Scheduling Parallel Computations

RAYMOND REITER*

University of Michigan,† Ann Arbor, Michigan

ABSTRACT. A model for parallel computations is given as a directed graph in which nodes represent elementary operations, and branches, data channels. The problem considered is the determination of an admissible schedule for such a computation; i.e. for each node determine a sequence of times at which the node initiates its operation. These times must be such that each node, upon initiation, is assured of having the necessary data upon which to operate. Necessary and sufficient conditions that a schedule be admissible are given. The computation rate of a given admissible schedule is defined and is shown to have a limiting value $1/\pi$ where π is a parameter dependent upon the cycles in the graph. Thus, the computation cannot proceed at a rate exceeding $1/\pi$. For $\gamma \geq \pi$, the class of all periodic admissible schedules with period γ is characterized by the solution space of a certain system of linear inequalities. In particular, then, the maximum computation rate of $1/\pi$ is attainable under a periodic admissible schedule with period π. A class of all-integer admissible schedules is given. Finally, an algorithm is given for the determination of the number of initiations of each node in the graph defining a parallel computation.

An example for a system of difference equations is given in detail.

KEY WORDS AND PHRASES: parallel computation, parallel algorithms, iterative algorithms, directed graph representation, multiple processors, processor initiation multiplicity, scheduling processor initiations, periodic initiation times, maximum computation rate, integer schedules

CR CATEGORIES: 5.9

1. *Introduction*

Virtually all models for parallel computations which have appeared in the literature represent the calculation as a directed graph in which nodes and branches represent, respectively, processing units (elementary operations) and data channels (precedence constraints) (see, e.g., [1–5]). It is of interest to determine a schedule for the execution of such a parallel algorithm, i.e. for each processor, to determine a sequence of times at which the processor initiates its operation. These times must be such that a given processor, upon initiation, is assured of having the necessary data upon which to operate. Such a schedule thus provides a very simple control of the parallel computation—merely signal each processor to begin at its appropriate times, quite independently of what the remaining processors in the system are doing.

The above-mentioned models for parallel computations may be distinguished according as they do (see [1, 3, 4]) or do not (see [2, 5]) allow branching, i.e. nodes which perform a decision as to which successor nodes take part in the calculation.

* Present address: The Institute of Computer Science, University of London, London, England.

† Systems Engineering Laboratory, Department of Electrical Engineering. This research was supported by Air Force contract AF 30(602)-3546.

In the case that branching is permitted, it is clear that no a priori schedule can be determined, since this is necessarily dependent on the initial data. Otherwise, such a schedule does exist. In this case, if the graph defining the computation is acyclic (see [2]), the determination of such a schedule is trivial. However, if the computation is iterative, i.e. the corresponding graph has cycles, the determination of a schedule is no longer obvious. Karp and Miller [5] have formulated a model for this class of parallel computations (i.e. which do not involve branching). In this paper scheduling processor initiations are considered for a special case of their model.

For our purposes, then, a parallel algorithm is specified by a directed graph G called a *computation graph.* G is given by

(i) a node set $N = \{n_1, n_2, \cdots, n_l\}$,

(ii) branches $d_1, d_2, \cdots, d_t$, where any given branch $d_p = (n_i, n_j)$ is directed from a specified node n_i to a specified node n_j,

(iii) two nonnegative integers A_{ij}, U_{ij}, where $U_{ij} \in \{0, 1\}$, and a positive real number τ_{ij}, associated with each branch $d_p = (n_i, n_j)$.

A node of G represents an operation in the computation or, equivalently, a processor assigned to perform that operation. Each branch $d_p = (n_i, n_j)$ represents a queue of data directed from the processor assigned to n_i to the processor associated with n_j. Thus the processor at n_i places the results of its calculations onto branch d_p and the queue of data on d_p is available as input to the processor at n_j. The parameters in (iii) are to be interpreted as follows:

A_{ij} = the initial number of data words on branch d_p;

U_{ij} = the number of data words placed on d_p upon the termination of the operation associated with n_i;

τ_{ij} = the time required by node n_i to place the result of its operation on branch (n_i, n_j). If n_i initiates at time t, then at time $t + \tau_{ij}$, n_i places U_{ij} data words on branch (n_i, n_j).

