UNIVERSITY AT STONY BROOK

CEAS Technical Report 757

Cost Efficient Processor Arrangement in Single Level Tree Networks

S. Charcranoon, T.G. Robertazzi and S. Luryi

March 30, 1998

Cost Efficient Processor Arrangement in Single Level Tree Networks

Saravut Charcranoon, Student Member, IEEE

Thomas G. Robertazzi, Senior Member, IEEE

Serge Luryi, Fellow, IEEE

Department of Electrical Engineering,

University at Stony Brook,

Stony Brook, NY 11794

Phone: (516) 632-8400

Fax: (516) 632-8494

March 8, 1998

Abstract

A study of the optimal placement of processors in a single level tree network processing a divisible load is presented. A set of conditions indicating when the current processor arrangement profile can be improved in terms of total processing cost is obtained. A heuristic algorithm based on a local search is developed from these conditions. The experimental results show an impressive quality of the solution both in effectiveness and in proximity of suboptimal solutions to an optimal solution. The efficiency of the algorithm in terms of the average number of processor arrangement profiles to be searched before a final solution is reached can be bounded by a low order polynomial in the number of children processors as $O(N^{1.6})$ for up to 19 processors. Several special cases of the model are considered. This work demonstrates the feasibility of cost accounting for future computer utilities.

Keywords Single-level tree network, divisible load sharing, processor arrangement, heuristic algorithm, local search, cost, computer utility.

1 Introduction

There is growing interest in networked distributed computing systems. This is evident from many new system technology developments as well as experiments on such systems. Among these are the widespread use of the Internet and intranets, the new development of portable and interoperable network software such as JAVA, CORBA, and the successful solution of cryptographic problems via distributed computing system. These factors naturally lead to the introduction of "computer utilities" in the near future. This concept introduces "resource utilization cost" into the problem of control and management in distributed computing systems. By including the resource utilization cost into such problems, the new problem setting requires not only system performance but also the corresponding incurred cost to be considered. One of the fundamental and interesting questions is how to arrange processors in a network topology such that the "total processing cost" incurred from utilizing network resources is minimized while the quality of service is at an accepatable level.

In scheduling in network-wide environments both computational time and communication time need to be considered together. Parallelism in the com-

putational load is another important aspect to be taken into account in order to realize the full capability of a distributed computing system. There is a significant class of loads which explicitly possess data parallelism. This is the class of divisible loads where load can be divided into arbitrary fractions and each fraction of load can be processed in parallel by different processors in the network. Example of divisible loads include the processing of large linear data files as in image processing, signal processing, massive computational or experimental data processing, some massive simulation programs, and cryptography.

So far there have been a number of works on divisible load sharing [1, 2, 3, 4, 5, 6]. All these works attempt to minimize the finish time based on the premise that every active processor stop computing at the same time. However there are only a few works [7, 8] which study the resource utilization problem in the context of divisible load theory. In [8], the authors present a study of optimizing computing cost in a bus network. Therein they provide a criteria to determine a sequence of load distribution that yields the minimum total computing cost. In [7], the authors generalize the previous work [8] to include both link transmission cost and processor cost in a single-level tree network. They find that it is not possible to derive a simple optimal

condition. Instead a set of conditions indicating when a sequence can be improved in total processing cost was obtained. In both works, however, it was a *logical interchange* of an adjacent link-processor pairs that was used as the primitive operation to find the best sequence of load distribution in terms of the total processing cost. In this paper, on the other hand, it is an *architectural rearrangement*, a reformation of processor-link pairs in a single level tree network, through a *physical interchange* of pairs of processors that is used as the basic operation to achieve a minimum total processing cost in a single-level tree network with both link cost and processor cost. In a processor arrangement problem the sequence of load distribution to each link is fixed throughout the course of processor arrangement procedure. The problem is which of N processors to connect to which of N links in a one to one manner.

In this work there are two objective functions to be optimized: the finish time and the total processing cost. It is well known that there are several approaches to solve multiple-objective functions optimization problems. The approach taken here is to find the minimal cost processor arrangement profile given that for any profile, finish time is minimized using the methodology of [3]. That is, for each arrangement profile considered, load is allocated so

that all processors stop computing at the same time instant and finish time is thus minimized for each specific profile. While other approaches are certainly possibly, we believe that the proposed approach is a natural one.

The goal of this paper is to present an analysis of processor arrangement in a single-level tree network where the processors are equipped with front-end processors. We optimize the network arrangement in terms of cost and finish time using adjacent processor pairwise swapping and a load distribution principle, both of which are described below. This analysis is developed to derive the necessary conditions for an improvement in total processing cost. A heuristic algorithm to search for a cost-efficient processor arrangement in an effective and efficient maner is then developed. It is based upon a local search with a multi-level neighborhood structure and multiple initial solutions as its central parts. The corresponding performance is also assessed and discussed.

The paper is organized as follows. The model and concept are presented in section 2. In section 3, processor arrangement and cost optimization are discussed. Cost efficient processor arrangements and the necessary cost improvement conditions in a general single-level tree network are developed in section 4. Section 5 presents the optimal conditions to obtain the minimum

total processing cost processor arrangements in a bus and related networks. Remarks on the analysis of the previous two sections are given in section 6. The heuristic cost efficient processor arrangement algorithm and its performance evaluation are developed and discussed in section 7 and 8. Finally the conclusion appears in section 9.

2 Models and Notations

2.1 Models Descriptions

In this paper, a single-level tree network where the root processor is equipped with a front-end processor is considered. A single-level tree network with $(N+1)$ processors and (N) links is shown in Figure 1. All the processors are connected to the root processor, p_0, via communication links. That is the children processors p_1,..., p_N are connected to the root processor p_0 via links l_1, l_2,..., l_N. Associated with the links and processors are the associated cost coefficients c_1^l, c_2^l,..., c_N^l and c_0^p, c_1^p, c_2^p,..., c_N^p, respectively as depicted in Figure 2. The root processor, assumed to be the only processor at which the load arrives, partitions the total processing load into $(N+1)$ fractions,

keeps its own fraction α_0, and distributes the other fractions α_1, α_2,..., α_N to the children processors p_1, p_2,..., p_N respectively and sequentially. Each processor begins computing immediately after receiving its assigned fraction of load and continues without any interruption until all of its assigned load fraction has been processed. We do not consider multi-installment strategies as in [3].

For clarity, a sequence of load distribution from the root processor to the children processors in a single-level tree network is represented by an ordered set as below,

$$\pi = \{p_0, (l_1, p_1), (l_2, p_2), ..., (l_j, p_j), ..., (l_N, p_N)\}$$

where (l_j, p_j) represents the j^{th} processor (p_j) connected to the root processor (p_0) via the j^{th} link (l_j).

This ordered set represents a sequence in which the root processor distributes load to the children processors (from p_0 to p_1,p_2,...,p_N). Without loss of generality, it is assumed that a sequence of load distribution is from left to right.

2.2 Notations

Let

α_i: The load fraction assigned to the i^{th} link-processor pair.

w_i: The inverse of the computing speed of the i^{th} processor.

z_i: The inverse of the link speed of the i^{th} link in the single level tree network.

T_{cp}: Time taken to process an entire load by a standard processor, $w_{standard} = 1$.

T_{cm}: Time taken to communicate an entire load by a standard link, $z_{standard} = 1$.

T_f: The finish time of an entire load, assuming that the load is delivered to the origination processor at time zero. Here the "finish time " is the time when the last processor ceases computation.

2.3 Optimal Finish Time Load Distribution

An equal division of load among processors does not in general give a minimum processing finish time even in a homogeneous network [3]. Instead, it is intuitive that to minimize the processing finish time the cost efficient load distribution should be such that all processors finish computing at the same time. In other words, cost efficient load distribution should not allow

any processor to finish its computation and then remain idle while other processors are still busy with their computations. Otherwise the processing finish time could be reduced by transferring some fractions of load from the busy processors to the idle processors. Formal proofs of this argument in the case of linear, bus, and tree networks appear in [3]. However, under certain sets of network parameters, in order to minimize the processing finish time, it is not necessary that all processors have to be utilized. In [3] conditions are found which determine which processors should be used to process the arriving load in the case of a single-level tree network. Still, the processors with non-zero assigned load have to finish computing at the same time. In this paper, it is assumed that all processors in the network are utilized.

Hence, throughout this paper, all processors are required to participate in load processing and they stop computing at the same time instant. Based on this assumption, the recursive equations for a single-level tree network are derived below. This is done by equating the finish times of all of the processors.

2.4 Fundamental Recusive Equations and Timing Diagram

The timing diagram of a single level tree network is given by Figure 3.

From the timing diagram, one can derive fundamental recursive equation as

$$\alpha_i w_i T_{cp} = \alpha_{i+1} z_{i+1} T_{cm} + \alpha_{i+1} w_{i+1} T_{cp} \quad , i = 0, ..., N-1 \tag{1}$$

They can be written in another form as,

$$\alpha_{i+1} = k_i \alpha_i = (\prod_{j=0}^{i} k_j)\alpha_0 \quad i = 0, ..., N-1 \tag{2}$$

where

$$k_i = \frac{\alpha_{i+1}}{\alpha_i}$$

$$= \frac{w_i T_{cp}}{z_{i+1} T_{cm} + w_{i+1} T_{cp}} \qquad i = 0, ..., N-1 \tag{3}$$

$$\tag{4}$$

Clearly, from Eq.(1) and (2), there are N equations and (N+1) unknowns. An additional equation, the normalization equation, is needed to solve this system of equations. The normalization equation is given as,

$$\alpha_0 + \alpha_1 + ... + \alpha_N = 1 \tag{5}$$

$$\sum_{i=0}^{N} \alpha_i = 1 \tag{6}$$

With the normalization equation, one can then resolve the recursive equations (1) to obtain the closed-form expression of α_0, the fraction of load of the root processor. Once α_0 is known, the other processor load fractions can be obtained by substituting α_0 into Eq.(2) and solving them recursively as shown below.

$$\begin{aligned}
\alpha_0 &= \left[1 + \sum_{i=1}^{N} \left[\prod_{j=0}^{i-1} k_j\right]\right]^{-1} && (7)\\
&= [1 + k_0 + k_0k_1 + ... + k_0k_1 \cdots k_{N-1}]^{-1} && (8)\\
&= \left[1 + \frac{w_0T_{cp}}{(z_1T_{cm} + w_1T_{cp})} + ... + \frac{\prod_{i=0}^{N-1}(w_iT_{cp})}{\prod_{i=1}^{N}(z_iT_{cm} + w_iT_{cp})}\right]^{-1} && (9)\\
&= \frac{1}{D}\prod_{i=1}^{N}(z_iT_{cm} + w_iT_{cp}) && (10)\\
\alpha_1 &= k_0\alpha_0 \\
&= \frac{w_0T_{cp}}{(z_1T_{cm} + w_1T_{cp})}\frac{1}{D}\prod_{i=1}^{N}(z_iT_{cm} + w_iT_{cp}) \\
&= \frac{1}{D}(w_0T_{cp})\prod_{i=2}^{N}(z_iT_{cm} + w_iT_{cp}) && (11)\\
\alpha_2 &= k_1\alpha_1 \\
&= \frac{w_1T_{cp}}{(z_2T_{cm} + w_2T_{cp})}(w_0T_{cp})\frac{1}{D}\prod_{i=2}^{N}(z_iT_{cm} + w_iT_{cp}) \\
&= \frac{1}{D}(w_0T_{cp})(w_1T_{cp})\prod_{i=3}^{N}(z_iT_{cm} + w_iT_{cp})
\end{aligned}$$

$$\vdots$$

$$\begin{aligned}
\alpha_n &= k_{n-1}\alpha_{n-1} \\
&= \frac{1}{D}\prod_{i=0}^{n-1}(w_i T_{cp}) \prod_{i=n+1}^{N}(z_i T_{cm} + w_i T_{cp})
\end{aligned}$$

$$\vdots$$

$$\begin{aligned}
\alpha_N &= k_{N-1}\alpha_{N-1} \\
&= \frac{1}{D}\prod_{i=0}^{N-1}(w_i T_{cp})
\end{aligned}$$

where

$$\begin{aligned}
D &= \prod_{i=1}^{N}(z_i T_{cm} + w_i T_{cp}) \\
&\quad + \sum_{n=1}^{N}\left(\prod_{i=0}^{n-1}(w_i T_{cp}) \prod_{i=n+1}^{N}(z_i T_{cm} + w_i T_{cp})\right)
\end{aligned} \tag{12}$$

where:

$$\begin{aligned}
w_0 T_{cp} &= \prod_{i=0}^{0}(w_i T_{cp}) \\
1 &= \prod_{i=N+1}^{N}(z_i T_{cm} + w_i T_{cp})
\end{aligned}$$

3 Processor Arrangement and Cost Optimization

3.1 Processor Arrangement

Processor arrangement refers to the connection between links and processors in a single-level tree network. Processor arrangement in general involves a permutation of the order of processors to receive fractions of load from the root processor while maintaining the original arrangement of links in a network throughout the course of the processor arrangement. In terms of an ordered set representation of a single-level tree network:

$$\pi = \{p_0, (l_1, p_1), ..., (l_{j-1}, p_{j-1}), \underline{(l_j, p_j)}, \underline{(l_{j+1}, p_{j+1})}, (l_{j+2}, p_{j+2}), ..., (l_N, p_N)\}$$

A processor arrangement determines which processor is connected to l_1, l_2, ..., l_N. A processor arrangement does not change the order of dispatching fractions of load from the root processor to links, i.e., an element l_j associated with each ordered pair is fixed during the course of processor arrangement. That is, the sequence of load distribution from the root processor does not change from the link point of view. This ordered set will be referred to as a processor arrangement profile. Therefore, a processor arrangement is

a mechanism to change from one processor arrangement profile to another processor arrangement profile. In contrast to the sequencing mechanism of [7], a processor arrangement requires a physical change of a link-processor pairs through processor reordering. In this work, a processor arrangement is performed to minimize total processing cost. One important specialization of processor arrangement is an adjacent pairwise swapped processor arrangement which will be discussed later.

3.2 Link-Processor Cost

The link-processor cost for processing a fraction of load at any processor is defined as the cost incurred from utilizing the processor and its corresponding link in order to successfully process the underlying fraction of load. Therefore, the link-processor cost consists of two major parts: the one incurred by communication over the link and the other incurred by the processor. Throughout this paper, we assume that the cost coefficients associated with links and processors are static. They do not change with either the level of load in progress or the time when the job arrives. This cost is defined only in terms of accounting for the duration during which the resource is busy serv-

ing the assigned divisible load. The link-processor cost is thus a monotonic increasing function of the service duration and moreover is a linear, regular and additive function. The processing costs associated with each network topology are as follows. Let:

w_n: the inverse of the computing speed of the n^{th} processor, with the unit of second per load.

z_n: the inverse of the link speed of the n^{th} link, with the unit of second per load.

c_n^p: the computing cost per second of utilizing the n^{th} processor.

c_n^l: the communication cost per second of utilizing the n^{th} link.

$c_n^p w_n$: the computing cost per load of utilizing the n^{th} processor.

