Abstract—An interesting optimization problem is examined where optimal load allocations depend on processor release times, but the timing of release times depends on the load allocation scenario used. This circular relationship between an optimal solution and release times causes system models whose timing relationships may not have an optimal solution. To obtain an optimal solution based on an assumed model and its arbitrary release times, we propose an exhaustive search algorithm as a starting point into open ended research on this topic on algorithmic scalability and feasibility conditions. Through simulation, the behavior of the exhaustive search algorithm is investigated and load scheduling trends with arbitrary release times are verified. A bus network (homogeneous single-level tree network) with arbitrary processor release times is considered. For the scheduling strategy, a sequential distribution with a staggered start scheduling scenario to minimize total processing finish time is assumed.

I. INTRODUCTION

The handling of large volumes of computational load on distributed and parallel networks is becoming a more and more challenging task. Over the past twenty years or so, a new mathematical tool, called divisible load scheduling theory, has been created to allow tractable performance analysis of systems incorporating communication and computation issues, as in parallel and distributed processing. Considerable attention was focused on minimizing the total processing time of the entire load in a large amount of studies.

In divisible load scheduling theory (DLT), it is assumed that computation and communication loads can be arbitrarily partitioned into infinitesimally small fractions and distributed among processors and links in the network. There are no precedence relations among the loads. A key feature of this divisible load scheduling theory [1]–[3] is that it uses a mathematical model. Based on these properties, a recursive deterministic mathematical formulation is often used to achieve optimal solutions. This is a powerful concept for the performance analysis of networks of processors and links. It has been studied for network topologies including linear daisy chain, bus network, and tree networks using a set of recursive equations [4]–[8]. Load distribution strategies for two and three dimensional mesh networks were presented [9]–[12]. In [13] the concept of time varying computational and communication environment was introduced. Conditions were found which determine which processors have to be utilized in order to minimize the finish time [14]. The computational complexity of DLT is considered in [15],[16]. DLT has been applied to various research areas such as computational biological [17] and image processing techniques [18].

Most of the previous studies adopting different load distribution strategies so far had assumed a network in which there are no time constraints. In other words, it was assumed that all of the processors in the system are available from the time instant at which the root processor starts the divisible load distribution. In practical scenarios, this is not always the case. We refer to the time instance at which a processor is available for processing the divisible load as its release time. Some recent studies [19],[20] present real-time scheduling strategies involving the release time of processors and processing deadlines based on the classic DLT analysis. Load scheduling adopting multi-installment techniques [21] in bus networks with release times is analyzed in [22]. Release time aware load scheduling with buffer size constraints is considered [23]. A heuristic strategy is proposed for scheduling in linear daisy chain networks with release times [24].

The analysis of these preceding studies are implicitly restricted to the scenario that all distributions of load fractions from a root processor can be completed before the earliest release time is reached. In this paper, we relax this restriction in that the distribution of the load fractions starts once the first child processor is released. We assume that processors in the network are not equipped with front end processors. Hence, computation occurs only after communication ends (i.e., staggered start). The load distribution occurs in a sequential manner. With the sequential distribution and staggered start scenario, many possible models with associated timing diagrams restricted with two constraints need to be considered to obtain an optimal solution in terms of minimum processing finish time.

The intriguing fact here is that the developed closed form solutions from the possible timing models with two constraints may give us multiple feasible solutions, but not an optimal solution. In other words, the solutions may not follow the timing models of analysis (leads to an infeasible solution). Put another way, optimal load allocations depend on release times but the timing of release times may not yield an optimal load allocation solution. Then, we may be caught in a vicious circle (the details appear in Section III). To resolve the circular reasoning problem, we propose an exhaustive search algorithm to search a timing model for an optimal solution.
Several important aspects regarding the scheduling behavior of the exhaustive search algorithm are demonstrated through a simulation study.

The remainder of this paper is organized as follows. Section II presents the types of notations and analytic background. Section III analyzes the timing diagrams based on the two timing constraints and demonstrates the exhaustive search algorithm. Simulation results showing the behavior of the exhaustive search algorithm are discussed in Section IV. The conclusion and open questions appear in Section V.

II. PROBLEM FORMULATION AND PRELIMINARY REMARKS

Consider the case where a bus network model (i.e., a homogeneous single-level tree network) consists of a root processor and \( n \) children processors. The children processors \( p_1, p_2, \ldots, p_n \) are connected to the root processor through a bus. The root processor divides the total load into \( n \) parts, namely, \( \alpha_1, \alpha_2, \ldots, \alpha_n \).

Then, the root processor distributes the load fractions to the children processors, sequentially (see Fig. 1).

