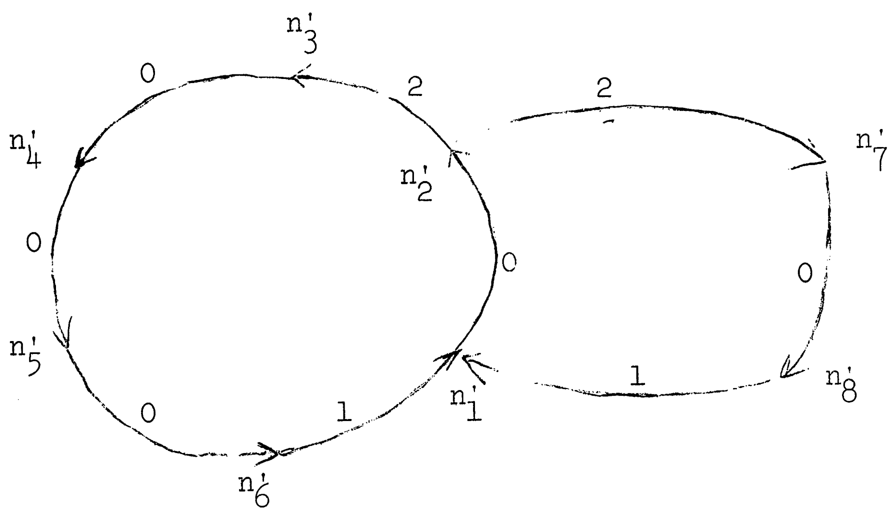
G:



$$\pi = \frac{\tau_1 + \tau_2 + \tau_3}{3} = 2.$$

We can transform G into an equivalent graph G' with unit node execution times.

G':



By simulating G' we obtain the following sequence of branch vectors:

t	b_{12}	b_{23}	b_{34}	b_{45}	b_{56}	b_{61}	b_{27}	b_{78}	b_{81}
0	0	2	0	0	0	1	2	0	1
1	1	1	1	0	0	0	1	1	0
2	0	1	1	1	0	0	1	1	1
3	0	0	1	1	1	0	0	1	2
4	0	0	0	1	1	1	0	0	3
5	1	0	0	0	1	1	0	0	2
6	1	1	0	0	0	1	1	0	1
7	1	1	1	0	0	0	1	1	0

Then $\vec{b}(7) = \vec{b}(1)$. Hence $\lambda = 6$, $\alpha = 3$. From the above table, we can determine the initiation times of n_1, n_2, n_3, n_4 of G, which are the initiation times of n_1', n_3', n_6', n_7', respectively. We obtain

n_1: 0, 4, 5, 6, 10, 11, 12, 16, 17, 18, ...,

n_2: 0, 1, 2, 6, 7, 8, 12, 13, 14, 18, ...,

n_3: 3, 4, 5, 9, 10, 11, 15, 16, 17, 21, ...,

n_4: 0, 1, 2, 6, 7, 8, 12, 13, 14, 18, ...,.

4. A Maximal Rate Periodic Schedule for Synchronous Computation Graphs

In Section 3 we noted that if π is not an integer, then no maximal rate periodic execution is possible for a synchronous computation graph. The remark following Lemma 12 suggests the following definition:

Let λ and α be positive integers. A synchronous computation graph G is said to have a <u>(λ,α) periodic execution</u> if for each node n_i of G, there exists integers $t_i^0 < t_i^1 < \dots < t_i^{\alpha-1} < t_i^0 + \lambda$ such that n_i initiates only at times

$$\begin{array}{l} t_i^0,\ t_i^1,\ \dots,\ t_i^{\alpha-1}, \\ t_i^0 + \lambda,\ t_i^1 + \lambda,\ \dots,\ t_i^{\alpha-1} + \lambda, \\ t_i^0 + 2\lambda,\ t_i^1 + 2\lambda,\ \dots,\ t_i^{\alpha-1} + 2\lambda, \\ \dots\dots\dots\dots\dots\dots \end{array}$$

Note that if π is an integer, then the results of Section 2 yield that a $(\pi,1)$ periodic schedule exists for synchronous computation graphs. Theorem 14 suggests the following question: Let $\pi = \lambda/\alpha$ where λ and α are integers. Does a synchronous computation graph G have a (λ,α) periodic execution? Clearly, if the answer is in the affirmative, then under this execution G computes at the maximal rate $1/\pi$. The principal result of this section is to provide just such an execution for a wide class of graphs. We shall treat the cases $\pi < 1$, $\pi \geq 1$ separately.

A. <u>$\pi < 1$</u>

Then we cannot hope to achieve a computation rate of $1/\pi$ since $1/\pi > 1$ whereas a node can initiate at most once per time unit. However, this maximal rate of one initiation per unit time for a synchronous computation graph can be achieved with $\gamma = 1$ under the assignment of Section 2B.