$c_n^l z_n$: the communication cost per load of utilizing the n^{th} link.

$(c_n^p w_n + c_n^l z_n)$: the processing cost per load of the n^{th} link-processor pair.

3.3 Total Cost

Total cost is a cost incurred for a network to process an entire load. It is a linear addition of all individual link-processor costs incurred by utilizing individual link-processor pairs. This individual cost depends on the assigned

fraction of load, which in turn is determined by a processor arrangement profile (by "profile" is meant a specific arrangement of processors). Therefore, this total cost depends on the processor arrangement profile.

In this subsection, the general form of the total cost in a single-level tree network is developed. Also its simple form, which is a ratio of numerator and denominator, is given. This simple form will facilitate the subsequent analysis.

Define:

$$C_0 = c_0^p w_0 T_{cp} \tag{13}$$

$$C_n = c_n^l z_n T_{cm} + c_n^p w_n T_{cp} \qquad , n = 1, ..., N \tag{14}$$

Recall that the root processor is the load origination processor. Therefore no communication cost is incurred by the root processor.

Now:

C_0: the cost of processing the entire of load on the root processor.

C_n: the cost of processing the entire of load on the n^{th} processor.

$\alpha_0 C_0$: the cost of processing the assigned fraction of load (α_0) on the root processor.

$\alpha_n C_n$: the cost of processing the assigned fraction of load (α_n) on

the n^{th} processor.

The total cost, C_{total}, is defined as a summation of the individual processing costs incurred at each link-processor pair. That is:

$$C_{total} = \alpha_0 C_0 + \sum_{n=1}^{N} \alpha_n C_n \tag{15}$$

$$= \alpha_0 (c_0^p w_0 T_{cp}) + \sum_{n=1}^{N} \alpha_n \left(c_n^l z_n T_{cm} + c_n^p w_n T_{cp} \right) \tag{16}$$

Note that in the following, in terms of notation the product signs do not distribute over other product signs.

Now, by substituting α_0 and all α_n from the previous section into equation (16) one obtains:

$$C_{total} = \frac{1}{D} \left\{ \prod_{i=1}^{N} (z_i T_{cm} + w_i T_{cp})(c_0^p w_0 T_{cp}) + \sum_{n=1}^{N} \left[\prod_{i=0}^{n-1} (w_i T_{cp}) \cdot \prod_{i=n+1}^{N} (z_i T_{cm} + w_i T_{cp})(c_n^l z_n T_{cm} + c_n^p w_n T_{cp}) \right] \right\} \tag{17}$$

One can also express the total cost while explicitly showing the processing cost incurred by the j^{th} and the $(j+1)^{st}$ link-processor pairs as:

$$
\begin{aligned}
C_{total} \;=\; & \frac{1}{D}\Bigg\{ \prod_{i=1}^{N}(z_i T_{cm} + w_i T_{cp})(c_0^p w_0 T_{cp}) \\
& + \sum_{n=1}^{j-1}\left[\prod_{i=0}^{n-1}(w_i T_{cp}) \prod_{i=n+1}^{N}(z_i T_{cm} + w_i T_{cp})(c_n^l z_n T_{cm} + c_n^p w_n T_{cp})\right] \\
& + \prod_{i=0}^{j-1}(w_i T_{cp}) \prod_{i=j+1}^{N}(z_i T_{cm} + w_i T_{cp})(c_j^l z_j T_{cm} + c_j^p w_j T_{cp}) \\
& + \prod_{i=0}^{j}(w_i T_{cp}) \prod_{i=j+2}^{N}(z_i T_{cm} + w_i T_{cp})(c_{j+1}^l z_{j+1} T_{cm} + c_{j+1}^p w_{j+1} T_{cp}) \\
& + \sum_{n=j+2}^{N}\left[\prod_{i=0}^{n-1}(w_i T_{cp}) \prod_{i=n+1}^{N}(z_i T_{cm} + w_i T_{cp})(c_n^l z_n T_{cm} + c_n^p w_n T_{cp})\right]\Bigg\}
\end{aligned}
\tag{18}
$$

Since the total cost can be put in a simple form as:

$$
C_{total} \;=\; \frac{N}{D} \tag{19}
$$

Thus, the corresponding numerator, N, is:

$$
\begin{aligned}
N \;=\; & \prod_{i=1}^{N}(z_i T_{cm} + w_i T_{cp})(c_0^p w_0 T_{cp}) \\
& + \sum_{n=1}^{j-1}\left[\prod_{i=0}^{n-1}(w_i T_{cp}) \prod_{i=n+1}^{N}(z_i T_{cm} + w_i T_{cp})(c_n^l z_n T_{cm} + c_n^p w_n T_{cp})\right] \\
& + \prod_{i=0}^{j-1}(w_i T_{cp}) \prod_{i=j+1}^{N}(z_i T_{cm} + w_i T_{cp})(c_j^l z_j T_{cm} + c_j^p w_j T_{cp}) \\
& + \prod_{i=0}^{j}(w_i T_{cp}) \prod_{i=j+2}^{N}(z_i T_{cm} + w_i T_{cp})(c_{j+1}^l z_{j+1} T_{cm} + c_{j+1}^p w_{j+1} T_{cp}) \\
& + \sum_{n=j+2}^{N}\left[\prod_{i=0}^{n-1}(w_i T_{cp}) \prod_{i=n+1}^{N}(z_i T_{cm} + w_i T_{cp})(c_n^l z_n T_{cm} + c_n^p w_n T_{cp})\right]
\end{aligned}
\tag{20}
$$

(21)

Again, with the terms due to the j^{th} and the $(j+1)^{st}$ link-processor explicitly shown, one has

$$
\begin{aligned}
D \;=\; & \prod_{i=1}^{j-1}(z_iT_{cm}+w_iT_{cp})(z_jT_{cm}+w_jT_{cp})(z_{j+1}T_{cm}+w_{j+1}T_{cp})\prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp}) \\
& +\sum_{n=1}^{j-1}\left(\prod_{i=0}^{n-1}(w_iT_{cp})\prod_{i=n+1}^{j-1}(z_iT_{cm}+w_iT_{cp})(z_jT_{cm}+w_jT_{cp})(z_{j+1}T_{cm}+w_{j+1}T_{cp})\right. \\
& \left.\cdot\prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp})\right) \\
& +\prod_{i=0}^{j-2}(w_iT_{cp})(w_{j-1}T_{cp})(z_{j+1}T_{cm}+w_{j+1}T_{cp})\prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp}) \\
& +\prod_{i=0}^{j-2}(w_iT_{cp})(w_{j-1}T_{cp})(w_jT_{cp})\prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp}) \\
& +\sum_{n=j+2}^{N}\left(\prod_{i=0}^{n-1}(w_iT_{cp})\prod_{i=n+1}^{N}(z_iT_{cm}+w_iT_{cp})\right) \qquad (22)
\end{aligned}
$$

This rational form of a numerator and a denominator of the total computing cost is useful in the subsequent analysis.

3.4 Cost Optimization

There are actually two optimization criteria involved in the problem considered in this paper. One is the above total cost and the second is finish

time. Generally, both should be minimized as much as possible. In this paper we choose to minimize total cost over all possible processor arrangements with finish time being minimized for the *given* processor arrangement chosen. While this is a natural and simple approach for this problem, other approaches to such dual optimization criteria problems are certainly possible.

3.5 Adjacent Pairwise Processor Swapping

3.5.1 Concepts and Notations

Adjacent pairwise processor swapping refers to a physical interchange of two processors in an adjacent link-processor pair of the current processor arrangement profile, keeping all other link-processor pairs in their respective positions.

Consider a processor arrangement profile called the "current" processor arrangement profile as shown in Figure 1. A swapped processor arrangement profile is a profile obtained by implementing a single adjacent pairwise processor swap of one of the adjacent link-processor pairs of the current profile as shown in Figure 4, a swap of p_j and p_{j+1}.

Here the term "current" profile is used with a view towards the algorithm

developed later.

In the ordered set representation, a current profile and an associated swapped profile can be expressed respectively as,

$$\pi = \{p_0, (l_1, p_1), ..., (l_{j-1}, p_{j-1}), \underline{(l_j, p_j)}, \underline{(l_{j+1}, p_{j+1})}, (l_{j+2}, p_{j+2}), ..., (l_N, p_N)\}$$

$$\pi' = \{p_0, (l_1, p_1), ..., (l_{j-1}, p_{j-1}), \underline{(l_j, p_{j+1})}, \underline{(l_{j+1}, p_j)}, (l_{j+2}, p_{j+2}), ..., (l_N, p_N)\}$$

3.5.2 Recursive Equations

As in section 2.4, we can derive a closed-form solution of the load fraction of each link-processor pair under an adjacent pairwise processor swapped arrangement π' (cf. Figure 5) as follow.

The set of general recursive equations of an adjacent pairwise processor swapped arrangement analogous to equation (1) and equation (2) is given as,

$$\alpha'_i w'_i T_{cp} = \alpha'_{i+1} z'_{i+1} T_{cm} + \alpha'_{i+1} w'_{i+1} T_{cp} \quad i = 0, ..., N-1 \qquad (23)$$

$$\alpha'_{i+1} = k'_i \alpha'_i \qquad (24)$$

Here, a mapping of w'_i to w_i of the original processor arrangement is given as follows

$$w'_j = w_{j+1}$$

$$w'_{j+1} = w_j$$

$$w'_k = w_k, \quad \forall k \neq j, j+1$$

One then has the following series of equations

$$\begin{aligned}
\alpha'_0 w_0 T_{cp} &= \alpha'_1 z_1 T_{cm} + \alpha'_1 w_1 T_{cp} \\
\alpha'_1 &= \left(\frac{w_0 T_{cp}}{z_1 T_{cm} + w_1 T_{cp}} \right) \alpha'_0 \\
&\vdots \\
\alpha'_{j-1} w_{j-1} T_{cp} &= \alpha'_j z_j T_{cm} + \alpha'_j w_{j+1} T_{cp} \\
\alpha'_j &= \left(\frac{w_{j-1} T_{cp}}{z_j T_{cm} + w_{j+1} T_{cp}} \right) \alpha'_{j-1} \\
\alpha'_j w_{j+1} T_{cp} &= \alpha'_{j+1} z_{j+1} T_{cm} + \alpha'_{j+1} w_j T_{cp} \\
\alpha'_{j+1} &= \left(\frac{w_{j+1} T_{cp}}{z_{j+1} T_{cm} + w_j T_{cp}} \right) \alpha'_j \\
\alpha'_{j+1} w_j T_{cp} &= \alpha'_{j+2} z_{j+2} T_{cm} + \alpha'_{j+2} w_{j+2} T_{cp} \\
\alpha'_{j+2} &= \left(\frac{w_j T_{cp}}{z_{j+2} T_{cm} + w_{j+2} T_{cp}} \right) \alpha'_{j+1} \\
&\vdots \\
\alpha'_{N-1} w_{N-1} T_{cp} &= \alpha'_N z_N T_{cm} + \alpha'_N w_N T_{cp} \\
\alpha'_N &= \left(\frac{w_{N-1} T_{cp}}{z_N T_{cm} + w_N T_{cp}} \right) \alpha'_{N-1}
\end{aligned}$$

Again incorporating the normalization equation, one solves the above system of $(N+1)$ equations and $(N+1)$ unknowns to obtain an expression for α'_0. Once α'_0 known, by recursively substituting α'_0 into the other equations, then all other α'_n will be obtained as below.

That is

$$\alpha'_0 = \frac{1}{D_{\pi'}} \prod_{i=1}^{N} (z_i T_{cm} + w_i T_{cp}) \tag{25}$$

In the explicit form the j^{th} and $(j+1)^{st}$ terms can be provided as

$$\begin{aligned}
\alpha'_0 &= \frac{1}{D_{\pi'}} \left(\prod_{i=1}^{j-1} (z_i T_{cm} + w_i T_{cp})(z_j T_{cm} + w_{j+1} T_{cp})(z_{j+1} T_{cm} + w_j T_{cp}) \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \right) \\
\alpha'_1 &= \frac{1}{D_{\pi'}} \left((w_0 T_{cp}) \prod_{i=2}^{j-1} (z_i T_{cm} + w_i T_{cp})(z_j T_{cm} + w_{j+1} T_{cp})(z_{j+1} T_{cm} + w_j T_{cp}) \right. \\
&\quad \left. \cdot \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \right) \\
&\vdots \\
\alpha'_{j-1} &= \frac{1}{D_{\pi'}} \left(\prod_{i=0}^{j-2} (w_i T_{cp})(z_j T_{cm} + w_{j+1} T_{cp})(z_{j+1} T_{cm} + w_j T_{cp}) \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \right) \\
\alpha'_j &= \frac{1}{D_{\pi'}} \left(\prod_{i=0}^{j-2} (w_i T_{cp})(w_{j-1} T_{cp})(z_{j+1} T_{cm} + w_j T_{cp}) \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \right) \\
\alpha'_{j+1} &= \frac{1}{D_{\pi'}} \left(\prod_{i=0}^{j-2} (w_i T_{cp})(w_{j-1} T_{cp})(w_{j+1} T_{cp}) \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \right) \\
&\vdots
\end{aligned}$$

$$\alpha'_N = \frac{1}{D_{\pi'}} \left(\prod_{i=0}^{N-1} (w_i T_{cp}) \right)$$

where,

$$\begin{aligned} D_{\pi'} = & \prod_{i=1}^{j-1} (z_i T_{cm} + w_i T_{cp})(z_j T_{cm} + w_{j+1} T_{cp})(z_{j+1} T_{cm} + w_j T_{cp}) \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \\ & + \sum_{n=1}^{j-1} \left(\prod_{i=0}^{n-1} (w_i T_{cp}) \prod_{i=n+1}^{j-1} (z_i T_{cm} + w_i T_{cp})(z_j T_{cm} + w_{j+1} T_{cp})(z_{j+1} T_{cm} + w_j T_{cp}) \right. \\ & \left. \cdot \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \right) \\ & + \prod_{i=0}^{j-2} (w_i T_{cp})(w_{j-1} T_{cp})(z_{j+1} T_{cm} + w_j T_{cp}) \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \\ & + \prod_{i=0}^{j-2} (w_i T_{cp})(w_{j-1} T_{cp})(w_{j+1} T_{cp}) \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \\ & + \sum_{n=j+2}^{N} \left(\prod_{i=0}^{n-1} (w_i T_{cp}) \prod_{i=n+1}^{N} (z_i T_{cm} + w_i T_{cp}) \right) \end{aligned} \tag{26}$$

3.6 Some Total Computing Cost Related Equations

In this subsection, the relevant equations arising from a total computing cost performance comparison of an original arrangement and a swapped arrangement are given. For the sake of clarity, some pertinent terms are restated here.

C_{total} the total computing cost of an original arrangement (π).

C'_{total} the total computing cost of an adjacent pairwise processor swapped arrangement (π').

N_π the numerator of C_{total} of an original arrangement (π).

$N_{\pi'}$ the numerator of C_{total} of an adjacent pairwise processor swapped arrangement (π').

D_π the denominator of C_{total} of an original arrangement (π).