Dynamically, a computation graph G functions as follows. Whenever every input branch to node n_i contains at least one data item, n_i becomes eligible for initiation. n_i is not required to initiate the moment that it becomes eligible; if at that time it does not initiate, then it must do so at some future time. Upon initiation, say at time t, n_i removes one data item from each of its input branches; if $d_p = (n_i, n_j)$ is an output branch from n_i, then at time $t + \tau_{ij}$, n_i places U_{ij} data items upon d_p. The computation defined by G terminates if there exists a time T such that for all $t \geq T$, no node of G is capable of initiating. Otherwise, G is nonterminating. These conditions are formalized in [5]; we give a different characterization in Section 3 when we define an admissible schedule.

Denote by X_i the number of initiations of n_i during the computation defined by G. Thus $0 \leq X_i \leq \infty$. In Section 2 a simple algorithm is provided for determining these numbers. In Section 3 the class of all admissible schedules is characterized, and a maximum computation rate periodic schedule for G is provided.

2. *Determining* $\{X_i\}$

We assume that each node of G initiates at least once, i.e. $X_i \geq 1$ for each n_i of G. This assumption is equivalent [5] to the requirement

$$\sum_{(n_i,n_j)\in C} A_{ij} \geq 1 \tag{2.1}$$

for each directed cycle C of G.

THEOREM 2.1. *Suppose $X_k \geq 1$ for each node n_k of G and that $\{A_{ij} \mid U_{ij} = 0\} \neq \phi$. Without loss of generality, suppose $A_{r1} = \min_{i,j}\{A_{ij} \mid U_{ij} = 0\}$. Then $X_1 = A_{r1}$.*

PROOF. Clearly, $X_1 \leq A_{r1}$. For an arbitrary node n_k, define $\alpha_k = \min_i\{A_{ik} \mid U_{ik} = 0\}$ if this set is nonempty, $\alpha_k = \infty$ otherwise. Then, in particular, $\alpha_1 = A_{r1}$. In general,

$$X_k = \min\{\alpha_k, \min_i\{X_i + A_{ik} \mid U_{ik} = 1\}\}.$$

Suppose, contrary to the statement of Theorem 2.1, that $X_1 < A_{r1} = \alpha_1$. Then

$$X_1 = \min_i\{X_i + A_{i1} \mid U_{i1} = 1\},$$
$$= X_{j_1} + A_{j_11}, \quad \text{say.}$$

Since $X_1 < \infty$, $X_{j_1} < \infty$. We prove that $X_{j_1} \neq \alpha_{j_1}$—for otherwise $X_{j_1} = \alpha_{j_1} = A_{ij_1}$ for some i. But then $X_1 = X_{j_1} + A_{j_11} = A_{ij_1} + A_{j_11} \geq A_{j_11} \geq A_{r1}$, contradicting the assumption $X_1 < A_{r1}$. It follows that

$$X_{j_1} = \min_i\{X_i + A_{ij_1} \mid U_{ij_1} = 1\},$$
$$= X_{j_2} + A_{j_2j_1}, \quad \text{say,}$$

i.e.

$$X_1 = X_{j_2} + A_{j_2j_1} + A_{j_11} \geq X_{j_2},$$

and a similar argument shows that $X_{j_2} \neq \alpha_{j_2}$. We can repeat this process, obtaining a sequence of nodes $n_{j_m}, n_{j_{m-1}}, \cdots, n_{j_1}, n_{j_0} = n_1$ with

$$X_{j_i} = X_{j_{i-1}} + A_{j_i,j_{i-1}}, \qquad i = 1, 2, \cdots, m. \tag{2.2}$$

Eventually some node must repeat, yielding a cycle C of G. If we add those equations of (2.2) corresponding to the branches of C, we obtain $\sum_{(n_i,n_j)\in C} A_{ij} = 0$, contradicting (2.1).