In this paper, the following assumptions are initially made:

1. The root processor (BCU) does not participate in load computation.
2. The root processor (BCU) distributes load fractions to all the children processors in a sequential manner.
3. The children processors are not equipped with front-end processors. If a processor does not have a front-end processor, it can compute or communicate, but not do both at the same time (i.e., staggered start scenario). The load distribution from the root processor to a child processor can not take place before the child processor is released. This is in addition of the constraint that the load distribution from the root processor begins only after the first child processor is released.
4. Each of the children processors have a capability to start processing as soon as the processor becomes available at release time \( r_i \), where \( i = 1, 2, \ldots, n \).
5. Each of the children processors stops computing at the same time. Intuitively, this is because otherwise some processors would be idle while others were still busy. A rigorous proof for this argument in the case of linear, bus, and tree network parameters, is given in [1].
6. The root processor (BCU) knows the release times of the processors.
7. Without loss of generality, the order of the children processors follows the order in which their release times increase, that is \( r_1 \leq r_2 \leq \ldots \leq r_n \). In other words, the sequence of load distribution by the root processor (BCU) follows this order and let this order be \( p_1, p_2, \ldots, p_n \).
8. Compared to the size of the data, the time to report solutions back to the root processor (BCU) is negligible.

The following notation is used in this paper.

\( \alpha_i \) : The load fraction assigned to the \( i^{th} \) child processor (where \( i = 1, 2, \ldots, n \)).

\( w \) : The inverse computing speed of the children processors.

\( z \) : The inverse communication speed of the links.

\( T_{cp} \) : Computing intensity constant. The entire load can be processed on the \( i^{th} \) child processor in time \( wT_{cp} \).

\( T_{cm} \) : Communication intensity constant. The entire load can be transmitted over the \( i^{th} \) link in time \( zT_{cm} \).

\( T_{f,n} \) : The finish time. Time at which \( n \) children processors complete computation.

\( T_1 \) : The total time that elapses between the beginning of the process at \( t = 0 \) and the time when the \( i^{th} \) child processor completes its computation (where \( i = 1, 2, \ldots, n \)).

\( r_i \) : The release time. Time at which the \( i^{th} \) child processor becomes ready for receiving assigned load from the root processor (BCU) (where \( i = 1, 2, \ldots, n \)).

\( s_i \) : The start time. Time at which the \( i^{th} \) child processor actually starts receiving the assigned load from the root processor (BCU) (where \( i = 1, 2, \ldots, n \)). Here, \( s_i \geq r_i \).

\( d \) : The absolute deadline. Time by which the load processing must complete.

\( D_i \) : The relative deadline. Time difference between the absolute deadline and the release time: \( D_i = d - r_i \) (where \( i = 1, 2, \ldots, n \)).

\( \Gamma_i \) : The pair of \( i^{th} \) child processor and \( (i+1)^{th} \) child processor, \( (p_i, p_{i+1}) \) (where \( i = 1, 2, \ldots, n-1 \)).

III. A BUS NETWORK WITH ARBITRARY RELEASE TIMES

Consider a cluster with a bus network architecture where loads are distributed sequentially to the children processors as soon as the processor becomes available (i.e., sequential distribution). As mentioned earlier, processors are not equipped with front-end processors, so that processors start computing only after they receive the whole of the load assigned to them (i.e., staggered start), and initial communication starts at \( r_1 \) (i.e., \( s_1 = r_1 \)), the time when the whole load is assumed to be present at the root processor. Note that it is assumed that the release times of the processors, \( r_i \) (where \( i = 1, 2, \ldots, n \)) are given and fixed arbitrary constants. Accordingly, timing models with two rigid constraints are considered:

- \( r_{i+1} - s_i \leq \alpha_i zT_{cm} \) : The case that the release time of the \( (i+1)^{th} \) child processor, \( r_{i+1} \), occurs before the \( (i+1)^{th} \) load assignment from the root processor (BCU) starts or occurs during the middle of the communication time for the \( i^{th} \) load assignment. Given the sequential load distribution and staggered start scenario, this timing constraint has no impact on the load scheduling. Here, \( i = 1, 2, \ldots, n-1 \).
worth mentioning that the two constraints are assumed in that a total of \( n \) children processors \((p_1, \ldots, p_n)\) satisfies Constraint I (see Fig. 2). Without this assumption, the derivation of the optimal solutions and corresponding performance evaluations is not possible.

From Fig. 2, we can express the start times in terms of \( \alpha_i \) as

\[
s_i = r_1 + \left( \sum_{i=1}^{i-1} \alpha_i \right) T_{cm} \quad i = 2, 3, \ldots, n
\]

By substituting equation (2) into Constraint I, we observe that the feasible release times must satisfy the following condition:

\[
r_{i+1} \leq r_1 + \left( \sum_{i=1}^{i} \alpha_i \right) T_{cm} \quad i = 1, 2, \ldots, n - 1
\]

Intuitively, the inequality (3) is recognizable from Fig. 2. Here, the latest release time is implicitly \( r_n < d \) to sustain the initial assumption that a total of \( n \) children processors participate in the parallel load processing. In the same context, if the earliest release time is \( r_1 \geq d \), the feasibility of all release times is lost so that load parallel processing is no longer possible.