$D_{\pi'}$ the denominator of C_{total} of an adjacent pairwise processor swapped arrangement (π').

As mentioned in subsection 3.3, one can express the total computing cost in a simple form as,

$$C_{total} = \frac{N_\pi}{D_\pi}$$
$$C'_{total} = \frac{N_{\pi'}}{D_{\pi'}}$$

3.6.1 An Adjacent Pairwise Processor Arrangement

The total computing cost of an adjacent pairwise processor swapped arrangement can be stated as

$$C'_{total} = \alpha'_0 C'_0 + \sum_{n=1}^{N} \alpha'_n C'_n \tag{27}$$

$$= \alpha'_0(c_0^p w_0 T_{cp}) + \sum_{n=1}^{j-1} \alpha'_n \left(c_n^l z_n T_{cm} + c_n^p w_n T_{cp}\right) + \alpha'_j(c_j^l z_j T_{cm} + c_{j+1}^p w_{j+1} T_{cp})$$
$$+ \alpha'_{j+1}(c_{j+1}^l z_{j+1} T_{cm} + c_j^p w_j T_{cp}) + \sum_{n=j+2}^{N} \alpha'_n \left(c_n^l z_n T_{cm} + c_n^p w_n T_{cp}\right) \tag{28}$$

By substituting α'_n from the previous section into (28) one obtains,

$$
\begin{aligned}
C'_{total} \;=\; & \frac{1}{D_{\pi'}} \Bigg\{ \prod_{i=1}^{j-1} (z_i T_{cm} + w_i T_{cp})(z_j T_{cm} + w_{j+1} T_{cp})(z_{j+1} T_{cm} + w_j T_{cp}) \\
& \cdot \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp})(c_0^p w_0 T_{cp}) \\
& + \sum_{n=1}^{j-1} \left(\prod_{i=0}^{n-1} (w_i T_{cp}) \prod_{i=n+1}^{j-1} (z_i T_{cm} + w_i T_{cp})(z_j T_{cm} + w_{j+1} T_{cp})(z_{j+1} T_{cm} + w_j T_{cp}) \right. \\
& \cdot \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp})(c_n^l z_n T_{cm} + c_n^p w_n T_{cp}) \\
& + \prod_{i=0}^{j-2} (w_i T_{cp})(w_{j-1} T_{cp})(z_{j+1} T_{cm} + w_j T_{cp}) \\
& \cdot \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp})(c_j^l z_j T_{cm} + c_{j+1}^p w_{j+1} T_{cp}) \\
& + \prod_{i=0}^{j-2} (w_i T_{cp})(w_{j-1} T_{cp})(w_{j+1} T_{cp}) \\
& \cdot \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp})(c_{j+1}^l z_{j+1} T_{cm} + c_j^p w_j T_{cp}) \\
& \left. + \sum_{n=j+2}^{N} \left(\prod_{i=0}^{n-1} (w_i T_{cp}) \prod_{i=n+1}^{N} (z_i T_{cm} + w_i T_{cp})(c_n^l z_n T_{cm} + c_n^p w_n T_{cp}) \right) \right\} \qquad (29)
\end{aligned}
$$

where

$$
\begin{aligned}
N_{\pi'} \;=\; & \prod_{i=1}^{j-1} (z_i T_{cm} + w_i T_{cp})(z_j T_{cm} + w_{j+1} T_{cp})(z_{j+1} T_{cm} + w_j T_{cp}) \\
& \cdot \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp})(c_0^p w_0 T_{cp}) \\
& + \sum_{n=1}^{j-1} \left(\prod_{i=0}^{n-1} (w_i T_{cp}) \prod_{i=n+1}^{j-1} (z_i T_{cm} + w_i T_{cp})(z_j T_{cm} + w_{j+1} T_{cp})(z_{j+1} T_{cm} + w_j T_{cp}) \right.
\end{aligned}
$$

$$
\begin{aligned}
&\cdot \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp})(c_n^l z_n T_{cm} + c_n^p w_n T_{cp}) \Bigg) \\
&+ \prod_{i=0}^{j-2} (w_i T_{cp})(w_{j-1} T_{cp})(z_{j+1} T_{cm} + w_j T_{cp}) \\
&\cdot \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp})(c_j^l z_j T_{cm} + c_{j+1}^p w_{j+1} T_{cp}) \\
&+ \prod_{i=0}^{j-2} (w_i T_{cp})(w_{j-1} T_{cp})(w_{j+1} T_{cp}) \\
&\cdot \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp})(c_{j+1}^l z_{j+1} T_{cm} + c_j^p w_j T_{cp}) \\
&+ \sum_{n=j+2}^{N} \left(\prod_{i=0}^{n-1} (w_i T_{cp}) \prod_{i=n+1}^{N} (z_i T_{cm} + w_i T_{cp})(c_n^l z_n T_{cm} + c_n^p w_n T_{cp}) \right) \qquad (30)
\end{aligned}
$$

$$
\begin{aligned}
D_{\pi'} \;=\; & \prod_{i=1}^{j-1} (z_i T_{cm} + w_i T_{cp})(z_j T_{cm} + w_{j+1} T_{cp})(z_{j+1} T_{cm} + w_j T_{cp}) \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \\
&+ \sum_{n=1}^{j-1} \left(\prod_{i=0}^{n-1} (w_i T_{cp}) \prod_{i=n+1}^{j-1} (z_i T_{cm} + w_i T_{cp})(z_j T_{cm} + w_{j+1} T_{cp})(z_{j+1} T_{cm} + w_j T_{cp}) \right. \\
&\left. \cdot \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \right) \\
&+ \prod_{i=0}^{j-2} (w_i T_{cp})(w_{j-1} T_{cp})(z_{j+1} T_{cm} + w_j T_{cp}) \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \\
&+ \prod_{i=0}^{j-2} (w_i T_{cp})(w_{j-1} T_{cp})(w_{j+1} T_{cp}) \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \\
&+ \sum_{n=j+2}^{N} \left(\prod_{i=0}^{n-1} (w_i T_{cp}) \prod_{i=n+1}^{N} (z_i T_{cm} + w_i T_{cp}) \right) \qquad (31)
\end{aligned}
$$

3.6.2 An Original Processor Arrangement

The total computing cost of an original arrangement is exactly that derived in subsection 3.3. For clarity it is restated again,

$$
\begin{aligned}
N_\pi \;=\; & \prod_{i=1}^{j-1}(z_iT_{cm}+w_iT_{cp})(z_jT_{cm}+w_jT_{cp})(z_{j+1}T_{cm}+w_{j+1}T_{cp}) \\
& \cdot \prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp})(c_0^pw_0T_{cp}) \\
& + \sum_{n=1}^{j-1}\left(\prod_{i=0}^{n-1}(w_iT_{cp})\prod_{i=n+1}^{j-1}(z_iT_{cm}+w_iT_{cp})(z_jT_{cm}+w_jT_{cp})(z_{j+1}T_{cm}+w_{j+1}T_{cp})\right. \\
& \left. \cdot \prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp})(c_n^lz_nT_{cm}+c_n^pw_nT_{cp})\right) \\
& + \prod_{i=0}^{j-2}(w_iT_{cp})(w_{j-1}T_{cp})(z_{j+1}T_{cm}+w_{j+1}T_{cp}) \\
& \cdot \prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp})(c_j^lz_jT_{cm}+c_j^pw_jT_{cp}) \\
& + \prod_{i=0}^{j-2}(w_iT_{cp})(w_{j-1}T_{cp})(w_jT_{cp}) \\
& \cdot \prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp})(c_{j+1}^lz_{j+1}T_{cm}+c_{j+1}^pw_{j+1}T_{cp}) \\
& + \sum_{n=j+2}^{N}\left(\prod_{i=0}^{n-1}(w_iT_{cp})\prod_{i=n+1}^{N}(z_iT_{cm}+w_iT_{cp})(c_n^lz_nT_{cm}+c_n^pw_nT_{cp})\right) \qquad (32)
\end{aligned}
$$

Similarly we have the denominator for an original arrangement as follows

$$
\begin{aligned}
D_\pi \;=\; & \prod_{i=1}^{j-1}(z_iT_{cm}+w_iT_{cp})(z_jT_{cm}+w_jT_{cp})(z_{j+1}T_{cm}+w_{j+1}T_{cp})\prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp}) \\
& +\sum_{n=1}^{j-1}\left(\prod_{i=0}^{n-1}(w_iT_{cp})\prod_{i=n+1}^{j-1}(z_iT_{cm}+w_iT_{cp})(z_jT_{cm}+w_jT_{cp})(z_{j+1}T_{cm}+w_{j+1}T_{cp})\right. \\
& \left.\cdot\prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp})\right) \\
& +\prod_{i=0}^{j-2}(w_iT_{cp})(w_{j-1}T_{cp})(z_{j+1}T_{cm}+w_{j+1}T_{cp})\prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp}) \\
& +\prod_{i=0}^{j-2}(w_iT_{cp})(w_{j-1}T_{cp})(w_jT_{cp})\prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp}) \\
& +\sum_{n=j+2}^{N}\left(\prod_{i=0}^{n-1}(w_iT_{cp})\prod_{i=n+1}^{N}(z_iT_{cm}+w_iT_{cp})\right) \qquad (33)
\end{aligned}
$$

3.6.3 The Difference Equations

The difference between the numerators and the denominators, based on the information just developed, can be given as,

$$
\begin{aligned}
N_\pi - N_{\pi'} \;=\; & \prod_{i=1}^{j-1}(z_iT_{cm}+w_iT_{cp})(z_{j+1}-z_j)(w_j-w_{j+1})T_{cm}T_{cp} \\
& \cdot\prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp})(c_0^p w_0 T_{cp}) \\
& +\sum_{n=1}^{j-1}\left(\prod_{i=0}^{n-1}(w_iT_{cp})\prod_{i=n+1}^{j-1}(z_iT_{cm}+w_iT_{cp})(z_{j+1}-z_j)(w_j-w_{j+1})T_{cm}T_{cp}\right.
\end{aligned}
$$

$$\cdot \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp})(c_n^l z_n T_{cm} + c_n^p w_n T_{cp}) \Bigg)$$
$$+ \prod_{i=0}^{j-1} (w_i T_{cp}) \Big[(w_j - w_{j+1})(z_{j+1} c_{j+1}^l - z_j c_j^l) T_{cm} T_{cp}$$
$$+ z_{j+1} T_{cm} T_{cp} (w_j c_j^p - w_{j+1} c_{j+1}^p) \Big] \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \tag{34}$$

$$D_\pi - D_{\pi'} = \prod_{i=1}^{j-1} (z_i T_{cm} + w_i T_{cp})(z_{j+1} - z_j)(w_j - w_{j+1}) T_{cm} T_{cp} \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp})$$
$$+ \sum_{n=1}^{j-1} \Bigg(\prod_{i=0}^{n-1} (w_i T_{cp}) \prod_{i=n+1}^{j-1} (z_i T_{cm} + w_i T_{cp})(z_{j+1} - z_j)(w_j - w_{j+1}) T_{cm} T_{cp}$$
$$\cdot \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \Bigg) \tag{35}$$

To simplify further, define

$$a = \prod_{i=1}^{j-1} (z_i T_{cm} + w_i T_{cp}) \tag{36}$$

$$b = \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \tag{37}$$

$$d = \prod_{i=0}^{j-1} (w_i T_{cp}) \tag{38}$$

$$C_0 = c_0^p w_0 T_{cp} \tag{39}$$

$$C_n = c_n^l z_n T_{cm} + c_n^p w_n T_{cp} \tag{40}$$

$$k_n = \prod_{i=0}^{n-1} (w_i T_{cp}) \prod_{i=n+1}^{j-1} (z_i T_{cm} + w_i T_{cp}) \tag{41}$$

Hence one has the simpler forms as below,

$$\begin{aligned} N_\pi - N_{\pi'} &= a\left[(z_{j+1}-z_j)(w_j-w_{j+1})T_{cm}T_{cp}\right]bC_0 \\ &\quad + \sum_{n=1}^{j-1}\left(k_n\left[(z_{j+1}-z_j)(w_j-w_{j+1})T_{cm}T_{cp}\right]bC_n\right) \\ &\quad + d\left[(w_j-w_{j+1})(z_{j+1}c^l_{j+1}-z_jc^l_j)+z_{j+1}(w_jc^p_j-w_{j+1}c^p_{j+1})\right]T_{cm}T_{cp}b \\ &= (z_{j+1}-z_j)(w_j-w_{j+1})T_{cm}T_{cp}b\left[aC_0+\sum_{n=1}^{j-1}k_nC_n\right] \\ &\quad + d\left[(w_j-w_{j+1})(z_{j+1}c^l_{j+1}-z_jc^l_j)\right. \\ &\quad \left. + z_{j+1}(w_jc^p_j-w_{j+1}c^p_{j+1})\right]T_{cm}T_{cp}b \end{aligned} \tag{42}$$

$$\begin{aligned} D_\pi - D_{\pi'} &= a\left[(z_{j+1}-z_j)(w_j-w_{j+1})T_{cm}T_{cp}\right]b \\ &\quad + \sum_{n=1}^{j-1}\left(k_n\left[(z_{j+1}-z_j)(w_j-w_{j+1})T_{cm}T_{cp}\right]b\right) \\ &= (z_{j+1}-z_j)(w_j-w_{j+1})T_{cm}T_{cp}\left[a+\sum_{n=1}^{j-1}k_n\right]b \end{aligned} \tag{43}$$

4 Cost Efficient Processor Arrangement

In this part, the conditions under which by transposing an adjacent pairwise processor pair the total computing cost performance will improve are found. The conditions under which it is better not transpose an adjacent pairwise processor pair are also found.

4.1 The C_{total} Conditions

In this subsection, we first use the simple expression of computing cost, the rational form, to derive some intermediate results. These results exhibit the total computing cost relationships of an original processor arrangement profile, C_{total}, and that of a transposed processor arrangement profile, C'_{total}. Incorporated with results from the previous sections regarding the numerators and denominators of the total computing costs, a number of lemmas are then provided.

We can state the difference of the total computing costs of an adjacent pairwise processor arrangement and an original arrangement in a simple form as,

$$C'_{total} - C_{total} = \frac{N_{\pi'}}{D_{\pi'}} - \frac{N_{\pi}}{D_{\pi}} \quad (44)$$

$$= \frac{N_{\pi'}D_{\pi} - N_{\pi}D_{\pi'}}{D_{\pi}D_{\pi'}} \quad (45)$$

To obtain the optimal conditions, the expression in (45) will be used. Note that both denominators, D_{π} and $D_{\pi'}$, are positive. Therefore to determine the relationships between the total computing cost, it is suffice to consider only the numerator, $N_{\pi'}D_{\pi} - N_{\pi}D_{\pi'}$.