B. $\pi \geq 1$

For any real number x we write $\lceil x \rceil$ to be the smallest integer containing x, and $\lfloor x \rfloor$ to be the largest integer contained in x.

Let G be a synchronous computation graph. Consider the following assignment of integers t_i^r to the nodes n_i of G:

$$t_i^r = \lceil \pi r + t_i \rceil \qquad r=0,1,\ldots,$$

where t_i is obtained as in Section 2B.

Lemma 16

i) $t_i^{r+1} > t_i^r$

ii) $t_i^{k\alpha+\beta} = k\lambda + t_i^\beta \qquad k=0,1,\ldots, \quad \beta=0,1,\ldots,$

PROOF:

(i) Write $\pi = \lambda/\alpha = \lfloor \pi \rfloor + R/\alpha \qquad 0 \leq R \leq \alpha-1$

Then $\lceil \pi r + t_i \rceil = \lceil \lfloor \pi \rfloor r + Rr/\alpha + t_i \rceil$

$= \lfloor \pi \rfloor r + \lceil Rr/\alpha + t_i \rceil$

Therefore

$t_i^{r+1} - t_i^r = \lfloor \pi \rfloor + \lceil (r+1)R/\alpha + t_i \rceil - \lceil rR/\alpha + t_i \rceil$

$\geq \lfloor \pi \rfloor$

≥ 1 since $\pi \geq 1$.

(ii) $t_i^{k\alpha+\beta} = \lceil (k\alpha+\beta)\lambda/\alpha + t_i \rceil$

$= \lceil k\lambda + \beta\, \lambda/\alpha + t_i \rceil$

$= k\lambda + \lceil \beta\, \lambda/\alpha + t_i \rceil$

$= k\lambda + t_i^\beta$

Theorem 17

Let G be synchronous computation graph with $\pi = \lambda/\alpha$, λ and α positive integers. Then G has a (λ,α) periodic execution.

PROOF: Suppose $n_i \rightarrow n_j$. Then for r=0,1,..., we have

$$t_i^r + \tau_i = \lceil \pi r + t_i \rceil + \tau_i$$

$$= \lceil \pi r + t_i + \tau_i \rceil$$

since τ_i is an integer

$$\leq \lceil \pi r + t_j + \pi A_{ij} \rceil \qquad \text{by Lemma 1}$$

$$= \lceil \pi(r + A_{ij}) + t_j \rceil$$

$$= t_j^{r+A_{ij}}$$

Then by Theorem 2 there exists an execution of G such that node n_k initiates only at times $t_k^0, t_k^1, \ldots, t_k^r, \ldots,$. The result now follows by Lemma 16.

It is clear that for a synchronous computation graph $\rho_G = \alpha/\lambda = 1/\pi$ so that G computes at the maximal rate.

The following computation aid may be found useful: If $n_i \rightarrow n_j$, and n_i, n_j both are on the same loop L for which $\pi_L = \pi$, then

$$t_i^r = t_j^{r+A_{ij}} - \tau_i \qquad r=0,1,\ldots,$$

PROOF: Let L be $n_i \rightarrow n_j \rightarrow n_k \rightarrow \ldots \rightarrow n_p \rightarrow n_m \rightarrow n_i$. Then by the proof of Theorem 17,

$$t_i^r \leq t_j^{r+A_{ij}}$$

$$t_j^{r+A_{ij}} \leq t^{r+A_{ij}+A_{jk}} - \tau_j$$

$$\cdots\cdots\cdots\cdots$$

$$t_m^{r+A_{ij}+\ldots+A_{pm}} \leq t_i^{r+A_{ij}+\ldots+A_{pm} + A_{mi}} - \tau_m$$

Therefore $t_i^r \leq t_i^{r+\sum_L A} - \sum_L \tau$ by adding the above inequalities

$$= \lceil \pi(r+\sum_L A) + t_i \rceil - \sum_L \tau$$

$$= \lceil \pi r + t_i + \sum_L \tau \rceil - \sum_L \tau$$

$$= \lceil \pi r + t_i \rceil \quad \text{since } \sum_L \tau \text{ is an integer}$$