COROLLARY 2.1. *The following algorithm yields $\{X_i \mid i = 1, 2, \cdots, l\}$ when $X_i \geq 1$, $i = 1, 2, \cdots, l$.*

1. *If $\{A_{ij} \mid U_{ij} = 0\} = \phi$, then each $X_i = \infty$. This follows from the results in* [5].
2. *Otherwise, let n_j be such that $A_{rj} = \min_{i,k}\{A_{ik} \mid U_{ik} = 0\}$.*

Put $X_j = A_{rj}$. If (n_j, n_k) is a branch of G, add a branch (n_k, n_k) with $U_{kk} = 0$ and initial data $A_{kk} = A_{jk} + X_j$. Remove n_j and all of its input and output branches. If the resulting graph is the null graph, we are through. If the resulting graph H has no branch (n_i, n_j) with $U_{ij} = 0$, put $X_k = \infty$ for all nodes n_k of H. Otherwise, return to (1) *with H.*

3. *Scheduling*

We formalize the notion of "valid node initiation times" given in Section 1. As before, we assume that $X_i \geq 1$ for each node n_i of G. A *schedule* is a set $\sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_l\}$, where each σ_i is a function $\sigma_i : \{1, 2, \cdots, X_i\} \rightarrow R$ such that for

$1 \leq k < r \leq X_i$, $\sigma_i(k) < \sigma_i(r)$. Here R is the set of real numbers. With each node n_i we associate a function $x_i : R \rightarrow \{0, 1, \cdots, X_i\}$.

$x_i(t) = 0$ if and only if $t < \sigma_i(1)$;

$x_i(t) = k$ for $1 \leq k < X_i$ if and only if $\sigma_i(k) \leq t < \sigma_i(k + 1)$;

$x_i(t) = X_i$ if and only if $\sigma_i(X_i) \leq t$.

For every branch (n_i, n_j) define

$$b_{ij}^{\sigma}(t) = A_{ij} + U_{ij}x_i(t - \tau_{ij}) - [x_j(t) - \epsilon_j(t)],$$

where

$\epsilon_j(t) = 1$ if there exists k, $1 \leq k \leq X_j$, such that $\sigma_j(k) = t$,
$\epsilon_j(t) = 0$ otherwise.

A schedule σ is called an *admissible schedule* if, for $j = 1, 2, \cdots, l$, $\sigma_j(k) = t$ implies $b_{ij}^{\sigma}(t) \geq 1$ for all branches (n_i, n_j) into n_j, and for all k, $1 \leq k \leq X_j$.

These definitions are to be interpreted as follows:

$\sigma_i(k) = t$ means that node n_i begins its kth initiation at time t under the schedule σ.

$x_i(t)$ is the number of initiations of node n_i, up to and including time t, under the schedule σ.

$b_{ij}^{\sigma}(t)$ is thus the number of data items on branch (n_i, n_j) at time t under the schedule σ.

An admissible schedule specifies those node initiation times corresponding to the presence of at least one data item on each of the node input branches. Thus n_j initiates at time t ($\sigma_j(k) = t$ for some k, $1 \leq k \leq X_j$) only if each branch (n_i, n_j) directed into n_j contains at least one data item ($b_{ij}^{\sigma}(t) \geq 1$).

A branch (n_i, n_j) is *essential* if $X_j > A_{ij}$. Otherwise it is *inessential*. In particular, every branch (n_i, n_j) with $U_{ij} = 0$ is inessential. The following theorem characterizes the class of admissible schedules for G.

THEOREM 3.1. *A schedule σ is admissible if and only if for each essential branch (n_i, n_j) of G,*

$$\sigma_i(r) + \tau_{ij} \leq \sigma_j(r + A_{ij}), \qquad r = 1, 2, \cdots, X_j - A_{ij}.$$

PROOF. *Sufficiency.* The proof is by contradiction. Let σ be admissible and suppose there exists an essential branch (n_i, n_j) and an integer r, $1 \leq r \leq X_j - A_{ij}$, such that

$$\sigma_i(r) + \tau_{ij} > \sigma_j(r + A_{ij}). \tag{3.1}$$

Let $t = \sigma_j(r + A_{ij})$ so that $x_j(t) = r + A_{ij}$. Then

$$\begin{aligned} b_{ij}^{\sigma}(t) &= A_{ij} + U_{ij}x_i(t - \tau_{ij}) - [x_j(t) - 1], \\ &= 1 - r + U_{ij}x_i(t - \tau_{ij}). \end{aligned}$$