Lemma 1 *In a single-level tree network, the total cost of a current proces-*

sor arrangement, C_{total}, is less than that of an associated swapped processor arrangement, C'_{total}, if one of the following conditions holds:

1. $N_{\pi'} > N_{\pi}$, $D_{\pi'} < D_{\pi}$.
2. $N_{\pi'} > N_{\pi}$, $D_{\pi'} = D_{\pi}$.
3. $N_{\pi'} > N_{\pi}$, $D_{\pi'} > D_{\pi}$ *and* $N_{\pi'}D_{\pi} > N_{\pi}D_{\pi'}$.
4. $N_{\pi'} = N_{\pi}$, $D_{\pi'} < D_{\pi}$.
5. $N_{\pi'} < N_{\pi}$, $D_{\pi'} < D_{\pi}$ *and* $N_{\pi'}D_{\pi} > N_{\pi}D_{\pi'}$.

Proof

From equation (45):

$$C'_{total} - C_{total} = \frac{N_{\pi'}D_{\pi} - N_{\pi}D_{\pi'}}{D_{\pi}D_{\pi'}}$$

Thus, $C'_{total} > C_{total}$ when :

$$N_{\pi'}D_{\pi} - N_{\pi}D_{\pi'} > 0 \tag{46}$$

$$N_{\pi'}D_{\pi} > N_{\pi}D_{\pi'} \tag{47}$$

By checking all possible cases of the relationship of (N_{π},$N_{\pi'}$) and that of (D_{π},$D_{\pi'}$) that satisfy equation (46) and equation (47), only these conditions result. Thus the lemma is proven. □

Lemma 2 *In a single-level tree network, the total cost of a current processor arrangement, C_{total}, is equal to that of an associated swapped processor arrangement, C'_{total}, if one of the following conditions holds:*

1. $N_{\pi'} > N_{\pi}$, $D_{\pi'} > D_{\pi}$ *and* $N_{\pi'}D_{\pi} = N_{\pi}D_{\pi'}$.
2. $N_{\pi'} = N_{\pi}$, $D_{\pi'} = D_{\pi}$.
3. $N_{\pi'} < N_{\pi}$, $D_{\pi'} < D_{\pi}$ *and* $N_{\pi'}D_{\pi} = N_{\pi}D_{\pi'}$.

Proof

From equation (45):

$$C'_{total} - C_{total} = \frac{N_{\pi'}D_{\pi} - N_{\pi}D_{\pi'}}{D_{\pi}D_{\pi'}}$$

Thus, $C'_{total} = C_{total}$ when :

$$N_{\pi'}D_{\pi} - N_{\pi}D_{\pi'} = 0 \tag{48}$$

$$N_{\pi'}D_{\pi} = N_{\pi}D_{\pi'} \tag{49}$$

By checking all possible cases of the relationship of (N_{π},$N_{\pi'}$) and that of (D_{π},$D_{\pi'}$) that satisfy equation (48) and equation (49), only these conditions result. Thus the lemma is proven. □

Lemma 3 *In a single-level tree network, the total cost of a current processor*

arrangement, C_{total}, is greater than that of an associated swapped processor arrangement, C'_{total}, if one of the following conditions holds:

1. $N_{\pi'} < N_{\pi}$, $D_{\pi'} > D_{\pi}$.
2. $N_{\pi'} < N_{\pi}$, $D_{\pi'} = D_{\pi}$.
3. $N_{\pi'} < N_{\pi}$, $D_{\pi'} < D_{\pi}$ *and* $N_{\pi'}D_{\pi} < N_{\pi}D_{\pi'}$.
4. $N_{\pi'} = N_{\pi}$, $D_{\pi'} > D_{\pi}$.
5. $N_{\pi'} > N_{\pi}$, $D_{\pi'} > D_{\pi}$ *and* $N_{\pi'}D_{\pi} < N_{\pi}D_{\pi'}$.

Proof

From equation (45):

$$C'_{total} - C_{total} = \frac{N_{\pi'}D_{\pi} - N_{\pi}D_{\pi'}}{D_{\pi}D_{\pi'}}$$

Thus, $C'_{total} < C_{total}$ when :

$$N_{\pi'}D_{\pi} - N_{\pi}D_{\pi'} < 0. \tag{50}$$

$$N_{\pi'}D_{\pi} < N_{\pi}D_{\pi'} \tag{51}$$

By checking all possible cases of the relationship of (N_{π},$N_{\pi'}$) and of (D_{π},$D_{\pi'}$) that satisfy equation (50) and equation (51), only these conditions result. Thus the lemma is proven. □

Lemma 4 *In a single-level tree network, the relationship between the numerator of a current processor arrangement (N_π) and the numerator of an associated swapped processor arrangement ($N_{\pi'}$) can be stated equivalently in terms of z_j, z_{j+1}, C_j, and C_{j+1} as follows:*

1. $N_{\pi'} > N_\pi$ *iff*

$$\left[(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) + z_{j+1}(w_j c^p_j - w_{j+1}c^p_{j+1})\right] < \frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1}) \cdot \left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\}$$

2. $N_{\pi'} = N_\pi$ *iff*

$$\left[(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) + z_{j+1}(w_j c^p_j - w_{j+1}c^p_{j+1})\right] = \frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1}) \cdot \left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\}$$

3. $N_{\pi'} < N_\pi$ *iff*

$$\left[(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) + z_{j+1}(w_j c^p_j - w_{j+1}c^p_{j+1})\right] > \frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1}) \cdot \left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\}$$

Proof

From the difference equation of equation (42):

$$N_\pi - N_{\pi'} = (z_{j+1} - z_j)(w_j - w_{j+1})T_{cm}T_{cp}b\left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]$$

$$+d\left[(w_j - w_{j+1})(z_{j+1}c_{j+1}^l - z_j c_j^l)\right.$$

$$\left. + z_{j+1}(w_j c_j^p - w_{j+1}c_{j+1}^p)\right] T_{cm}T_{cp}b$$

Then, these conditions are obvious from the above equation. □

Lemma 5 *In a single-level tree network, the relationship between the denominator of a current processor arrangement (D_π) and the denominator of an associated swapped processor arrangement ($D_{\pi'}$) can be stated equivalently in terms of z_j, z_{j+1}, C_j, and C_{j+1} as follows:*

1. $D_{\pi'} > D_\pi$ *iff*

$$(z_{j+1} - z_j)(w_j - w_{j+1}) < 0$$

2. $D_{\pi'} = D_\pi$ *iff*

$$(z_{j+1} - z_j)(w_j - w_{j+1}) = 0$$

3. $D_{\pi'} < D_\pi$ *iff*

$$(z_{j+1} - z_j)(w_j - w_{j+1}) > 0$$

Proof

From the difference equation of equation (43):

$$D_\pi - D_{\pi'} = (z_{j+1} - z_j)(w_j - w_{j+1})T_{cm}T_{cp}\left[a + \sum_{n=1}^{j-1} k_n\right] b$$

Then, these conditions are obvious from the above equation. □

4.2 The Intermediate Results

From the lemmas in the previous subsection, a number of intermediate results can be derived as follows.

4.2.1 The Intermediate Results from Lemma 1 where $C'_{total} > C_{total}$

1. $N_{\pi'} > N_{\pi}$, $D_{\pi'} < D_{\pi}$

   That is,

$$\left[(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) + z_{j+1}(w_j c^p_j - w_{j+1}c^p_{j+1})\right] < \frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1})\left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\} \quad (52)$$

   and

$$(z_{j+1} - z_j)(w_j - w_{j+1}) > 0 \quad (53)$$

2. $N_{\pi'} > N_{\pi}$, $D_{\pi'} = D_{\pi}$

   That is,

$$\left[(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) + z_{j+1}(w_j c^p_j - w_{j+1}c^p_{j+1})\right] < \frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1})\left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\} \quad (54)$$

and

$$(z_{j+1} - z_j)(w_j - w_{j+1}) = 0 \tag{55}$$

3. $N_{\pi'} > N_\pi$, $D_{\pi'} > D_\pi$ and $N_{\pi'} D_\pi > N_\pi D_{\pi'}$

That is

$$\left[(w_j - w_{j+1})(z_{j+1} c_{j+1}^l - z_j c_j^l) + z_{j+1}(w_j c_j^p - w_{j+1} c_{j+1}^p)\right] < \frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1})\left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\} \tag{56}$$

and

$$(z_{j+1} - z_j)(w_j - w_{j+1}) < 0 \tag{57}$$

and

$$N_{\pi'} D_\pi > N_\pi D_{\pi'}$$

4. $N_{\pi'} = N_\pi$, $D_{\pi'} < D_\pi$

That is,

$$\left[(w_j - w_{j+1})(z_{j+1} c_{j+1}^l - z_j c_j^l) + z_{j+1}(w_j c_j^p - w_{j+1} c_{j+1}^p)\right] = \frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1})\left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\} \tag{58}$$

and

$$(z_{j+1} - z_j)(w_j - w_{j+1}) > 0 \tag{59}$$

5. $N_{\pi'} < N_\pi$, $D_{\pi'} < D_\pi$ and $N_{\pi'}D_\pi > N_\pi D_{\pi'}$

That is

$$\left[(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) + z_{j+1}(w_j c^p_j - w_{j+1}c^p_{j+1})\right] > \frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1})\left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\} \quad (60)$$

and

$$(z_{j+1} - z_j)(w_j - w_{j+1}) > 0 \quad (61)$$

and

$$N_{\pi'}D_\pi > N_\pi D_{\pi'}$$

4.2.2 The Intermediate Results from Lemma 2 where $C'_{total} = C_{total}$

1. $N_{\pi'} > N_\pi$, $D_{\pi'} > D_\pi$ and $N_{\pi'}D_\pi = N_\pi D_{\pi'}$

That is

$$\left[(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) + z_{j+1}(w_j c^p_j - w_{j+1}c^p_{j+1})\right] < \frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1})\left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\} \quad (62)$$

and

$$(z_{j+1} - z_j)(w_j - w_{j+1}) < 0 \quad (63)$$

and

$$N_{\pi'} D_{\pi} = N_{\pi} D_{\pi'}$$

2. $N_{\pi'} = N_{\pi}$, $D_{\pi'} = D_{\pi}$

That is,

$$\left[(w_j - w_{j+1})(z_{j+1} c_{j+1}^{l} - z_j c_j^{l}) + z_{j+1}(w_j c_j^{p} - w_{j+1} c_{j+1}^{p})\right] = \frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1})\left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\} \tag{64}$$

and

$$(z_{j+1} - z_j)(w_j - w_{j+1}) = 0 \tag{65}$$

3. $N_{\pi'} < N_{\pi}$, $D_{\pi'} < D_{\pi}$ and $N_{\pi'} D_{\pi} = N_{\pi} D_{\pi'}$

That is

$$\left[(w_j - w_{j+1})(z_{j+1} c_{j+1}^{l} - z_j c_j^{l}) + z_{j+1}(w_j c_j^{p} - w_{j+1} c_{j+1}^{p})\right] > \frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1})\left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\} \tag{66}$$

and

$$(z_{j+1} - z_j)(w_j - w_{j+1}) > 0 \tag{67}$$

and

$$N_{\pi'} D_{\pi} = N_{\pi} D_{\pi'}$$

4.2.3 The Intermediate Results from Lemma 3 where $C'_{total} < C_{total}$

1. $N_{\pi'} < N_{\pi}$, $D_{\pi'} > D_{\pi}$

   That is,

$$\left[(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) + z_{j+1}(w_j c^p_j - w_{j+1}c^p_{j+1})\right] > \frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1})\left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\} \quad (68)$$

   and

$$(z_{j+1} - z_j)(w_j - w_{j+1}) < 0 \quad (69)$$

2. $N_{\pi'} < N_{\pi}$, $D_{\pi'} = D_{\pi}$

   That is,

$$\left[(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) + z_{j+1}(w_j c^p_j - w_{j+1}c^p_{j+1})\right] > \frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1})\left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\} \quad (70)$$

   and

$$(z_{j+1} - z_j)(w_j - w_{j+1}) = 0 \quad (71)$$

3. $N_{\pi'} < N_{\pi}$, $D_{\pi'} < D_{\pi}$ and $N_{\pi'}D_{\pi} < N_{\pi}D_{\pi'}$

   That is

$$\left[(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) + z_{j+1}(w_j c^p_j - w_{j+1}c^p_{j+1})\right] >$$

$$\frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1})\left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\} \qquad (72)$$

and

$$(z_{j+1} - z_j)(w_j - w_{j+1}) > 0 \qquad (73)$$

and

$$N_{\pi'} D_\pi < N_\pi D_{\pi'}$$

4. $N_{\pi'} = N_\pi$, $D_{\pi'} > D_\pi$

That is,

$$\left[(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) + z_{j+1}(w_j c^p_j - w_{j+1}c^p_{j+1})\right] = \frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1})\left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\} \qquad (74)$$

and

$$(z_{j+1} - z_j)(w_j - w_{j+1}) < 0 \qquad (75)$$

5. $N_{\pi'} > N_\pi$, $D_{\pi'} > D_\pi$ and $N_{\pi'} D_\pi < N_\pi D_{\pi'}$

That is

$$\left[(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) + z_{j+1}(w_j c^p_j - w_{j+1}c^p_{j+1})\right] < \frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1})\left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\} \qquad (76)$$

and

$$(z_{j+1} - z_j)(w_j - w_{j+1}) < 0 \tag{77}$$

and

$$N_{\pi'} D_\pi < N_\pi D_{\pi'}$$

4.3 The Main Results

From the findings in the previous subsection, the conditions for processor arrangement can be divided into two cases as follows:

4.3.1 Case I $(z_j - z_{j+1})(w_j - w_{j+1}) = 0$ or $D_{\pi'} = D_\pi$

This case includes:

1. $N_{\pi'} > N_\pi$, $D_{\pi'} = D_\pi$, where $C'_{total} > C_{total}$.

   From equation (54) and equation (55), one has:

   $$(w_j - w_{j+1})(z_{j+1} c^l_{j+1} - z_j c^l_j) < z_{j+1}(w_{j+1} c^p_{j+1} - w_j c^p_j) \tag{78}$$

2. $N_{\pi'} = N_\pi$, $D_{\pi'} = D_\pi$, where $C'_{total} = C_{total}$.