Using equation (1) and (2), we can obtain the optimal amount of \( \alpha_i \) as

\[
\alpha_i = \frac{q^{i-1} \alpha_1}{q^n - 1} \quad i = 2, 3, \ldots, n
\]

Here, \( q = \frac{w_{T_{cp}}}{z_{T_{cm}} + w_{T_{cp}}} \). The fractions of the total load should sum to one (normalization)

\[
\sum_{i=1}^{n} \alpha_i = 1
\]

By applying \( n \) equations, equation (4) and (5), one can obtain the optimal amount of load fraction, \( \alpha_1 \) as

\[
\alpha_1 = \frac{1}{\sum_{i=1}^{n} q^{i-1}} = \frac{1 - q}{1 - q^n}
\]

By substituting \( \alpha_1 \) into equation (4), the other \( n - 1 \) solutions for the optimal load fraction, \( \alpha_2, \ldots, \alpha_n \), can be obtained.

From equation (6), the minimum finish time is given by

\[
T_{f,n} = T_1 (= T_2 = T_3 = \ldots = T_n)
\]

\[
= r_1 + \alpha_1 z_{T_{cm}} + \alpha_1 w_{T_{cp}}
\]

\[
= r_1 + \frac{1 - q}{1 - q^n} (w_{T_{cp}} + z_{T_{cm}})
\]

The preceding derivation of the optimal solution has been already appeared in [25] introducing the load scheduling scenario with sequential distribution and staggered start. The derivational similarity occurs because the timing diagram of sequential distribution and staggered start does not vary according to the release time under Constraint I as we mentioned previously.

The minimum number of children processors, \( n^{\text{min}} \), that the
whole load needs at start time \( s_1 (= r_1) \) to meet the absolute deadline, \( d \) can be derived as follow

\[
T_{f,n} \leq r_1 + D_1
\]

This implies that

\[
r_1 + \frac{1 - q}{1 - q^n} (wT_{cp} + zT_{cm}) \leq r_1 + D_1
\]

since \( 1 - q^n > 0 \) from the fact that \( 0 < q < 1 \)

\[
1 - q^n \geq \frac{(1 - q) (wT_{cp} + zT_{cm})}{D_1}
\]

\[
q^n \leq 1 - \frac{zT_{cm}}{D_1}
\]

Thus, we have

\[
n \geq \frac{\ln \eta}{\ln q}
\]

where, \( \eta = 1 - \frac{zT_{cm}}{D_1} \). Here, \( 0 < \eta < 1 \) otherwise the distribution of the partitioned load can not meet the deadline and even can not have enough time for computing (see Fig. 2). Therefore, the minimum number of children processors that the load needs at time \( s_1 (= r_1) \) to meet its deadline is

\[
n^{\text{min}} \geq \left\lfloor \frac{\ln \eta}{\ln q} \right\rfloor
\]

**B. Constraint II :** \( r_{i+1} - s_i > \alpha_i zT_{cm} \)

In this subsection, we derive closed form solutions for an evaluation of the performance of a different case by following a similar procedure described in the earlier case. Note that the release time \( r_i \) is the same as the start time \( s_i \) for \( i = 1, 2, \ldots, n \) under Constraint II as shown in Fig. 3. Under this constraint, it is quite possible that some children processors have too large a release time so that the processors can not be assigned any fraction of the load from the root processor (BCU). The criteria for the feasible release times is also derived in this subsection. We assume that \( n \) children processors participating in distributed computing satisfy Constraint II initially. As in the previous subsection, load distribution equations in this subsection will not be reasonable without the assumption.

Rewriting equation (1) based on the timing diagram (see Fig. 3) yields

\[
\alpha_i = \alpha_1 + \frac{r_i - r_i}{wT_{cp} + zT_{cm}} \quad i = 2, 3, \ldots, n
\]

Also, the fractions of the total load should sum to one (normalization). This gives \( n \) linear equations, equation (5) and (13), with \( n \) unknowns. Then the normalization equation (5) becomes

\[
n\alpha_1 + \frac{(n - 1) r_1 - \sum_{i=2}^{n} r_i}{wT_{cp} + zT_{cm}} = 1
\]

From equation (14), one can obtain the optimal amount of load fraction, \( \alpha_1 \) as

\[
\alpha_1 = \frac{1}{n} \left( 1 - \frac{(n - 1) r_1 - \sum_{i=2}^{n} r_i}{wT_{cp} + zT_{cm}} \right)
\]

\[
= \frac{1}{n} \left( 1 + \frac{-nr_1 + \sum_{i=1}^{n} r_i}{wT_{cp} + zT_{cm}} \right)
\]

(15)

Substituting equation (15) into (13) yields

\[
\alpha_i = \frac{1}{n} \left( 1 + \frac{-nr_1 + \sum_{i=1}^{n} r_i}{wT_{cp} + zT_{cm}} \right) + \frac{r_1 - r_i}{wT_{cp} + zT_{cm}}
\]

\[
= \frac{1}{n} \left( 1 + \frac{-nr_1 + \sum_{i=1}^{n} r_i}{wT_{cp} + zT_{cm}} \right) \quad i = 2, 3, \ldots, n
\]