Since (n_i, n_j) is essential, $U_{ij} = 1$. Moreover, by (3.1), $t - \tau_{ij} < \sigma_i(r)$so that $x_i(t - \tau_{ij}) < r$. Hence $b_{ij}^{\sigma}(t) < 1$, which contradicts the admissibility of σ.

Necessity. We must prove, for any node n_j of G and for $k = 1, 2, \cdots, X_j$, that $b_{ij}^{\sigma}(\sigma_j(k)) \geq 1$ for every branch (n_i, n_j) directed into n_j. If (n_i, n_j) is inessential this remark is valid. Otherwise $k = r + A_{ij}$ for some r, $1 \leq r \leq X_j - A_{ij}$. Put $t = \sigma_j(k)$. Then by assumption $t - \tau_{ij} \geq \sigma_i(r)$. Thus $x_i(t - \tau_{ij}) \geq r$. Hence

$$b_{ij}^{\sigma}(t) = A_{ij} + x_i(t - \tau_{ij}) - [x_j(t) - 1] \geq A_{ij} + r - [r + A_{ij} - 1],$$
$$= 1,$$

which establishes the theorem.

A. Periodic Admissible Schedules. The actual implementation of the algorithm defined by a computation graph requires some central control unit whose function will be to signal, at appropriate times, the various processor initiations. The simplest possible control unit would signal these initiations at periodic intervals, the period being the same for each processor. To that end, we make the following definition: An admissible schedule σ is *periodic* with period $\gamma > 0$ if there exist real numbers t_i such that $\sigma_i(k) = t_i + (k - 1)\gamma, \quad k = 1, 2, \cdots, X_i$.

As an immediate corollary to Theorem 3.1 we have

Corollary 3.1. *G has an admissible periodic schedule with period γ if and only if there exist real numbers t_i, $i = 1, 2, \cdots, l$, which satisfy*

$$t_j - t_i \geq \tau_{ij} - \gamma A_{ij} \tag{3.2}$$

for every essential branch (n_i, n_j) of G.

In particular, if G has an admissible periodic schedule with period γ, then $\gamma \geq \max \{\tau_{ii}/A_{ii}\}$, the max taken over all essential branches of the form (n_i, n_i). Let G' be the graph obtained from G by removing all inessential branches, all essential branches of the form (n_i, n_i), and all the resulting (if any) isolated nodes (i.e. nodes without input and output branches). Suppose G' has m nodes $n_1, n_2, \cdots, n_m$ and n branches $d_1, d_2, \cdots, d_n$. Let $A' = (\alpha_{rs})$ be the edge-node incidence matrix of G':

$\alpha_{rs} = 1$ if branch d_r is directed into n_s;
$\alpha_{rs} = -1$ if branch d_r is directed out from n_s;
$\alpha_{rs} = 0$ otherwise.

Let $\mathbf{t}$ be a column vector with rth component t_i, $i = 1, 2, \cdots, m$, and let $\mathbf{a}$ be a column vector with rth component $a_r = \tau_{ij} - \gamma A_{ij}$, where $d_r = (n_i, n_j)$. Then by Corollary 3.1 and the above remarks, the following can be stated:

G has a periodic admissible schedule with period γ if and only if
(i) $\gamma \geq \max_i \{\tau_{ii}/A_{ii}\}$, and
(ii) there exists $\mathbf{t}$ such that

$$A'\mathbf{t} \geq \mathbf{a}. \tag{3.3}$$

By [6, Cor. 1, p. 157], (3.3) has a solution if and only if, for every cycle C of G', $\sum_C a_i \leq 0$, with summation over the branches d_i of C. Thus (3.3) has a solution if and only if

$$\gamma \geq \max_{C \in G'} \left\{ \frac{\sum_{(n_i,n_j) \in C} \tau_{ij}}{\sum_{(n_i,n_j) \in C} A_{ij}} \right\}.$$