   From equation (64) and equation (65), one has:

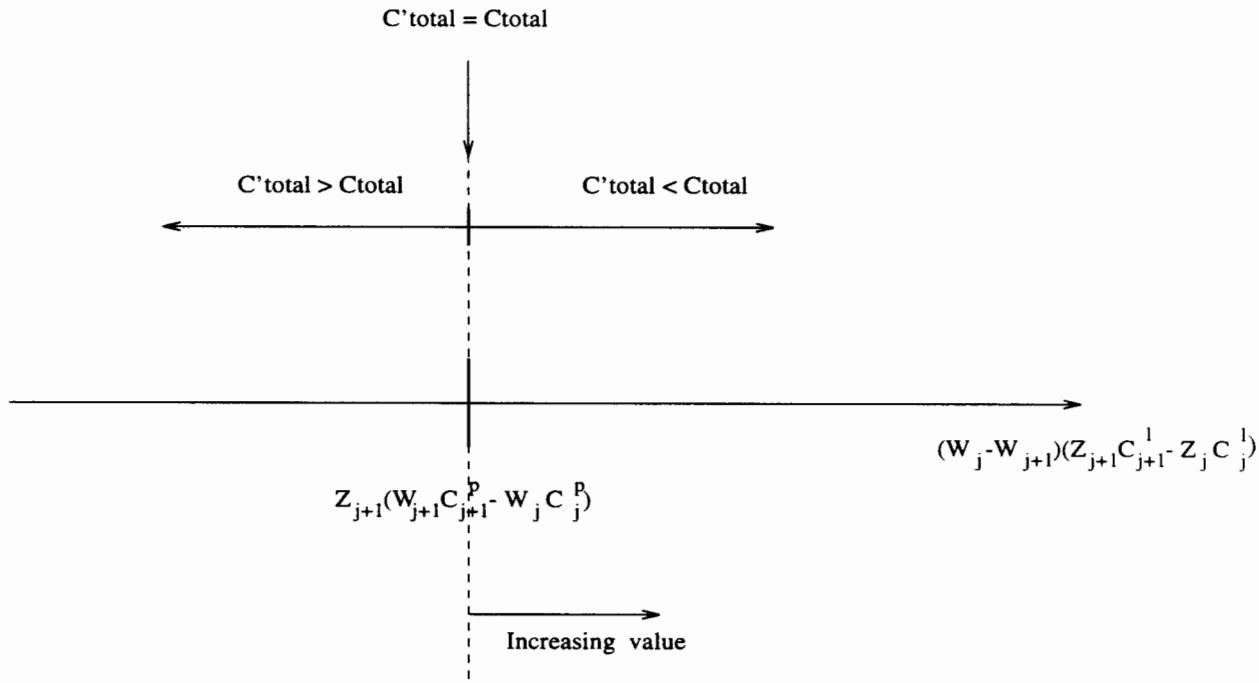
   $$(w_j - w_{j+1})(z_{j+1} c^l_{j+1} - z_j c^l_j) = z_{j+1}(w_{j+1} c^p_{j+1} - w_j c^p_j) \tag{79}$$

3. $N_{\pi'} < N_{\pi}$, $D_{\pi'} = D_{\pi}$, where $C'_{total} < C_{total}$.

From equation (70) and equation (71), one has:

$$(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) > z_{j+1}(w_{j+1}c^p_{j+1} - w_j c^p_j) \qquad (80)$$

It can be represented by the figure below:



Case I: $D'_{\pi'} = D'$

In this figure $(w_j - w_{j+1})(z_{j+1}c_{j+1}^l - z_j c_j^l)$ is a testing variable and $z_{j+1}(w_{j+1}c_{j+1}^p - w_j c_j^p)$ is a threshold. If the testing value $(w_j - w_{j+1})(z_{j+1}c_{j+1}^l - z_j c_j^l)$ is less than the threshold value $z_{j+1}(w_{j+1}c_{j+1}^p - w_j c_j^p)$, then $C'_{total} > C_{total}$. If they are equal, then $C'_{total} = C_{total}$. Otherwise $C'_{total} < C_{total}$.

4.3.2 Case II $(z_j - z_{j+1})(w_j - w_{j+1}) \neq 0$ or $D_{\pi'} \neq D_\pi$

Let

$$\Delta = \frac{1}{d}\left\{(z_j - z_{j+1})(w_j - w_{j+1})\left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\} \tag{81}$$

This case includes:

A. Lemma 1 Related Results

1. $N_{\pi'} > N_\pi$, $D_{\pi'} < D_\pi$, where $C'_{total} > C_{total}$.

   From equation (52) and equation (53), one has:

$$(w_j - w_{j+1})(z_{j+1}c_{j+1}^l - z_j c_j^l) < z_{j+1}(w_{j+1}c_{j+1}^p - w_j c_j^p) + \underbrace{\Delta}_{negative} \tag{82}$$

2. $N_{\pi'} > N_\pi$, $D_{\pi'} > D_\pi$, where $C'_{total} > C_{total}$.

   From equation (56) and equation (57), one has:

$$(w_j - w_{j+1})(z_{j+1}c_{j+1}^l - z_j c_j^l) < z_{j+1}(w_{j+1}c_{j+1}^p - w_j c_j^p) + \underbrace{\Delta}_{positive} \tag{83}$$

and

$$N_{\pi'}D_{\pi} \quad > \quad N_{\pi}D_{\pi'}$$

3. $N_{\pi'} = N_{\pi}$, $D_{\pi'} < D_{\pi}$, where $C'_{total} > C_{total}$.

   From equation (58) and equation (59), one has:

$$(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) = z_{j+1}(w_{j+1}c^p_{j+1} - w_j c^p_j) + \underbrace{\Delta}_{negative} \qquad (84)$$

4. $N_{\pi'} < N_{\pi}$, $D_{\pi'} < D_{\pi}$, where $C'_{total} > C_{total}$.

   From equation (60) and equation (61), one has:

$$(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) \quad > \quad z_{j+1}(w_{j+1}c^p_{j+1} - w_j c^p_j) + \underbrace{\Delta}_{negative} \qquad (85)$$

and

$$N_{\pi'}D_{\pi} \quad > \quad N_{\pi}D_{\pi'}$$

B. Lemma 2 Related Results

1. $N_{\pi'} > N_{\pi}$, $D_{\pi'} > D_{\pi}$, where $C'_{total} = C_{total}$.

   From equation (62) and equation (63), one has:

$$(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) \quad < \quad z_{j+1}(w_{j+1}c^p_{j+1} - w_j c^p_j) + \underbrace{\Delta}_{positive} \qquad (86)$$

and

$$N_{\pi'}D_{\pi} \quad = \quad N_{\pi}D_{\pi'}$$

2. $N_{\pi'} < N_{\pi}$, $D_{\pi'} < D_{\pi}$, where $C'_{total} = C_{total}$.

From equation (66) and equation (67), one has:

$$(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) \;>\; z_{j+1}(w_{j+1}c^p_{j+1} - w_j c^p_j) + \underbrace{\Delta}_{negative} \quad (87)$$

and

$$N_{\pi'}D_{\pi} \;=\; N_{\pi}D_{\pi'}$$

C. Lemma 3 Related Results

1. $N_{\pi'} < N_{\pi}$, $D_{\pi'} > D_{\pi}$, where $C'_{total} < C_{total}$.

From equation (68) and equation (69), one has:

$$(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) > z_{j+1}(w_{j+1}c^p_{j+1} - w_j c^p_j) + \underbrace{\Delta}_{positive} \quad (88)$$

2. $N_{\pi'} < N_{\pi}$, $D_{\pi'} < D_{\pi}$, where $C'_{total} < C_{total}$.

From equation (72) and equation (73), one has:

$$(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) \;>\; z_{j+1}(w_{j+1}c^p_{j+1} - w_j c^p_j) + \underbrace{\Delta}_{negative} \quad (89)$$

and

$$N_{\pi'}D_{\pi} \;<\; N_{\pi}D_{\pi'}$$

3. $N_{\pi'} = N_{\pi}$, $D_{\pi'} > D_{\pi}$, where $C'_{total} < C_{total}$.

From equation (74) and equation (75), one has:

$$(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) = z_{j+1}(w_{j+1}c^p_{j+1} - w_j c^p_j) + \underbrace{\Delta}_{positive} \quad (90)$$

4. $N_{\pi'} > N_\pi$, $D_{\pi'} > D_\pi$, where $C'_{total} < C_{total}$.

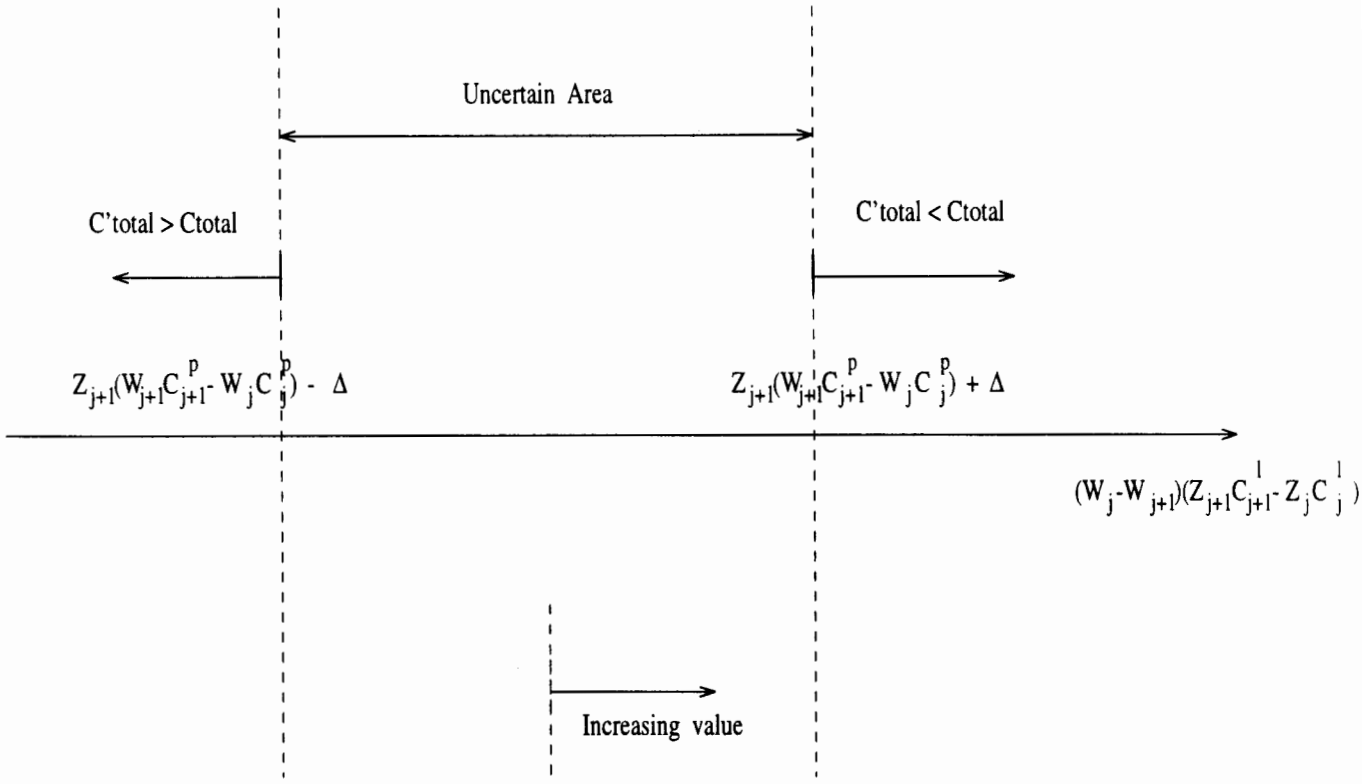
From equation (76) and equation (77), one has:

$$(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) \quad < \quad z_{j+1}(w_{j+1}c^p_{j+1} - w_j c^p_j) + \underbrace{\Delta}_{positive} \quad (91)$$

and

$$N_{\pi'}D_\pi \quad < \quad N_\pi D_{\pi'}$$

It can be represented by the figure below:



$$\Delta = \frac{1}{d}\left\{|(z_j - z_{j+1})(w_j - w_{j+1})|\left[aC_0 + \sum_{n=1}^{j-1} k_nC_n\right]\right\} \tag{92}$$

Note Δ is assumed positive here.

In this figure $(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_jc^l_j)$ is a testing variable, $z_{j+1}(w_{j+1}c^p_{j+1} - w_jc^p_j) + \Delta$ is an upper threshold, and $z_{j+1}(w_{j+1}c^p_{j+1} - w_jc^p_j) - \Delta$ is a lower threshold. If the testing value $(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_jc^l_j)$ is less than the lower threshold value $z_{j+1}(w_{j+1}c^p_{j+1} - w_jc^p_j) - \Delta$, then $C'_{total} > C_{total}$. If the

testing value $(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j)$ is greater than the upper threshold value $z_{j+1}(w_{j+1}c^p_{j+1} - w_j c^p_j) + \Delta$, then $C'_{total} < C_{total}$. Otherwise, the relationship of C'_{total} and C_{total} need further checking as will be discussed later.

4.3.3 Uncertain Area Revisited

The uncertain area resulted from the uncertain cases where $N_{\pi'} > N_\pi$, $D_{\pi'} > D_\pi$ according to section 4.3.2.A.2, 4.3.2.B.1, 4.3.2.C.4 and $N_{\pi'} < N_\pi$, $D_{\pi'} < D_\pi$ according to section 4.3.2.A.4, 4.3.2.B.2, 4.3.2.C.2. So as to simplify the conditions involved in that area, an alternative means of analysis of the two uncertain cases will be provided in the following.

1. $N_{\pi'} > N_\pi$, $D_{\pi'} > D_\pi$

   Let

$$D_{\pi'} = D_\pi + \Delta_D \quad , \Delta_D > 0$$

$$N_{\pi'} = N_\pi + \Delta_N \quad , \Delta_N > 0$$

$$\frac{N_\pi}{D_\pi} - \frac{N_{\pi'}}{D_{\pi'}} = \frac{N_\pi}{D_\pi} - \frac{(N_\pi + \Delta_N)}{(D_\pi + \Delta_D)}$$

$$
\begin{aligned}
&= \frac{N_\pi D_\pi + N_\pi \Delta_D - N_\pi D_\pi - D_\pi \Delta_N}{D_\pi (D_\pi + \Delta_D)} \\
&= \frac{N_\pi \Delta_D - D_\pi \Delta_N}{D_\pi (D_\pi + \Delta_D)} \qquad (93)
\end{aligned}
$$

(a) $C'_{total} > C_{total}$ associated with 4.3.2.A.2.

$$
\begin{aligned}
N_\pi \Delta_D - D_\pi \Delta_N &< 0 \\
\frac{N_\pi}{D_\pi} &< \frac{\Delta_N}{\Delta_D} \\
&< \frac{(N_{\pi'} - N_\pi)}{(D_{\pi'} - D_\pi)} \qquad (94)
\end{aligned}
$$

(b) $C'_{total} = C_{total}$ associated with 4.3.2.B.1.

$$
\begin{aligned}
N_\pi \Delta_D - D_\pi \Delta_N &= 0 \\
\frac{N_\pi}{D_\pi} &= \frac{\Delta_N}{\Delta_D} \\
&= \frac{(N_{\pi'} - N_\pi)}{(D_{\pi'} - D_\pi)} \qquad (95)
\end{aligned}
$$

(c) $C'_{total} < C_{total}$ associated with 4.3.2.C.4.

$$
\begin{aligned}
N_\pi \Delta_D - D_\pi \Delta_N &> 0 \\
\frac{N_\pi}{D_\pi} &> \frac{\Delta_N}{\Delta_D} \\
&> \frac{(N_{\pi'} - N_\pi)}{(D_{\pi'} - D_\pi)} \qquad (96)
\end{aligned}
$$