(16)

From equation (16), we can see that the \( \alpha_i \) depends on the prior values of release time and the number of the processors as compared to equation (4) given in the case of Constraint I. This property of the case of Constraint II opens the possibility that a condition for the normalized solution, \( 0 < \alpha_i \leq 1 \), is violated. To generate the condition for the feasible processor’s release times, we can use the necessary condition \( 0 < \alpha_i \leq 1 \) for \( i = 1, 2, \ldots, n \). This leads to the following important criterion for the feasible release times

\[
0 < \frac{1}{n} \left( 1 + \frac{-nr_1 + \sum_{i=1}^{n} r_i}{wT_{cp} + zT_{cm}} \right) \leq 1
\]

(17)

We can rewrite above inequality as follows

\[
C_1 \leq r_i < C_2
\]

(18)

where

\[
C_1 = \frac{wT_{cp} + zT_{cm} + \sum_{i=1}^{n} r_i}{n}
\]

\[
C_2 = \frac{wT_{cp} + zT_{cm} + \sum_{i=1}^{n} r_i}{n}
\]

(19)

In order to obtain the feasible release times, it is required to apply the condition (18) iteratively by reducing the number of processors, \( n \) until all release times satisfy the necessary
condition (18). Note that for the optimal solutions, the release times should admit Constraint II, which is the sufficient condition for the optimal solutions. As we mentioned in the case of Constraint I, the latest release time is implicitly less than the absolute deadline (i.e., $r_n < d$) to sustain the initial assumption that a total of $n$ children processors participate in the parallel load processing. When the earliest release time passes over the absolute deadline (i.e., $r_1 \geq d$), the $n$ children processors then lose feasibility in their ability to have a feasible solution.

From equation (15), the minimum finish time is given by

$$T_{f,n} = T_1 (= T_2 = T_3 = \ldots = T_n) = r_1 + \alpha_1 z T_{cm} + \alpha_1 w T_{cp}$$

$$= r_1 + \frac{1}{n} \left(1 + \frac{\sum_{i=1}^{n} r_i}{w T_{cp} + z T_{cm}}\right) (w T_{cp} + z T_{cm})$$

$$= \frac{1}{n} \left(w T_{cp} + z T_{cm} + \sum_{i=1}^{n} r_i\right)$$

$$= \frac{1}{n} \left(w T_{cp} + z T_{cm} + \mu\right)$$

(20)

Here, let $\mu = \frac{1}{n} \sum_{i=1}^{n} r_i$, which is an average value of $n$ release time points.

Using the similar method shown in the previous subsection, the minimum number of children processors, $n^{\text{min}}$, that the load needs at start time $s_1 (= r_1)$ to meet the absolute deadline, $d$ can be derived as follows

$$T_{f,n} \leq r_1 + D_1$$

(21)

This implies that

$$\frac{1}{n} (w T_{cp} + z T_{cm}) + \mu \leq r_1 + D_1$$

(22)

Thus, we have

$$n \geq \frac{w T_{cp} + z T_{cm}}{r_1 + D_1 - \mu}$$

(23)

Here, it is obvious that $r_1 + D_1$, which is the absolute deadline, $d$ is larger than $\mu$. This is because all release times $r_i$ of the eligible children processors are surely less than $d$.

Therefore, the minimum number of children processors that the load needs at time $s_1 (= r_1)$ to meet its deadline is

$$n^{\text{min}} = \left\lceil \frac{w T_{cp} + z T_{cm}}{r_1 + D_1 - \mu} \right\rceil$$

(24)

C. Exhaustive search algorithm

One may recognize that a schedule will not necessarily involve only one of the two constraints (i.e., Constraint I and Constraint II). An actual scheduling may consist of a mixture of both constraints. In this subsection, we focus on scheduling divisible loads with arbitrary release time with no prior knowledge of constraints. In case that $n$ processors get involved in the distributed computing, $2^{n-1}$ possible models should be taken into account. Fig. 4 shows an example of a timing diagram among $2^{n-1}$ possible cases which the exhaustive search algorithm takes into account. Note that the diagram shows one of the possible cases where the load distribution follows the decreasing order of the release time.

From the Fig. 4, one can see that the example diagram is composed of $\Gamma_1(I) \ldots \Gamma_{k-1}(I)$, $\Gamma_k(II) \ldots \Gamma_{k+m-1}(II)$, and $\Gamma_{k+m}(I) \ldots \Gamma_{n-1}(I)$. Here, we denote the $i$th pair of processors, $\Gamma_i$, satisfying Constraint I and Constraint II as $\Gamma_i(I)$ and $\Gamma_i(II)$ respectively.