Let us define the *maximum cycle ratio* of G,

$$\pi = \max \left\{ \frac{\sum_{(n_i,n_j) \in C} \tau_{ij}}{\sum_{(n_i,n_j) \in C} A_{ij}} \right\},$$

the max over all cycles C of G (including cycles of the form (n_i, n_i)) consisting of essential branches. We then have

COROLLARY 3.2. *G has a periodic admissible schedule with period γ if and only if* (i) *G has no cycles consisting of essential branches and $\gamma > 0$, or* (ii) *G has at least one cycle consisting of essential branches and $\gamma \geq \pi$.*

Case (i) would not often be encountered, at least not in the context of parallel computation, since, under these conditions, a node is permitted to initiate before it has terminated its previous initiation. The usual situation is to require termination before the next initiation. Such a constraint corresponds to computation graphs G for which each node n_i has (implicitly) a branch (n_i, n_i) with $A_{ii} = U_{ii} = 1$ and $\tau_{ii} = \max\{\tau_{ij}\}$, the max taken over all branches (n_i, n_j) out from n_i.

π thus emerges as the minimum possible period of any periodic admissible schedule. Define the *computation rate* of a node n_i, under an admissible schedule σ, to be

$$\rho_i^\sigma = \frac{X_i}{\sigma_i(X_i) - \sigma_i(1)} \quad \text{if} \quad X_i < \infty,$$

$$= \lim_{t\to\infty} \frac{t}{\sigma_i(t)} \qquad \text{if} \quad X_i = \infty,$$

and this limit exists. In particular, under a periodic admissible schedule σ', with period π,

$$\rho_i^{\sigma'} = \frac{1}{\pi}\left(\frac{X_i}{X_i - 1}\right). \tag{3.4}$$

Let n_1 be a node which lies on a cycle $C: n_1, n_2, \cdots, n_r$, with maximum cycle ratio π. Let σ be any admissible schedule for G where without loss of generality we assume that $\sigma_1(1) = 0$. Finally, suppose that $X_1 = k\alpha + \nu$, $k \geq 1$, $0 \leq \nu \leq \alpha - 1$, where $\alpha = \sum_{(n_i,n_j)\in C} A_{ij}$. Then by Theorem 3.1 the following inequalities hold:

$$\sigma_1(\nu) + \tau_{12} \leq \sigma_2(\nu + A_{12}),$$
$$\sigma_2(\nu + A_{12}) + \tau_{23} \leq \sigma_3(\nu + A_{12} + A_{23}),$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$
$$\sigma_r(\nu + A_{12} + \cdots + A_{r-1,r}) + \tau_{r,1} \leq \sigma_1(\nu + \alpha),$$
$$\sigma_1(\nu + \alpha) + \tau_{12} \leq \sigma_2(\nu + \alpha + A_{12}),$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$
$$\sigma_r(\nu + (k-1)\alpha + A_{12} + \cdots + A_{r-1,r}) + \tau_{r,1} \leq \sigma_1(\nu + k\alpha).$$

Adding these inequalities yields

$$\sigma_1(\nu) + k \sum_{(n_i,n_j)\in C} \tau_{ij} \leq \sigma_1(X_1),$$

i.e.

$$k \sum_{(n_i,n_j)\in C} \tau_{ij} \leq \sigma_1(X_1).$$

Using the relation $\sum_{(n_i,n_j)\in C} \tau_{ij} = \pi\alpha$, we obtain

$$\rho_1^\sigma \leq \frac{1}{\pi}\left(\frac{X_1}{X_1 - \nu}\right).$$

Comparing with (3.4), we see that asymptotically (i.e. $X_1 \to \infty$),

$$\rho_1^{\sigma} \sim \leq \rho_1^{\sigma'} \sim \frac{1}{\pi}.$$

Thus, asymptotically, n_1 cannot achieve a computation rate greater than that attained under a periodic admissible schedule σ' with period π. Moreover, this maximum computation rate is $1/\pi$. In this sense then, σ' provides a best admissible schedule for G, with respect to both computation speed and ease of processor control.