2. $N_{\pi'} < N_\pi$, $D_{\pi'} < D_\pi$

Let

$$\begin{aligned} D_\pi &= D_{\pi'} + \Delta_D \quad , \Delta_D > 0 \\ N_\pi &= N_{\pi'} + \Delta_N \quad , \Delta_N > 0 \end{aligned}$$

$$\begin{aligned} \frac{N_\pi}{D_\pi} - \frac{N_{\pi'}}{D_{\pi'}} &= \frac{(N_{\pi'} + \Delta_N)}{(D_{\pi'} + \Delta_D)} - \frac{N_{\pi'}}{D_{\pi'}} \\ &= \frac{N_{\pi'}D_{\pi'} + D_{\pi'}\Delta_N - N_{\pi'}D_{\pi'} - N_{\pi'}\Delta_D}{D_{\pi'}(D_{\pi'} + \Delta_D)} \\ &= \frac{D_{\pi'}\Delta_N - N_{\pi'}\Delta_D}{D_{\pi'}(D_{\pi'} + \Delta_D)} \end{aligned} \tag{97}$$

(a) $C'_{total} > C_{total}$ associated with 4.3.2.A.4.

$$\begin{aligned} D_{\pi'}\Delta_N - N_{\pi'}\Delta_D &< 0 \\ \frac{N_{\pi'}}{D_{\pi'}} &> \frac{\Delta_N}{\Delta_D} \\ &> \frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})} \end{aligned} \tag{98}$$

(b) $C'_{total} = C_{total}$ associated with 4.3.2.B.2

$$\begin{aligned} D_{\pi'}\Delta_N - N_{\pi'}\Delta_D &= 0 \\ \frac{N_{\pi'}}{D_{\pi'}} &= \frac{\Delta_N}{\Delta_D} \\ &= \frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})} \end{aligned} \tag{99}$$

(c) $C'_{total} < C_{total}$ associated with 4.3.2.C.2.

$$\begin{aligned} D_{\pi'}\Delta_N - N_{\pi'}\Delta_D &> 0 \\ \frac{N_{\pi'}}{D_{\pi'}} &< \frac{\Delta_N}{\Delta_D} \\ &< \frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})} \end{aligned} \tag{100}$$

Therefore, in the uncertain area, one has:

1. $D_{\pi'} > D_\pi$ or $(z_{j+1} - z_j)(w_j - w_{j+1}) < 0$

   (a) $C'_{total} > C_{total}$, if

$$C_{total} = \frac{N_\pi}{D_\pi} < \frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})} \tag{101}$$

   (b) $C'_{total} = C_{total}$, if

$$C_{total} = \frac{N_\pi}{D_\pi} = \frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})} \tag{102}$$

   (c) $C'_{total} < C_{total}$, if

$$C_{total} = \frac{N_\pi}{D_\pi} > \frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})} \tag{103}$$

2. $D_{\pi'} < D_\pi$ or $(z_{j+1} - z_j)(w_j - w_{j+1}) > 0$

   (a) $C'_{total} > C_{total}$, if

$$C'_{total} = \frac{N_{\pi'}}{D_{\pi'}} > \frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})} \tag{104}$$

(b) $C'_{total} = C_{total}$, if

$$C'_{total} = \frac{N_{\pi'}}{D_{\pi'}} = \frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})} \tag{105}$$

(c) $C'_{total} < C_{total}$, if

$$C'_{total} = \frac{N_{\pi'}}{D_{\pi'}} < \frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})} \tag{106}$$

From equation (42) and equation (43), one has:

$$\begin{aligned} \frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})} &= \frac{aC_0 + \sum_{n=1}^{j-1} k_n C_n}{a + \sum_{n=1}^{j-1} k_n} + \frac{d(z_{j+1}c^l_{j+1} - z_j c^l_j)}{(z_{j+1} - z_j)(a + \sum_{n=1}^{j-1} k_n)} \\ &\quad + \frac{d(w_j c^p_j - w_{j+1}c^p_{j+1})z_{j+1}}{(z_{j+1} - z_j)(w_j - w_{j+1})(a + \sum_{n=1}^{j-1} k_n)} \end{aligned} \tag{107}$$

4.3.4 The Resulting Theorems

Theorem 1 *In a single-level tree network, if one of the following conditions is satisfied, then the total cost of the current processor arrangement* $C_{total}(\pi_{(p_1,\ldots,\underline{p_j,p_{j+1}},\ldots,p_N)})$ *is less than the total cost of the adjacent pairwise swapped processor arrangement* $C'_{total}(\pi'_{(p_1,\ldots,\underline{p_{j+1},p_j},\ldots,p_N)})$ *for* $1 \leq j < N$.

1). $(z_{j+1} - z_j)(w_j - w_{j+1}) = 0$ *and*

$(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) < z_{j+1}(w_{j+1}c^p_{j+1} - w_j c^p_j)$

2). $(z_{j+1} - z_j)(w_j - w_{j+1}) \neq 0$ *and*

$$(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) \leq z_{j+1}(w_{j+1}c^p_{j+1} - w_j c^p_j) - \Delta$$

3). $(z_{j+1} - z_j)(w_j - w_{j+1}) < 0$ *and* $C_{total} < \frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})}$

4). $(z_{j+1} - z_j)(w_j - w_{j+1}) > 0$ *and* $C'_{total} > \frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})}$

Proof

The first condition results from equation (78).

The second condition results from equation (82) and equation (84).

The third condition results from equation (101).

The fourth condition results from equation (104).

Thus the theorem is proved. □

Theorem 2 *In a single-level tree network, if one of the following conditions is satisfied, then the total cost of the current processor arrangement* $C_{total}(\pi_{(p_1,\ldots,\underline{p_j,p_{j+1}},\ldots,p_N)})$ *is equal to the total cost of the adjacent pairwise swapped processor arrangement* $C'_{total}(\pi'_{(p_1,\ldots,\underline{p_{j+1},p_j},\ldots,p_N)})$ *for* $1 \leq j < N$.

1). $(z_{j+1} - z_j)(w_j - w_{j+1}) = 0$ *and*

$$(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) = z_{j+1}(w_{j+1}c^p_{j+1} - w_j c^p_j)$$

2). $(z_{j+1} - z_j)(w_j - w_{j+1}) \neq 0$ *and* $C_{total} = \frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})}$

Proof

The first condition results from equation (79).

The second condition results from equation (102) and equation (105).

Thus the theorem is proved. □

Theorem 3 *In a single-level tree network, if one of the following conditions is satisfied, then the total cost of the current processor arrangement* $C_{total}(\pi_{(p_1,\ldots,\underline{p_j,p_{j+1}},\ldots,p_N)})$ *is greater than the total cost of the adjacent pairwise swapped processor arrangement* $C'_{total}(\pi'_{(p_1,\ldots,\underline{p_{j+1},p_j},\ldots,p_N)})$ *for* $1 \leq j < N$.

1). $(z_{j+1} - z_j)(w_j - w_{j+1}) = 0$ *and*

$(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) > z_{j+1}(w_{j+1}c^p_{j+1} - w_j c^p_j)$

2). $(z_{j+1} - z_j)(w_j - w_{j+1}) \neq 0$ *and*

$(w_j - w_{j+1})(z_{j+1}c^l_{j+1} - z_j c^l_j) \geq z_{j+1}(w_{j+1}c^p_{j+1} - w_j c^p_j) + \Delta$

3). $(z_{j+1} - z_j)(w_j - w_{j+1}) < 0$ *and* $C_{total} > \frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})}$

4). $(z_{j+1} - z_j)(w_j - w_{j+1}) > 0$ *and* $C'_{total} < \frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})}$

Proof

The first condition results from equation (80).

The second condition results from equation (88) and equation (90).

The third condition results from equation (103).

The fourth condition results from equation (106).

Thus the theorem is proved. □

Here:

$$\Delta = \frac{1}{d}\left\{|(z_j - z_{j+1})(w_j - w_{j+1})|\left[aC_0 + \sum_{n=1}^{j-1} k_n C_n\right]\right\}$$

$$\frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})} = \frac{aC_0 + \sum_{n=1}^{j-1} k_n C_n}{a + \sum_{n=1}^{j-1} k_n} + \frac{d(z_{j+1}c_{j+1}^l - z_j c_j^l)}{(z_{j+1} - z_j)(a + \sum_{n=1}^{j-1} k_n)} + \frac{d(w_j c_j^p - w_{j+1}c_{j+1}^p)z_{j+1}}{(z_{j+1} - z_j)(w_j - w_{j+1})(a + \sum_{n=1}^{j-1} k_n)}$$

5 Bus and Related Networks

5.1 Bus Network

A bus network is a special case of a single-level tree network where all link speeds and link costs are equal. The following lemma and theorem can be obtained:

Lemma 6 *In a bus network, the total cost of a current processor arrangement, C_{total}, is less than or equal to (greater than) that of an associated swapped processor arrangement, C'_{total}, if, for the current processor arrangement,*

$$c_j^p w_j \leq (>) c_{j+1}^p w_{j+1}$$

Proof

In case of a bus network one has $z_j = z_{j+1}$ and $z_j c_j^l = z_{j+1} c_{j+1}^l$, therefore $(z_{j+1} - z_j)(w_j - w_{j+1}) = 0$. For the case $C'_{total} > C_{total}$, from condition 1 of theorem 1,2 one has,

$$\begin{aligned} z_{j+1}(c_{j+1}^p w_{j+1} - c_j^p w_j) &\geq 0 \\ c_j^p w_j &\leq c_{j+1}^p w_{j+1} \end{aligned}$$

A similar result holds for the case of $C_{total} > C'_{total}$ if one uses condition 1 of theorem 3. The lemma is thus proved. □

Lemma 7 *In a bus network, if the processors are arranged such that for every adjacent pair of processors the condition $c_j^p w_j \leq c_{j+1}^p w_{j+1}$ is satisfied, then there is no other processor arrangement profile with a lower total cost.*

Proof

By contradiction, assume that an underlying processor arrangement profile π is arranged such that for every adjacent pair the condition $c_j^p w_j \leq c_{j+1}^p w_{j+1}$ holds, and there exists another processor arrangement profile π', a permutation of π, that has at least one adjacent pair of processors with a condition $c_j^p w_j > c_{j+1}^p w_{j+1}$ which gives a lower total cost than π. By lemma 6, a total cost of π' can be decreased by swapping a pair of processors with the condition $c_j^p w_j > c_{j+1}^p w_{j+1}$. Lemma 6 can then be applied recursively to the resulting processor arrangement profile as far as it has such an adjacent pair of processors with the condition $c_j^p w_j > c_{j+1}^p w_{j+1}$. The total cost of the current processor arrangement profile is decreased each time lemma 6 is applied. Finally, π is reached from π' through a series of lemma 6 applications. Therefore a total cost of π is lower than that of π'. This contradicts the hypothesis that π' gives a lower total cost that π. The lemma is thus proved. □

Theorem 4 *In a bus network, the total cost, C_{total}, is minimized over all processor arrangement profiles if and only if the processors are arranged to satisfy the following condition:*

$$c_1^p w_1 \leq c_2^p w_2 \leq ... \leq c_N^p w_N$$

Proof

Assume that $c_j^p w_j$ are not all identical.

The "only if" part ($\Rightarrow$): By contradiction, suppose that the processors are arranged in such a way that C_{total} is minimized over all processor arrangement profiles and there exists at least one adjacent pair of processors in that arrangement such that $c_j^p w_j > c_{j+1}^p w_{j+1}$. Then by lemma 6 there exists another processor arrangement with a lower total cost by swapping processor j and $(j+1)$. This contradicts the hypothesis that C_{total} is minimal. Therefore the ($\Rightarrow$) is proved.

The "if" part ($\Leftarrow$): If the processors are arranged such that for every adjacent pair $c_j^p w_j \leq c_{j+1}^p w_{j+1}$, then by lemma 7 there is no other processor arrangement that can lower the total cost further. Thus, C_{total} of such an arrangement is indeed minimal.

The theorem is then proved. □

5.2 The Related Networks

In this subsection, a single-level tree network with identical processors and a homogeneous single-level tree network are investigated.

Lemma 8 *In a single-level tree network where all processors have the same computing speeds, the total cost of a current processor arrangement, C_{total}, is less than or equal to (greater than) that of an associated swapped processor arrangement, C'_{total}, if, for the current processor arrangement,*

$$c_j^p \leq (>) c_{j+1}^p$$

Proof

Since $w_j = w_{j+1}$, therefore $(z_{j+1} - z_j)(w_j - w_{j+1}) = 0$. For the case $C'_{total} > C_{total}$, from condition 1 of theorem 1,2 one has,

$$\begin{aligned} z_{j+1}(c_{j+1}^p w_{j+1} - c_j^p w_j) &\geq 0 \\ c_j^p &\leq c_{j+1}^p \end{aligned}$$

Similarly for the case of $C_{total} > C'_{total}$ if one uses condition 1 of theorem 3. The lemma is thus proved. □

Lemma 9 *In a single-level tree network where all processors have the same computing speeds, if the processors are arranged such that for every adjacent pair of processors the condition $c_j^p \leq c_{j+1}^p$ is satisfied, then there is no other processor arrangement profile with a lower total cost.*

Proof

By contradiction, assume that an underlying processor arrangement profile π is arranged such that for every adjacent pair the condition $c_j^p \leq c_{j+1}^p$ holds, and there exists another processor arrangement profile π', a permutation of π, that has at least one adjacent pair of processors with a condition $c_j^p > c_{j+1}^p$ which gives a lower total cost than π. By lemma 8, a total cost of π' can be decreased by swapping a pair of processors with the condition $c_j^p > c_{j+1}^p$. Lemma 8 can then be applied recursively to the resulting processor arrangement profile as far as it has such an adjacent pair of processors with the condition $c_j^p > c_{j+1}^p$. The total cost of the current processor arrangement profile is decreased each time lemma 8 is applied. Finally, π is reached from π' through a series of lemma 8 applications. Therefore a total cost of π is lower than that of π'. This contradicts the hypothesis that π' gives a lower total cost that π. The lemma is thus proved. □

Theorem 5 *In a single-level tree network where all processors have the same computing speeds, the total cost, C_{total}, is minimized over all processor arrangement profiles if and only if the processors are arranged to satisfy the following condition:*

$$c_1^p \leq c_2^p \leq \ldots \leq c_N^p$$

Proof

Assume that c_j^p are not all identical.

The "only if" part ($\Rightarrow$): By contradiction, suppose that the processors are arranged in such a way that C_{total} is minimized over all processor arrangement profiles and there exists at least one adjacent pair of processors in that arrangement such that $c_j^p > c_{j+1}^p$. Then by lemma 8 there exists another processor arrangement with a lower total cost by swapping processor j and $(j+1)$. This contradicts the hypothesis that C_{total} is minimal. Therefore the $(\Rightarrow)$ is proven.

The "if" part ($\Leftarrow$): If the processors are arranged such that for every adjacent pair $c_j^p \leq c_{j+1}^p$, then by lemma 9 there is no other processor arrangement which can lower the total cost further. Thus, C_{total} of such an arrangement is indeed minimal.