$$= t_i^r$$

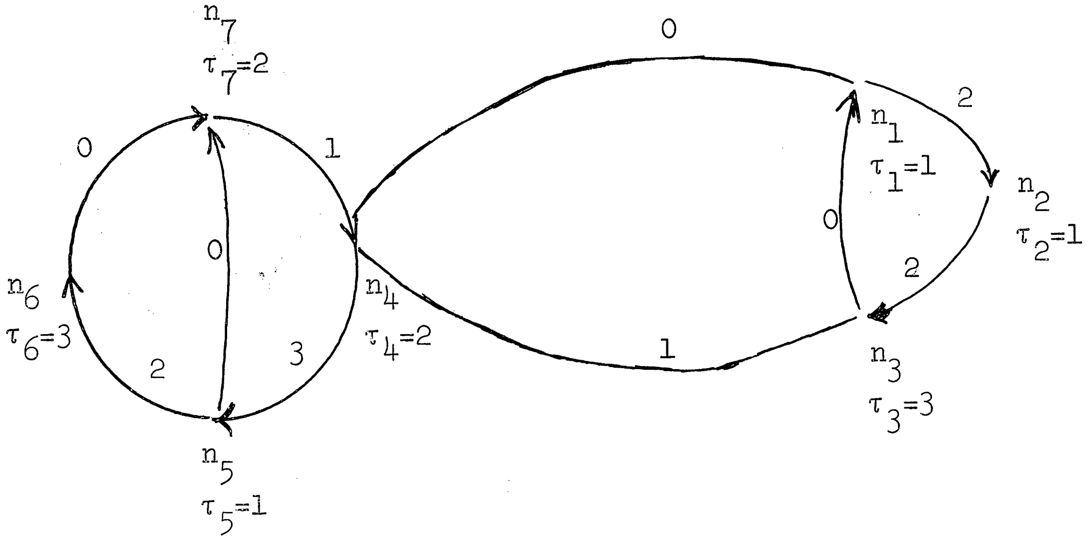
i.e. equality must hold for each of the above inequalities and in particular, for the first.

The obvious dual results hold for synchronous computation graphs under the substitution of T_i for t_i, i.e.

$$T_i^r = \lceil \pi r + T_i \rceil \qquad r=0,1,\ldots,$$

defines a (λ,α) periodic execution.

An Example



$\pi = 8/6 = 4/3$ corresponding to the loop $n_4 \to n_5 \to n_6 \to n_7 \to n_4$.

Take $\lambda = 4$, $\alpha = 3$.

By the methods of Section 2 we obtain:

$t_1 = 0$ $\quad t_2 = -4/3$ $\quad t_3 = -3$ $\quad t_4 = -11/3$ $\quad t_5 = -17/3$

$t_6 = -22/3$ $\quad t_7 = -13/3$.

Hence

$t_1^0 = 0$	$t_1^1 = 2$	$t_1^2 = 3$
$t_2^0 = -1$	$t_2^1 = 0$	$t_2^2 = 2$
$t_3^0 = -3$	$t_3^1 = -1$	$t_3^2 = 0$
$t_4^0 = -3$	$t_4^1 = -2$	$t_4^2 = -1$
$t_5^0 = -5$	$t_5^1 = -4$	$t_5^2 = -3$
$t_6^0 = -7$	$t_6^1 = -6$	$t_6^2 = -4$
$t_7^0 = -4$	$t_7^1 = -3$	$t_7^2 = -1$

5. Conclusion

A maximal computation rate periodic schedule of node initiation times has been presented for computation graphs G with U = W = T = 1 for all branches of G. This schedule is of two forms, according as G computes asynchronously or synchronously. Furthermore, an analysis has been given of the so-called free running execution of G and this is found to yield the maximum computation rate of G.

An open problem is a similar analysis for non-terminating computation graphs G in which U, W, and T are not restricted to be 1. Another area for future research is as follows:

Suppose all nodes of G are capable of performing the same functions e.g. each node is a computer. Then under the various periodic schedules of this paper, what is the minimum number v of "computers" necessary to perform the computation without decreasing the computation rate of G?

A more general problem is the following: Suppose the node set of G to be partitioned into sets $N_1, N_2, \ldots, N_m$ where all of the nodes n in N_i are capable of performing the same functions. Again, what is the minimum number of "computers" necessary to perform the computation under the various periodic schedules without decreasing the computation rate of G? A related problem is the following: Suppose $k < v$ "computers" are available. How can we choose the minimum value of γ such that the computation may be effected with period γ and such that the k "computers" are utilized.

References

[1] Cuninghame-Green, R. A., Describing industrial processes with interference and approximating their steady-state behavior. Operational Research Quarterly, vol. 13, No. 1.

[2] Ford, L. R., Jr. and Fulkerson, D. P., Flows in Networks (Princeton: Princeton University Press, 1962), pp. 131-133.

[3] Karp, R. M. and Miller, R. E., Properties of a model for parallel computations: Determinacy, termination, queueing. IBM Research Paper RC-1285.

[4] Lawler, E. L., Extremal cycles in weighted linear graphs. Unpublished paper, Systems Engineering Laboratory of The University of Michigan, Ann Arbor.

[5] Reiter, R., A study of a model for parallel computation. The University of Michigan's Information Systems Laboratory Technical Report ISL-65-4.