B. Solutions of $A'\mathbf{t} \geq \mathbf{a}$. Define

$$d(i, j) = \max_{p(i,j)} \left\{ \sum_{(n_k, n_r) \in p(i,j)} a_{kr} \right\},$$

the max taken over all paths $p(i, j)$ from n_i to n_j of G'. $d(i, j) = -\infty$ if no such path exists. $d(i, j)$ is thus the maximum path length from n_i to n_j.

A node n_i is a *source* if there exists a path from n_i to every other node of G'. It follows [6, 9.5, p. 182] that if G' has a source n_1, say, then a solution of this system when $\gamma \geq \pi$ is given by

$$t_1 = 0, \qquad t_i = d(1, i), \quad i = 2, \cdots, m.$$

A node n_i is a *sink* if there exists a path from every other node of G' to n_i. Dually, if n_1 is a sink, a solution of this system is given by

$$T_1 = 0, \qquad T_i = -d(i, 1), \quad i = 2, \cdots, m.$$

The condition $\sum_{d_r \in C} a_r \leq 0$ for every cycle C of G', i.e. the condition $\gamma \geq \pi$, provides an efficient algorithm for determining t_i or T_i [6, 9.4, p. 180]. An algorithm for determining the parameter π is given in [7]. For solutions in the case of more general graphs G', and a more extensive analysis of the solution space of $A'\mathbf{t} \geq \mathbf{a}$, the reader is referred to [8].

C. Integer Admissible Schedules. Suppose that the node initiation times of G are to be governed by a clock signal. Then these times are constrained to be integer multiples of the clock period. Thus, it is of interest to determine an *integer admissible schedule* for G, i.e. an admissible schedule σ such that $\sigma_i(k)$ is an integer $i = 1, 2, \cdots, l,$ $k = 1, 2, \cdots, X_i$. To that end, we assume that τ_{ij} is a positive integer for each branch (n_i, n_j) of G. We also assume that no node of G is permitted to initiate until it has terminated its previous initiation; i.e. if σ is an admissible schedule, then $\sigma_i(r) + \tau_i \leq \sigma_i(r + 1)$, $i = 1, 2, \cdots, l,$ $r = 1, 2, \cdots, X_{i-1}$, where $\tau_i = \max\{\tau_{ij}\}$, the max taken over all branches (n_i, n_j) out from n_i. In particular, then,

$$\ulcorner \sigma_i(r) \urcorner < \ulcorner \sigma_i(r + 1) \urcorner, \tag{3.5}$$

where for any real number x we write $\ulcorner x \urcorner$ to be the smallest integer containing x.

Corollary 3.3. *Let σ be an admissible schedule for G. Then σ' defined by $\sigma'_i(r) = \ulcorner \sigma_i(r) \urcorner$, $i = 1, 2, \cdots, l,$ $r = 1, 2, \cdots, X_i$, is an integer admissible schedule.*

Proof. By (3.5), σ' is a schedule. We prove it is admissible. To that end, suppose that (n_i, n_j) is an essential branch of G and consider, for $r = 1, 2, \cdots,$ $X_j - A_{ij}$,

$$\begin{aligned}\sigma_i'(r) + \tau_{ij} &= \lceil \sigma_i(r) \rceil + \tau_{ij}, \\ &= \lceil \sigma_i(r) + \tau_{ij} \rceil \qquad \text{since } \tau_{ij} \text{ is an integer,} \\ &\leq \lceil \sigma_j(r + A_{ij}) \rceil \qquad \text{by Theorem 3.1,} \\ &= \sigma_j'(r + A_{ij}).\end{aligned}$$

Hence, by Theorem 3.1, σ' is an admissible schedule.