The theorem is thus proved. □

Lemma 10 *In a homogeneous single-level tree network where all the link speeds and the processor speeds are the same, the total cost of a current processor arrangement, C_{total}, is less than or equal to (greater than) that of an associated swapped processor arrangement, C'_{total}, if, for the current*

processor arrangement,

$$c_j^p \leq (>) c_{j+1}^p$$

Proof

Since $w_j = w_{j+1}$ and $z_j = z_{j+1}$, therefore $(z_{j+1} - z_j)(w_j - w_{j+1}) = 0$. For the case $C'_{total} > C_{total}$, from condition 1 of theorem 1,2 one has,

$$\begin{aligned} z_{j+1}(c_{j+1}^p w_{j+1} - c_j^p w_j) &\geq 0 \\ c_j^p &\leq c_{j+1}^p \end{aligned}$$

Similarly for the case of $C_{total} > C'_{total}$ if one uses condition 1 of theorem 3. The lemma is thus proved. □

Lemma 11 *In a homogeneous single-level tree network where all the link speeds and the processor speeds are the same, if the processors are arranged such that for every adjacent pair of processors the condition* $c_j^p \leq c_{j+1}^p$ *is satisfied, then there is no other processor arrangement profile with a lower total cost.*

Proof

The proof is similar to the one in lemma 9 except one uses lemma 10 instead of lemma 8. □

Theorem 6 *In a homogeneous single-level tree network, where all the link and processor speeds are the same, the total cost, C_{total}, is minimized over all processor arrangement profiles if and only if the processors are arranged to satisfy the following condition:*

$$c_1^p \leq c_2^p \leq ... \leq c_N^p$$

Proof

This is a special case of theorem 5. The proof is similar to the one for theorem 5 except one uses lemma 10 instead of lemma 8 and lemma 11 instead of lemma 9. □

6 Analysis Remarks

It should be noted that in a general single-level tree network, as is evident from theorem 1-3 there is no simple condition to check for an optimal total cost processor arrangement. This is due to the fact that there are several interrelated conditions when C_{total} is compared to C'_{total}. There may exist two processor arrangements with all adjacent pair conditions compliant to theorem 1 or 2 such that one cannot simply tell which arrangement yields a lower total cost or whether they both are optimal. However, if special

cases of a single-level tree network are considered as shown in section 5, simple optimal conditions for a processor arrangement can be established as indicated in theorem 4, 5, and 6.

The analysis of a processor arrangement in a general single-level tree network, even though it may not provide a simple check for optimality, sheds light on the derivation of the conditions for the optimal processor arrangement in the special case networks. In addition, it is useful, through the use of theorem 1-3, to identify an adjacent processor pair that could be rearranged to reduce total cost in the process of searching for the optimal processor arrangement in a general single-level tree network. This makes the cost efficient processor arrangement algorithm to be discussed appealing in terms of its simplicity, complexity, feasibility, and monotonic improvement of its solutions.

In the case of a bus network as indicated by theorem 4, C_{total} is minimized over all processor arrangement profiles if and only if the processor costs, $c_i^p w_i$, are ordered in a non-decreasing maner. This is exactly the optimality condition found in [8]. Therein the authors use a sequencing with an adjacent pairwise swapping to find a sequence of load distribution with a minimum total cost. This is because in a bus network where all link speeds

are identical and link costs have the same values, to do sequencing, i.e. a logical interchange of the processors, is equivalent to the problem of processor arrangement, a physical interchange of the processors, of this paper.

For a single-level tree network where all processors have the same computing speeds, C_{total} is minimized over all processor arrangement profiles when the processor cost coefficients, c_i^p, are ordered in a non-decreasing maner. This can be explained as follows. The root processor always distributes fractions of load through $l_1, l_2, ..., l_N$ sequentially in this order no matter which processor is attached to which link. Since $w_j = w_{j+1}$ for all $j = 1$ to $N-1$, the closed-form expression of α_i is indepedent of the processor arrangement; consequently the communication time is unchanged due to processor arrangements as well. Moreover $\alpha_1 > \alpha_2 > ... > \alpha_N$. Note that in this case $\frac{\alpha_i}{\alpha_{i+1}} = \frac{(z_{i+1}T_{cm}+wT_{cp})}{wT_{cp}} > 1$. In order to minimize the total cost the processor with the lowest processor cost coefficient should receive the largest fraction of load (α_1) and the next lowest processor cost coefficient processor receive the next largest fraction of load (α_2) and so on.

Similarly for the case of a homogeneous single-level tree network which is a special case of the single-level tree where all of processors have the same speed, the total cost, C_{total}, is minimized over all processor arrangement pro-

files when the processor cost coefficients, $c_i^p{}'s$, are ordered in a non-decreasing maner.

Note that in our analysis, the processor arrangement does not involve the root processor. It consists only of an arrangement of the children processors.

7 Heuristic Processor Arrangement Algorithm

In this section, a discussion of the performance of two basic greedy algorithms for processor arrangement is presented. A heuristic algorithm for processor arrangement, which is a modified version of the two basic algorithms, is proposed so as to improve the performance. This algorithm uses several starting points in order to produce a final solution. The mechanism to generate a new processor arrangement profile, π', is no longer restricted to an adjacent pairwise processor swapping as in the case of sequencing of [7]. It extends to cover the cases of swapping two processors which are two positions apart, three positions apart and so on. Finally an exhaustive permutation algorithm is presented which serves as a tool to obtain globally optimal solutions for testing purposes.

7.1 Two Basic Greedy Processor Arrangement Algorithms

Both of these basic algorithms are based on a nearest neighbor search which repeatedly improves the current solution until no further improvement can be made. The first algorithm is the theorem-based greedy algorithm. This algorithm is based on the theorems developed in the previous section to identify which adjacent pair of processors could be swapped to reduce total cost. A greedy strategy is then applied to select among the candidate adjacent pairs of processors an actual adjacent pair to swap in order to obtain the best improvement in terms of total cost. The second algorithm is the direct cost greedy algorithm. This algorithm computes the total cost associated with each adjacent pairwise processor swapped profile, π', and the current profile, π, directly. It then uses a greedy strategy to select the profile with the lowest total cost. This direct cost greedy algorithm can be considered as a variant, but more general version, of the theorem-based greedy algorithm. However they are equivalent in terms of the final solution convergence. Experimental results using the direct cost greedy algorithm show that the number of suboptimal convergence solutions are quite large (cf.table 1). This indicates

that both algorithms may not be effective enough and need to be modified to improve the performance. This therefore leads to a heuristic algorithm.

7.2 Heuristic Method

In order to improve the performance of a processor arrangement algorithm four main approaches are introduced. First, several initial processor arrangements, which serve as starting points of the heuristic algorithm, are generated according to patterns determined by the orderings of some network parameters. Secondly, the extent of a neighborhood to be search is enlarged to cover not only a neighborhood obtained by adjacent processor pairwise swapping. Thirdly, an exhaustive search over an entire neighborhood is employed. A greedy strategy is applied to select the best processor arrangement profile in terms of a total processing cost. Finally, restart searching, the procedure determining a pair of processors to start over with after a new processor profile has been obtained, is used as the order of searching. Integrating all of these approaches constitutes the proposed heuristic algorithm.

Table 1: Number network parameter sets with local optimal solutions. The number of randomly generated network parameter sets per run is 100

Run	No.of suboptimal solutions
1	15
2	13
3	13
4	18
5	9
6	9
7	10
8	8
9	10
10	9

7.3 Elements of the Heuristic Algorithm

7.3.1 The Starting Points

In this heuristic algorithm, there are a number of starting points (set to 19 in this work). Each starting point is obtained by ordering the randomly generated network parameters including w_i, z_i, c_i^l, c_i^p. The *order* of w_0, z_i, c_i^l are fixed for every starting point according to the original order resulting from the random generation procedure. That is the root node, link speeds, and link cost coefficients are kept in their respective orders while w_i and c_i^p are rearranged according to some specific patterns. These patterns are:

1. Non-decreasing and non-increasing orderings of c_i^p.
2. Non-decreasing and non-increasing orderings of c_i^l.
3. Non-decreasing and non-increasing orderings of z_i.
4. Non-decreasing and non-increasing orderings of w_i.
5. Non-decreasing and non-increasing orderings of $c_i^l z_i T_{cm} + c_i^p w_i T_{cp}$.
6. Non-decreasing and non-increasing orderings of $c_i^p w_i T_{cp}$.

7. Non-decreasing and non-increasing orderings of the sum of the positions in the rank of c_i^l and c_i^p.

8. Non-decreasing and non-increasing orderings of the sum of the positions in the rank of c_i^p and $c_i^p w_i T_{cp}$.

9. Non-decreasing and non-increasing orderings of the sum of the positions in the rank of c_i^l and $c_i^l z_i T_{cm} + c_i^p w_i T_{cp}$.

10. The original ordering from the randomly generating procedure.

Note that a rank is a list of all processors where the order in the list is determined by their corresponding parameter values, i.e., c_i^l, c_i^p, $(c_i^p w_i T_{cp})$, or $(c_i^l z_i T_{cm} + c_i^p w_i T_{cp})$. All the processors in a rank are ordered in a non-decreasing maner of their pertinent parameter values as given above.

7.3.2 The Nmove Procedure

This is a procedure to obtain a solution of each starting point which is called an intermediate solution of the algorithm. It consists of four main aspects as follows:

1. N-Position Apart Processor Swapping

N-position apart swapping refers to an interchange of a pair of processors which are apart by N positions and leaves other processors in their respective positions. For example, an adjacent pair of processors is considered to be 1- position apart; a pair of processors which has one processor between them is called a 2-position apart pair of processors and so on. The implementation of any n-position apart processor swapping always starts from swapping the processor in the first (say leftmost) position of the underlying processor arrangement profile with the corresponding processor and then moves to the processor in the second position and so on.

2. Neighborhood Generating Function

This subprocedure generates different levels of the neighborhood of the current processor arrangement profile by starting from swapping an adjacent pair of processors which gives $(N-1)$ neighbors. Next, if required by the algorithm, it swaps a pair of processors which are two positions apart which gives $(N-2)$ new distinct neighbors. The subprocedure continues, if necessary, until a pair of processors with $(N-1)$ positions apart is swapped which produces one new neighbor.

3. Greedy Strategy

   In order to obtain a new processor arrangement profile, a greedy strategy is used to select the profile with the lowest total cost from the underlying level of the neighborhood set which is currently searched. If there is no neighbor which has a lower total cost than the current one, then the next level of neighborhood set will be searched.

4. Restart Searching Strategy

   The order of searching used in this procedure is a "restart searching". That is, once a new processor arrangement profile is obtained and adopted as a new current processor arrangement profile, the procedure searches the neighborhood of this new current profile by starting in the 1-position apart mode of operation from the leftmost position.

7.3.3 Nmove2 Procedure

The Nmove2 subprocedure is similar to the Nmove subprocedure described above except that it starts by swapping a pair of processors which are two positions apart, then three positions apart and so on. It continues until a pair of processors which are $(N-1)$ positions apart is swapped then moves

to swap an adjacent pair of processors, i.e., the 1-position apart pair of processors, is the final level of generated neighborhood. The restart searching of the neighborhood of the new profile is from the 2-position apart mode of operation.

7.3.4 The Intermediate Solutions

The intermediate solution is a processor arrangement profile obtained from the heuristic algorithm when the algorithm is initialized with a specific starting point. Therefore, in this work, the intermediate solutions result from using the starting points according to the patterns given in 1-9 in subsection 7.3.1 with the Nmove procedure, and the original ordering as stated in 10 in subsection 7.3.1 with the Nmove2 procedure. Totally there are 19 intermediate solutions. It should be noted here that the intermediate solution is in fact a local optimal solution.

7.3.5 The Final Solution

The final solution is obtained by comparing the intermediate solutions from the Nmove and Nmove2 procedures which are initialized with different starting points. The one with the lowest total cost will be a final solution to the

algorithm.

7.4 The Heuristic Algorithm Description

The algorithm is initialized by generating 19 starting points. A certain set of orderings determined by $w_i, z_i, c_i^p, c_i^l, c_i^p T_{cp}$ and $c_i^l T_{cm}$ as mentioned in subsection 7.3.1 are formed. These orderings are then used to produce certain ordering patterns, the patterns of processor index orderings. From these ordering patterns, the starting points are obtained through rearranging the order of the original processors according to the corresponding patterns while keeping all links in their original order. In this rearrangement process the root processor is not taken into consideration.

For each starting point according to 1-9 in subsection 7.3.1, the Nmove procedure is called to find the minimum total cost processor arrangement profile, which is an intermediate solution. The procedure begins searching for a minimum total cost processor arrangement profile by first examining the 1-position apart neighbors to check if there is any profile with a lower total cost than the current profile. If there is no such a profile, the Nmove procedure then moves to search the next level of neighbor which is the 2-position

apart and checks for a lower total cost profile. The procedure continues searching until it finds a lower total cost profile or otherwise terminates if it can not find such a profile after searching all levels of the neighborhood of the current processor arrangement profile. In this last case the current processor arrangement profile is the solution. If the procedure finds a lower total cost profile before terminating then that processor arrangement profile will be adopted as the new processor arrangement profile and the process starts over again according to the restart searching strategy. In case that there is more than one profile with a lower total cost, the greedy strategy is applied and the lowest total cost profile among them will be chosen as a new current profile. For an original order starting point from randomly generating procedure with the Nmove2 procedure which is according to 10 in subsection 7.2.1, the process is similar to that described above except that the Nmove2 procedure is called instead of the Nmove procedure.

The algorithm continues finding a solution for each starting point, i.e., an intermediate solution, until all of them have been processed through the Nmove or Nmove2 procedure accordingly. Finally, the algorithm compares all the intermediate solutions obtained from the Nmove and Nmove2 procedures to reach the final solution. The solution whose C_{total} is less than or equal to

that of the others is chosen to be the final solution.

7.5 The Heuristic Processor Arrangement Algorithm

Given an arbitrary single level tree network and z_i, w_i, c_i^l, c_i^p, T_{cm}, and T_{cp}.

Given an initial processor arrangement profile.

Step 0. Set an initial processor arrangement profile to be $\pi^H(0)$.

Set N to 19.

Step 1. Compute an associated C_{total} of $\pi^H(0)$, set it to $C_{int}^H(0)$.

Step 2. For $j = 0$ to 19

2.1) Call Nmove or Nmove2.

2.2) Set the intermediate solution obtained in (2.1) to $\pi_{int}^H(j)$.

Compute the associated C_{total}, set it to $C_{int}^H(j)$.

2.3) $j = j + 1$

Step 3. For $k = 0$ to 19

Compare $C_{int}^H(k)$, find the index, k, with the lowest $C_{int}^H(k)$.

Step 4. Output $\pi^H(k)$ as the final processor arrangement profile and $C_{int}^H(k)$ as its associated total cost. Stop.

7.6 Exhaustive Permutation Algorithm

The main function of the exhaustive permutation algorithm is to find the global minimum total cost processor arrangement profile and the corresponding values by exhaustively examining all possible processor arrangement profiles. The algorithm computes a total cost directly from the closed-form expression of the total cost. It then moves to the next processor arrangement profile by permuting the current order of the children processors in a lexicograhpic maner while keeping the original arrangement of links. It continues to generate the permutation and compute an associated total cost until all possible permutations are generated. Finally, all obtained total costs are compared to find the minimal one(s).