From the timing diagram, Fig. 4, one can write the following set of deterministic equations in the system

$$T_1 = r_1 + \alpha_1 (z T_{cm} + w T_{cp})$$

$$T_2 = s_2 + \alpha_2 (z T_{cm} + w T_{cp})$$

$$\vdots$$

$$T_k = s_k + \alpha_k (z T_{cm} + w T_{cp})$$

$$T_{k+1} = r_{k+1} + \alpha_{k+1} (z T_{cm} + w T_{cp})$$

$$T_{k+2} = r_{k+2} + \alpha_{k+2} (z T_{cm} + w T_{cp})$$

$$\vdots$$

$$T_{k+m} = r_{k+m} + \alpha_{k+m} (z T_{cm} + w T_{cp})$$

$$T_{k+m+1} = s_{k+m+1} + \alpha_{k+m+1} (z T_{cm} + w T_{cp})$$

$$\vdots$$

$$T_n = s_n + \alpha_n (z T_{cm} + w T_{cp})$$

(25)

In order to achieve the optimal solutions for the minimum finish time, all processors stop processing at the same time as we mentioned earlier (i.e., $T_1 = T_2 = T_3 = \ldots = T_n$).

Based on the above set of equations, one can write the
The pattern of the series of equations under assumption of either Constraint I or Constraint II can be also seen as shown in Section III-A and Section III-B. As we mentioned earlier, the fractions of the total load should sum to one

\[ \sum_{i=1}^{n} \alpha_i = 1 \]  

(27)

Based on the above total of \( n \) equations (i.e., equation (26) and (27)), we can obtain the solutions for the fraction of load to be assigned to \( n \) processors. By using this analytical method, we can obtain solutions for all \( 2^{n-1} \) possible cases. Then, we can find an optimal case based on the obtained values of \( \alpha_i \)'s. The sets of deterministic equations obtained from \( 2^{n-1} \) possible cases are solved via linear programming (LP). In terms of the required computing power, linear programming is a quite useful method for drastically improved computing speed to find the optimal solution. Assuming that the simplex method is used for solving LP, its worst-case computational complexity is exponential (i.e., \( O(2^n) \) for some constant \( c > 1 \).

For the linear programming, equation (26) can be expressed in canonical form as

\[ A \cdot \alpha = b \]  

(28)

\( \alpha \) represents the \( n \times 1 \) vector of solution variables to be determined. From equation (26), the \( n \times n \) known coefficient matrix, \( A \) and the \( n \times 1 \) vector of known coefficient, \( b \) can be written as

\[
\begin{pmatrix}
\alpha_2 \\
\alpha_3 \\
\vdots \\
\alpha_{k+1} \\
\vdots \\
\alpha_{n-1} \\
\alpha_n
\end{pmatrix} = \begin{pmatrix}
q\alpha_1 \\
q\alpha_2 \\
\vdots \\
q\alpha_{k-1} \\
\alpha_1 + \frac{r_1 - r_{k+1}}{wT_{cp} + zT_{cm}} \\
\alpha_1 + \frac{r_1 - r_{k+2}}{wT_{cp} + zT_{cm}} \\
\vdots \\
\alpha_1 + \frac{r_1 - r_{k+m}}{wT_{cp} + zT_{cm}} \\
q\alpha_{n-2} \\
q\alpha_{n-1}
\end{pmatrix}
\]  

(26)

\[
\begin{bmatrix}
-\gamma & 1 & 0 & \cdots & 0 \\
0 & -\gamma & 1 & 0 & \cdots \\
0 & 0 & -\gamma & 1 & 0 & \cdots \\
-1 & 0 & \cdots & 0 & 1 & 0 & \cdots \\
-1 & 0 & \cdots & 0 & 1 & 0 & \cdots \\
0 & \cdots & -\gamma & 1 & 0 & \cdots \\
0 & \cdots & 0 & -\gamma & 1 \\
0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & -\gamma \\
1 & 1 & 1 & 1 & \cdots & 1 & 1
\end{bmatrix}
\]

\( b = \begin{pmatrix}
0 \\
\vdots \\
0 \\
-\gamma \\
0 \\
\vdots \\
0 \\
0 \\
1 \\
1 \\
0 \cdots 0 \cdots 1 \cdots 1 \\
0 \cdots 0 \cdots 1 \cdots 1
\end{pmatrix}^T
\]

Similarly, other candidates of the timing model can be

**Algorithm 1 Exhaustive search algorithm**

1. procedure EXHAUSTIVE SEARCH\( (n, r) \)
2. \( n \leftarrow \) The total # of candidate processors
3. \( r \leftarrow \{ r_1, r_2, \ldots , r_n \} \) where \( r_1 \leq r_2 \leq \cdots \leq r_n \)
4. while \( n \geq 1 \) do
5. for All \( 2^{n-1} \) possible timing models do
6. Obtain feasible solutions for \( \alpha \) from LP
7. if An optimal timing model with an optimal solution exists then
8. return \( \alpha \) \hspace{1em} // Optimal solution
9. else
10. Eliminate the candidacy of processor with \( r_n \)
11. \( r \leftarrow r \setminus r_n \)
12. \( n \leftarrow n - 1 \)
13. if \( n = 0 \) then
14. return Infeasible scenario! \hspace{1em} // No candidate processor
15. end if
16. Break for
17. end if
18. end for
19. end while
20. end procedure