Note that σ and σ' of Corollary 3.3 yield the same asymptotic node computation rates for G. Let us consider the form of σ' when σ is the periodic admissible schedule of subsection 3-A. Clearly, if γ is an integer, σ' assumes the form

$$\sigma_i'(k) = \lceil t_i \rceil + (k - 1)\gamma, \qquad k = 1, 2, \cdots, X_i,$$

which is still periodic. If γ is not an integer, but rational, say $\gamma = \lambda/\alpha$ with λ, α positive integers, it is easy to see that

$$\sigma_i'(k\alpha + \beta) = k\lambda + \sigma_i'(\beta), \qquad \beta = 0, 1, \cdots, \alpha - 1, \quad 1 \leq k\alpha + \beta \leq X_i,$$

where each $\sigma_i'(\beta)$ is an integer satisfying

$$\sigma_i'(0) < \sigma_i'(1) < \cdots < \sigma_i'(\alpha - 1) < \sigma_i'(0) + \lambda.$$

Thus, under σ', each node of G has a "fundamental period" λ, during which it initiates α times. Clearly, under σ' every node of G has asymptotic computation rate $1/\gamma$.

D. Example. Computation graphs are ideal for representing systems of difference equations. As an example, consider the system

$$x_{i+1} = y_i z_i - x_i, \qquad y_{i+1} = |x_i + y_i| + x_i z_i, \qquad z_{i+1} = x_i y_i / z_i,$$

with x_0, y_0, z_0 given. Suppose the values x_N, y_N, z_N are desired for some fixed $N \geq 1$. A possible computation graph is the following (see Figure 1). All branches have $U = 1$ unless otherwise indicated. Branches for which $A \neq 0$ are labeled with the appropriate value, e.g. $1(x_0)$ means that the corresponding branch has $A = 1$ and that initial data item is the value x_0. Branches labeled N, $U = 0$ are dummies for which $A = N$; their function is to terminate the computation upon depletion of their queues. Upon termination, the branches (n_2, n_2), (n_6, n_1), and (n_8, n_8) contain the values x_N, y_N, and z_N, respectively.

Assume that the times required by the various node functions are: addition and subtraction, 1 time unit; absolute value, 2 time units; multiplication, 3 time units; division, 4 time units. Then $\pi = \frac{11}{2}$, corresponding to the cycles n_1, n_2, n_7, n_8, n_1 and n_3, n_6, n_7, n_8, n_3. Applying Corollary 2.1, we obtain $X_1 = X_2 = X_6 = X_8 = N$, $X_3 = X_4 = X_5 = X_7 = N + 1$. If we choose n_1 as a source we obtain, with $\gamma = \pi$, $t_1 = 0$, $t_2 = 3$, $t_3 = 0$, $t_4 = -\frac{3}{2}$, $t_5 = -\frac{1}{2}$, $t_6 = 3$, $t_7 = -\frac{3}{2}$, $t_8 = \frac{3}{2}$. If we choose n_8 as a sink, we obtain a different set of starting times: $T_1 = -\frac{3}{2}$, $T_2 = \frac{3}{2}$, $T_3 = -\frac{3}{2}$, $T_4 = -\frac{3}{2}$, $T_5 = -\frac{1}{2}$, $T_6 = \frac{3}{2}$, $T_7 = -3$, $T_8 = 0$. Thus, for example, under this schedule, n_1 initiates at times $-\frac{3}{2}$, 4, $\frac{19}{2}$, 15, $\cdots$, $-\frac{3}{2} + 11(N - 1)/2$. If an all-integer schedule is required with asymptotic computation rate $1/\pi$, we obtain the two schedules σ', σ'' defined by

$$\sigma_i'(r) = \lceil \pi(r - 1) + t_i \rceil, \qquad \sigma_i''(r) = \lceil \pi(r - 1) + T_i \rceil.$$