The linear programming problem can be defined as the problem of minimizing the objective function, the finish time, \( T_{f,n} \) subject to linear constraints. Here, the objective function is

\[ \text{Min}(T_{f,n}) = \text{Min}(r_1 + \alpha_1(zT_{cm} + wT_{cp})) \]  

(29)

The constraints are \( n \) equalities (equation (26) and (27)) and the bounded nonnegative constraints

\[ 0 \leq \alpha_i \leq 1 \quad i = 1, 2, \ldots , n \]  

(30)
From Fig. 4, additional \( n - 1 \) inequality constraints for the release times

\[
  r_i \leq T_{f,n} - \alpha_i(zT_{cm} + wT_{cp}) \quad i = 2, 3, \ldots, n
\]

is derived in order to check the feasibility of the release times for the case that mixture of Constraint I and Constraint II happens. Using equation (29), the inequality constraints (31) can be reformulated as

\[
  \alpha_i - \alpha_1 \leq \frac{r_1 - r_i}{zT_{cm} + wT_{cp}} \quad i = 2, 3, \ldots, n
\]

Another inequality constraint can be obtained from the time consumption for the total load distribution from BCU to all participating processors as

\[
  r_1 + zT_{cm} \sum_{i=1}^{n} \alpha_i \leq T_{f,n} - \alpha_n wT_{cp}
\]

Using equation (29), the inequality constraints (33) can be reformulated as

\[
  \alpha_n wT_{cp} - \alpha_1(zT_{cm} + wT_{cp}) \leq -zT_{cm}
\]

Based on the \( n \) equality constraints (i.e., (26) and (27)) and inequality constraints (i.e., (32) and (34)), the feasibility of the solutions based on the candidates of the timing model can be checked. One critical point here is that the feasible solutions obtained by LP based on the timing model may not satisfy the timing model, ironically. This is because the issue of the release time’s feasibility studied in the section of Constraint II, Section III-B implicitly arises. To deal with this issue, an exhaustive search algorithm (see Pseudo code in Algorithm 1) to obtain an optimal timing diagram for a certain network with arbitrary release time is proposed. To clarify the feasible release time issue, it is required to check if the feasible solution obtained by LP satisfies the timing model for its own solution. If there exists no feasible solutions satisfying its own timing model, the algorithm concludes that there is no optimal timing model for the given release times. Then, the candidacy of the \( n^{th} \) processor with the largest candidate release time, \( r_n \) is rejected. The largest release time, \( r_n \), is regarded as an infeasible release time. Then, the new loop of the algorithm with \( n - 1 \) candidate processors without \( n^{th} \) processor will be considered. The algorithm continues running until finding an unique optimal solution. The obtained unique timing model is the optimized model for the minimum finish time with the generalized scenario containing both cases of Constraint I and Constraint II. Note that the number of candidate processors, \( n \) is chosen implicitly based on the \( n \) processors’ load finish time within the absolute deadline (i.e., \( T_{f,n} \leq d \)). Thus, running time of the LP based exhaustive search algorithm considering \( 2^{n-1} \) possible timing models increase exponentially in the worst-case. Naturally this is not a scalable solution—something that awaits further research. One critical drawback of the exhaustive search algorithm is that the algorithm requires high-computational power to consider the possible cases, especially as the number of processors increases and the number of investigated loops of exponential computational complexity increases. Our contribution is that the analysis of the generalized case will provide the flexibility for studying various scenarios with arbitrary release times.

![Fig. 5. Average number of utilized processors against average release time, \( r \) and inverse computing speed, \( wT_{cp} \), \( zT_{cm}=0.2 \)](image)

IV. PERFORMANCE EVALUATION

Now we demonstrate how the exhaustive search algorithm works by means of an illustrative example. Consider a system with 12 children processors, with randomly generated release times by the exponential distribution, \( r_1, r_2, \ldots, r_{12} \) as 0.52809, 0.98233, 1.0046, 1.5934, 2.0759, 2.1932, 3.1361, 3.5003, 5.3652, 5.3765, 5.8674, and 6.1941 sec respectively and inverse speed parameters \( wT_{cp} = 1.2 \) sec/load and \( zT_{cm} = 0.8 \) sec/load. The sets of deterministic equations obtained from \( 2^{11} \) possible timing models are solved via linear programming (MATLAB linear programming function, linprog)). We have the following information generated by the exhaustive search algorithm.

The optimal solutions for normalized fraction of loads are achieved as \( \alpha_1 = 0.52438, \alpha_2 = 0.29726, \) and \( \alpha_3 = 0.17836 \).

From the above results, we observe that \( 12^{th}, 11^{th}, 10^{th}, 9^{th}, 8^{th}, 7^{th}, 6^{th}, 5^{th} \) and \( 4^{th} \) processors are removed from the distributed computing candidates at the \( 1^{st}, 2^{nd}, 3^{rd}, 4^{th}, 5^{th}, 6^{th}, 7^{th}, 8^{th}, \) and \( 9^{th} \) iteration of the algorithm respectively, and the fraction of loads load shown above are allocated to the remaining 3 children processors with release times, \( r_1 = 0.52809 \) sec, \( r_2 = 0.98233 \) sec, and \( r_3 = 1.0046 \) sec respectively. Further, it is shown that the optimal timing diagram for the \( 1^{st} \) pair of processors, \( \Gamma_1 \), follows Constraint II (i.e., \( \Gamma_1(II) \)) and for the \( 2^{nd} \) pair of processors, \( \Gamma_2 \), follows Constraint I (i.e., \( \Gamma_2(I) \)). We can confirm the optimality of the timing diagram via checking the Constraints as follows

\[
  s_1 = r_1 = 0.52809 \text{ sec} \\
  \Gamma_1(II) : r_2 - s_1 = 0.4542\text{sec} > \alpha_1 zT_{cm} (= 0.4195\text{sec})
\]

is satisfied. Thus, \( s_2 = r_2 \).

\[
  \Gamma_2(I) : r_3 - s_2 = 0.0223\text{sec} \leq \alpha_2 zT_{cm} (= 0.2378\text{sec})
\]

is satisfied. Thus, \( s_3 = r_2 + \alpha_2 zT_{cm} (= 1.2201\text{sec}) \)

The minimized finish time, \( T_{f,3} = 1.5769 \) sec. Based on this minimized finish time, the fact that the 9 processors having release times larger than the minimized finish time lose their candidacy can be reverified.

To investigate the behavior of the algorithm, we assume 7 candidate children processors are initially given with 7 random release times (i.e., \( r_1, r_2, \ldots, r_7 \)) generated from the
exponential distribution. The sets of deterministic equations obtained from $2^6$ possible timing models are solved via the MATLAB linear programming function, \textit{linprog}(). Simulation is performed 100 times with network variables (e.g., inverse speed parameters and release times). For generality, network speed parameters are set as the communication speed is faster than the computing speed (i.e., $w_{TCP} \geq z_{TCP}$). 

Fig. 5 describes the average number of processors actually utilized in the processing against inverse computing speed $w_{TCP}$ and release times $r$. Fig. 6 describes the average finish time performance against inverse computing speed $w_{TCP}$ and release times $r$. From Fig. 5, we can observe that the average number of processors participating in the processing decreases as the average release time increases. For the exhaustive search algorithm, the number of candidate processors decreases by one per iteration in the case that no optimal timing diagram is found. This corresponds to the rejection of the candidacy of a child processor with the largest release time as we mentioned before. It can be expected that the possibility that the release time is infeasible increases as the average release time increases. Thus, intuitively the number of rejected processors increases as the average release time increases. Interestingly, from Fig. 5, we can observe that the average number of processors participating in the processing slightly increases as computing speed decreases (i.e., as $w_{TCP}$ increases). This phenomena implies that the relative increment in the minimum finish time according to a slower computing speed (see Fig. 6) contributes to degradation of the rejection rate of processor candidates. A sharper variation in the number of utilized processors against average release times than against inverse computing speed also corresponds to the variation of average minimum finish time against same variables as shown in Fig. 6. As the time until the candidate processors are available to receive their own divisible load takes longer, the average time to finish a whole load increases. Also, the decreasing number of participating children processors as the release time increases contributes to the longer time spend to end the processing of the whole load. 

Similarly, Fig. 7 and Fig. 8 describes the average number of processors actually utilized in the processing and the average finish time performance against inverse communication speed $w_{TCP}$ and release times $r$, respectively. Similar to the trend against inverse computing speed, slower communication speed (i.e., larger value of $z_{TCP}$) contributes to longer minimum finish time so that processors with later release times tends to be utilized, not rejected. Faster communication speed and faster release time contribute to smaller average minimum finish time as shown in Fig. 8.