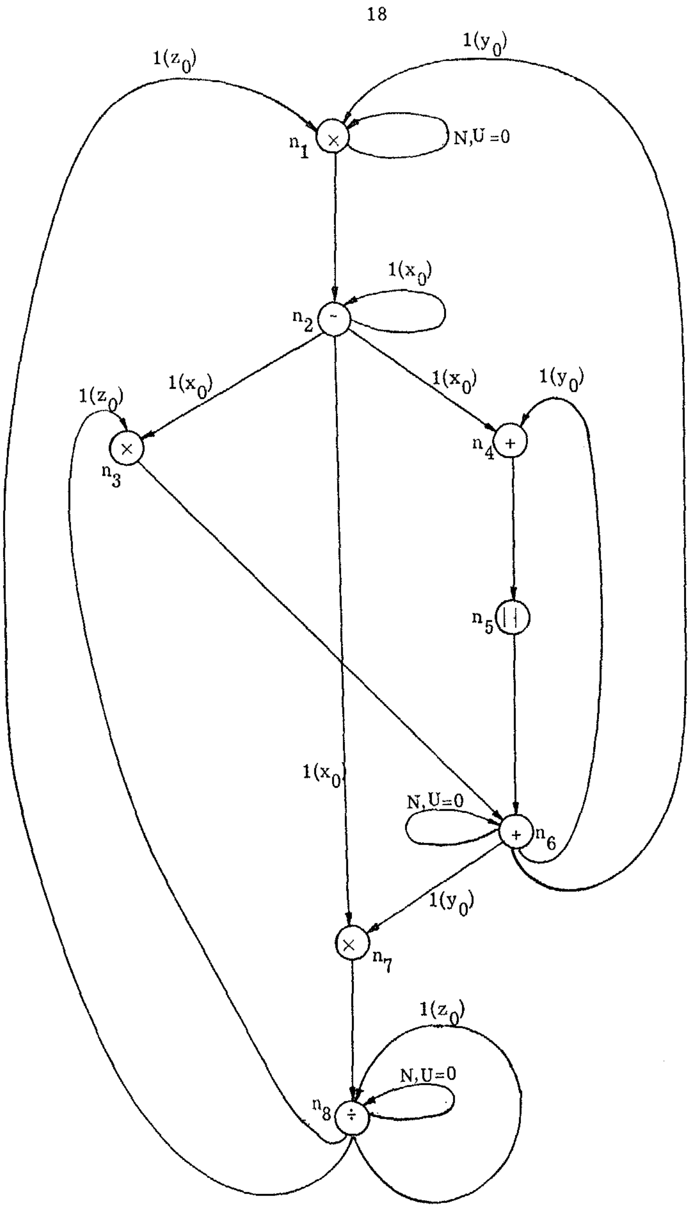

FIG. 1

Thus, for example, under σ'', n_1 initiates at times -1, 4, 10, 15, 21, 27, $\cdots$, $\lceil -\frac{3}{2} + 11(N-1)/2 \rceil$.

ACKNOWLEDGMENT. The author wishes to thank Dr. R. M. Karp of IBM Corporation for many fruitful discussions connected with this work.

REFERENCES

1. ESTRIN, G., BUSSELL, B., TURN, R., AND BIBB, J. Parallel processing in a restructurable computer system. *IEEE Trans. EC-12* (1963), 747–755.

2. DORN, W. S., HSU, N. C., AND RIVLIN, T. J. Some mathematical aspects of parallel computation. RC-647, IBM Research Center, Yorktown Heights, N. Y., 1962.
3. HELLER, J. Sequencing aspects of multiprogramming. *J. ACM 8*, 3 (July 1961), 426–439.
4. SCHWARTZ, E. S. An automatic sequencing procedure with application to parallel programming. *J. ACM 8*, 4 (Oct. 1961), 513–537.
5. KARP, R. M., AND MILLER, R. E. Properties of a model for parallel computations: Determinacy, termination, queueing. *SIAM J. Appl. Math. 14* (1966), 1390–1411.
6. BERGE, C., AND GHOUILA-HOURI, A. *Programming, Games and Transportation Networks.* Wiley, New York, 1965.
7. DANTZIG, G. B., BLATTNER, W. O., AND RAO, M. R. Finding a cycle in a graph with minimum cost to time ratio with application to a ship routing problem. Tech. Rep. 66-1, Operations Res. House, Stanford U., Stanford, Calif., Nov. 1966.
8. REITER, R. A study of a model for parallel computation. Ph.D. diss., Dep. of Commun. Sciences, U. of Michigan, Ann Arbor, Mich., June 1967.

RECEIVED JANUARY, 1968