With 7 candidate children processors (i.e., 6 processor pairs, $\Gamma_1, \Gamma_2, \ldots, \Gamma_6$), the occurrence trend of the processor pair satisfying the Constraint I, $\Gamma(I)$ and satisfying the Constraint II, $\Gamma(II)$ is illustrated in Fig. 9, Fig. 10, Fig. 11, Fig. 12, and Fig. 13 with different values of speed parameters. The values on the upper side and lower side of basis zero show the normalized average occurrence frequency of the $\Gamma(I)$ and $\Gamma(II)$ respectively over the z-axis. It can be observed that children processor pairs having smaller release time
are likely to satisfy the Constraint I. That is, the lower the index of the children processor pair, the more $\Gamma(I)$ dominates. The lower index of the children processor pair implies the smaller release time as the distribution order of processor (i.e., $r_1 \leq r_2 \leq \ldots \leq r_n$). This phenomena can be interpreted as the probability that a child processor waits until its own optimal amount of load is distributed from the root processor even though the child processor is released from its prior load processing (i.e., the probability of $\Gamma(I)$ increases as the release times of the children processors become earlier). By similar intuition, it seems reasonable that the network with the higher indexed processor pairs with larger release times experiences the delay shown in the case of the Constraint II. This is because the probability that the root processor waits until processors are available (released) to distribute the optimally divided loads to the children processors (i.e., the probability of $\Gamma(II)$) increases as the release times of the children processors become larger. Dominant zero and near zero average frequency appearing in the higher index of the children processor pair as release time increases seems reasonable in that the children processors’ candidacy is eliminated due to their releases times occurring beyond the minimum finish time. This trend can be intuitively approximated through the average minimized finish time information shown in Fig. 6 and Fig. 8. Based on the information for the trend of $\Gamma(I)$ and $\Gamma(II)$ in Fig. 9 with speed parameters, $wT_{cp} = 1.2$, $zT_{cm} = 0.8$, faster communication speed (i.e., $zT_{cm} = 0.2$) is applied in Fig. 10. Higher frequency of $\Gamma(II)$ appears in the lower index of the children processor pairs as compared to Fig. 9. This is because decreased communication latency tends to relatively increase the probability that the root processor waits until processors are released. The decreased communication time contributes to an overall decrement in the minimum finish time so that more processors tends to lose their candidacy due to their overdue release times. Thus, zero average frequency appears in a relatively larger range as compared to Fig. 9. Due to a similar reason, overall average frequency appears lower than the case of Fig. 9.

Now, based on the data for the trend of $\Gamma(I)$ and $\Gamma(II)$ in Fig. 9 with speed parameters, $wT_{cp} = 1.2$, $zT_{cm} = 0.8$, slower communication speed (i.e., $zT_{cm} = 1.2$) is applied in Fig. 11. As we can expect, relatively higher frequency
of $\Gamma(I)$ can be observed in the lower index of the children processor pairs than Fig. 9. The increased communication latency due to the slower communication speed contributes to increase the probability that each of the children processor waits until its own optimal load allocation is started (i.e., $\Gamma(I)$). Also, as compared to Fig. 9, the overall increment in frequency of $\Gamma(I)$ can be seen. This is because the increment in the minimum finish time due to the slower communication speed tends to take into account processors with relatively larger release times.

Similarly, in Fig. 12 and Fig. 13, the trend of $\Gamma(I)$ and $\Gamma(II)$ for a relatively faster computing speed (i.e., $wT_{cp} = 0.8$) and a relatively slower computing speed (i.e., $wT_{cp} = 2.0$) than the case of Fig. 9 is depicted respectively. In Fig. 12, it can be seen that the frequency of $\Gamma(I)$ is dominant as compared to Fig. 9. This phenomena can be explained in terms of the amount of optimal divisible load allocation. Due to the faster computing speed, the amount of load to be assigned to children processors tends to be more dense for the earlier processing units. In other words, the amount of optimal load fraction assigned to earlier processing units increases as to minimize the finish time as the computing speed increases. Thus, the communication time relatively increases so that it can have the similar effect of decreasing communication speed shown in Fig. 11. On the contrary, the slower computing time causes a load balanced trend in terms of optimal load allocation so that the amount of load to be assigned to the earlier processing units relatively decreases. Thus, the communication time relatively decreases so that the frequency of $\Gamma(II)$ increases similar to the case of increasing communication speed in Fig. 10.

From the simulation results, we can observe that the exhaustive search algorithm behaves in a reasonable manner over the generalized models which are a mixture of Constraint I and Constraint II.

V. CONCLUSION AND OPEN QUESTIONS

In this paper, the divisible load distribution problem for a bus network adopting sequential distribution and a staggered start strategy is examined. An analysis was provided based on the real-time assumption that processors in the network have arbitrary release times. Unlike the previous literature, where a single scenario of the divisible load distribution is taken into account, we can see that the scenario here involving arbitrary release times can be restricted in two ways. The performance analysis of the network following each of the two constraints exclusively is derived. Based on the analysis of the case of the two constraints, an analysis for one of the possible models, mixed two constraints, in a realistic situation is presented. Based on the analysis of a realistic case, we proposed an exhaustive search algorithm for obtaining an optimal solution of the scheduling of divisible loads.

The approach demonstrated in this paper can initiate a means for solving the divisible load scheduling problem under actual circumstances with arbitrary release times. One of the most intriguing problems is to develop a computationally efficient heuristic algorithm, which is unlike the one here and scalable,
to deal with the exponentially growing possible timing models. As for more rigorous and comprehensive analytic results, the investigation of the following issues are also expected to extend our study:

- An analytical model based on the simultaneous distribution and simultaneous start strategy (with front-end processor).
- The performance analysis of the heterogeneous network parameters.
- On-line processors that leave and enter the parallel processing network environment.

ACKNOWLEDGMENT

The authors would like to thank Dr. Jason Hung for useful conversations on the subject of the paper and the anonymous referees for their valuable and constructive comments. Dr. Thomas Robertazzi wishes to acknowledge the support of DOE grant DE-SC0003361 on his contribution.

REFERENCES