8 Performance Evaluation

In this section, two prime performance aspects of the proposed heuristic algorithm are studied, i.e., the quality of solutions and the time complexity.

8.1 Quality of Solutions

In this work, the quality of solutions is measured by two criteria, the ability to achieve an optimal solution and the proximity of the final solutions to the optimal solution. In terms of an ability to achieve an optimal solution, the performance metric in this case is a number of final solutions which are suboptimal. For the proximity of the final solutions to the optimal solution, we consider in two cases, i.e., the averge case and the worst case. There are two metrics associated with each case, i.e., the ratio and the relative difference.

8.1.1 Performance Metrics

Let

I : the network instance.

$C_H(I)$: the total cost of the final solution of the network instance I obtained by the proposed heuristic algorithm.

$C_{OPT}(I)$: the minimum total cost of the network instance I.

$|I|$: the cardinality of the set of network instances in the experiment.

Define:

A number of suboptimal solutions = A number of final solutions with total cost higher than that of an optimal solution.

$$\text{The average ratio,} \quad R_{AV} = \frac{\sum_{\forall I}\left(\frac{C_{OPT}(I)}{C_H(I)}\right)}{|I|} \tag{108}$$

$$\text{The worst-case ratio,} \quad R_{WST} = \min_{\forall I}\left(\frac{C_{OPT}(I)}{C_H(I)}\right) \tag{109}$$

$$\text{The average relative difference,} \quad \Delta_{AV} = \frac{\sum_{\forall I}(\frac{C_H(I)-C_{OPT}(I)}{C_{OPT}(I)})}{|I|} \tag{110}$$

$$\text{The worst-case relative difference,} \quad \Delta_{WST} = \max_{\forall I}(\frac{C_H(I)-C_{OPT}(I)}{C_{OPT}(I)}) \tag{111}$$

8.1.2 Experimental Procedure

Two experiment sets were conducted on single-level tree networks so as to evaluate the quality of the solutions produced by the proposed heuristic algorithm. The general procedure for the two experiment sets are as follows. The network parameters, $z_j, w_j, c_j^l, c_j^p, T_{cm}, T_{cp}$ are generated uniformly and independently in the interval [0,10] for z_j, w_j, [0,20] for c_j^l, c_j^p, and [0,5] for

T_{cm}, T_{cp}. In each experiment set, it consists of two subexperiments. That is, a subexperiment conducted on single-level tree network where $z_i T_{cm} \geq w_i T_{cp}$ and the other with $z_i T_{cm} \leq w_i T_{cp}$ for all $i = 1$ to N (N is a number of children processors).

The first experiment set was performed on single-level tree networks with one root processor and five children processors ($N = 5$). The total number of run is 10 and for each run 20000 network parameters were generated for each subexperiment set.

The second experiment set, a number of children processors were varied from 5 to 10. For $N = 5$ to 8, 20000 network parameters were generated, and for $N = 9$ to 10, 10000 network parameters were generated.

8.1.3 Results and Discussion

The first experimental results (cf.table 2,3) show that the probability of converging to a suboptimal solution is extremly small in both subexperiments, based on a network parameters are generated randomly and uniformly on the specified intervals. In addition, the quality of the suboptimal solutions is also impressive, that is the closeness of the suboptimal solutions to the optimal solution is evident from the experimental results both in the average

Table 2: $z_i T_{cm} \leq w_i Tcp$

Run	No.of suboptimal solutions	Average Case		Worst-Case	
		R_{AV}	Δ_{AV}	R_{WST}	Δ_{WST}
1	0	NA	NA	NA	NA
2	0	NA	NA	NA	NA
3	1	99.963	0.0366	99.963	0.0366
4	0	NA	NA	NA	NA
5	0	NA	NA	NA	NA
6	0	NA	NA	NA	NA
7	1	99.573	0.4292	99.573	0.4292
8	1	99.628	0.3733	99.628	0.3733
9	0	NA	NA	NA	NA
10	0	NA	NA	NA	NA

Table 3: $z_i T_{cm} \geq w_i Tcp$

Run	No.of suboptimal solutions	Average Case		Worst-Case	
		R_{AV}	Δ_{AV}	R_{WST}	Δ_{WST}
1	0	NA	NA	NA	NA
2	0	NA	NA	NA	NA
3	0	NA	NA	NA	NA
4	1	99.838	0.1627	99.838	0.1627
5	1	99.147	0.8605	99.147	0.8605
6	1	99.883	0.1175	99.883	0.1175
7	1	99.841	0.1597	99.841	0.1597
8	1	99.367	0.6372	99.367	0.6372
9	1	95.689	4.5051	95.689	4.5051
10	2	99.827	0.1732	99.786	0.2147

Table 4: $z_i T_{cm} \leq w_i Tcp$

No. of children processors	No.of suboptimal solutions	Average Case		Worst-Case	
		R_{AV}	Δ_{AV}	R_{WST}	Δ_{WST}
6	1	99.936	0.0582	99.936	0.0582
7	8	99.669	0.3333	99.757	1.2585
8	7	99.929	0.0713	99.626	0.3751
9*	9	99.925	0.0747	99.774	0.2263

Table 5: $z_i T_{cm} \geq w_i Tcp$

No. of children processors	No.of suboptimal solutions	Average Case		Worst-Case	
		R_{AV}	Δ_{AV}	R_{WST}	Δ_{WST}
6	1	99.816	0.1841	99.816	0.1841
7	4	99.686	0.3169	98.889	1.1237
8	10	99.988	0.0116	99.966	0.0342
9*	7	99.980	0.0198	99.919	0.0813

case and the worst case.

The second experimental results (cf. table 4,5) also show the effectiveness of the heuristic algorithm when the number of children processors was increased. The quality of the suboptimal solutions both in terms of average case and worst case is very impressive. From the results, the proximity of the solutions to the optimal solution does not deteriorate with the number of children processors.

It can be seen from the experimental results that the proposed heuristic algorithm converges to the optimal solution with high probability. Moreover, the experimental results indicate the goodness of the suboptimal solutions obtained by the algorithm in terms of the proximity of the solutions to the optimal solution in both the average case and the worst case.

8.2 Efficiency

In this section, the efficiency of the proposed heuristic algorithm in terms of computation time performance metric is studied through a series of experiments. The average number of iterations needed to search the neighborhoods before an intermediate solution (a local optimum) was obtained was mea-

sured. The experiments involved running the algorithm on 5000 randomly generated single-level tree networks of each network size, with the number of children processors ranging from 5 to 20. The network parameters were generated in the same way as in the previous section. The expected number of iterations needed in the Nmove or Nmove2 procedure for each starting point was then computed. Efficiency is defined as the total number of iterations needed from all the starting points to obtained a final solution. In order to evaluate efficiency, the average number of iterations to obtain the intermediate solution was calculated when the proposed heuristic algorithm was initialized with each starting point. The efficiency is thus proportional to the number of starting points times the average number of iterations.

The experimental results are shown in table 6. From table 6, the average number of iterations needed in the range used can be asymptotically bounded by $O(N^{1.6})$, where N is a number of children processors. Consequently an efficiency as a total number of iterations needed can also be asymptotically bounded by $O(N^{1.6})$. That is an efficiency is bounded by a low-order polynimial in N which is consistent with the results in [9].

It should be noted here that for a local search algorithm to solve a combinatorial optimization, the worst-case time complexity, an upper bound on

Table 6: The average number of iterations and its corresponding asymptotic bounds

No.of pairs	Average	$N^{1.7}$	$N^{1.65}$	$N^{1.6}$	$N^{1.5}$
5	4.8	15.4	14.2	13.1	11.2
6	7.2	21.0	19.2	17.6	14.7
7	10.2	27.3	24.8	22.5	18.5
8	13.7	34.3	31.0	27.9	22.6
9	17.6	41.9	37.5	33.6	27.0
10	22.1	50.1	44.7	39.8	31.6
11	27.1	58.9	52.3	46.4	36.5
12	32.7	68.3	60.3	53.3	41.6
13	38.9	78.3	68.9	60.6	46.9
14	45.4	88.8	77.8	68.2	52.4
15	52.6	99.8	87.2	76.2	58.1
16	60.2	111.4	97.0	84.4	64.0
17	68.6	123.5	107.2	93.1	70.1
18	77.3	136.1	117.8	102.0	76.4
19	86.8	149.2	128.8	111.2	82.8

the computation time, is not known for many problems or may require an exponential number of iterations as seen in case of the simplex algorithm. In this paper, the worst-case time complexity, which is always found through a theoretical analysis, is not investigated due to its apparent intractability.

9 Conclusions

In this paper, total cost and finish time are minimized in a natural manner in a single-level tree network. The total cost was defined as a summation of the computing and communication costs. The analysis involved a load distribution principle and an adjacent processor pairwise swapping. It was found that it was not possible to develop a simple condition for cost optimally arranging a general single-level tree network. However, for special cases of the single-level tree networks, (i.e., a bus network, a single-level tree network with processors with identical computing speeds and a homogeneous single-level tree network), a simple condition for optimizing processor arrangement can be established. The simple optimal condition requires the processor cost to be ordered in a non-decreasing manner for a bus network; while it requires the processor cost coefficient to be arranged in a non-decreasing manner in

the latter two cases.

A cost efficient processor arrangement algorithm, based on a local search, to find a cost efficient processor arrangement profile was then proposed. It is a heuristic algorithm which is based upon the multi-level neighborhood structure, multiple-initial solutions, the greedy strategy, and the restart search strategy concepts. An experimental performance evaluation involving running the algorithm on a number of single-level tree networks with various sizes was performed. The results indicate an impressive quality of the solution. This is in terms of the effectiveness, a small probability of suboptimal solution convergence, and the goodness of the proximity of the suboptimal solution to the optimal solution in both the average and the worst cases, (both of which are within 5% of the optimal solution). In addition, the experimental results show that the goodness of the suboptimal solution did not deteriorate with the increasing number of children processors. Finally an efficiency in terms of the number of iterations needed to reach a final solution can be bounded by a low-order polynomial in a number of children processors, $O(N^{1.6})$.

This work shows how one could optimize total processing cost in a single-level tree network which models a networked distributed computing system.

It clearly indicates the feasibility of including resource utilization cost into the problem of scheduling divisible load in the distributed computing systems. This bodes well for future implementation of computer utilities.

References

[1] S. Bataineh and T.G. Robertazzi, "Distributed computation for a bus network with communication delays," In *Proceedings of the 1991 Conference on Information Sciences and Systems*, The John Hopkins University, Baltimore, MD., pp 709-714, March, 1991.

[2] V. Bharadwaj, D. Ghose and V. Mani, "Optimal sequencing and arrangement in distributed single-level tree networks with communication delays," *IEEE Transactions on Parallel and Distributed Systems,* vol. 5, no. 9, pp. 968-976, September, 1994.

[3] V. Bharadwaj, D. Ghose, V. Mani and T.G. Robertazzi, *Scheduling Divisible Loads in Parallel and Distributed Systems,* Los Alamitos, California: IEEE Computer Society Press, 1996.

[4] J. Blazewicz and M. Drozdowski, "Scheduling divisible jobs on hypercubes," *Parallel Computing,* vol. 21, pp. 1945-1956, 1995.

[5] Y. C. Cheng and T.G. Robertazzi, "Distributed computation with communication delays," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 24, no. 6, pp. 700-712, November, 1988.

[6] Y. C. Cheng and T.G. Robertazzi, "Distributed computation for a tree network with communication delays," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 26, no. 3, pp. 511-516, May, 1990.

[7] S. Charcranoon and T.G. Robertazzi, "Optimizing computing and communication costs in single-level tree networks," *University at Stony Brook College of Engineering and Applied Science Technical Report 748, September 22, 1997.*

[8] J. Sohn, T.G. Robertazzi and S. Luryi, "Optimizing computing costs using divisible load analysis," accepted for publication in the *IEEE Transactions on Parallel and Distributed Systems.*

[9] C.A.Tovey, "On the number of iterations of local improvement algorithms," *Operations Research Letters,* vol. 2, no. 5, pp. 231-238, Decem-

ber 1983.

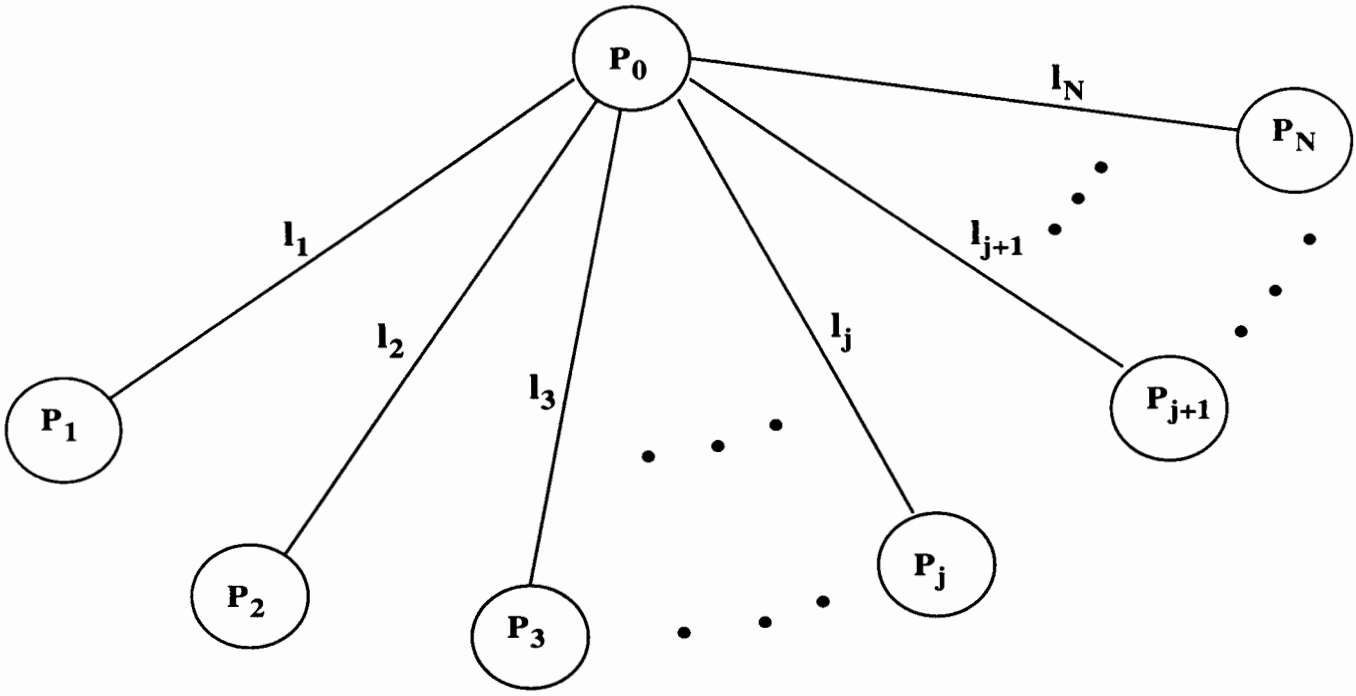


Figure 1: Single level tree network: Normal Case

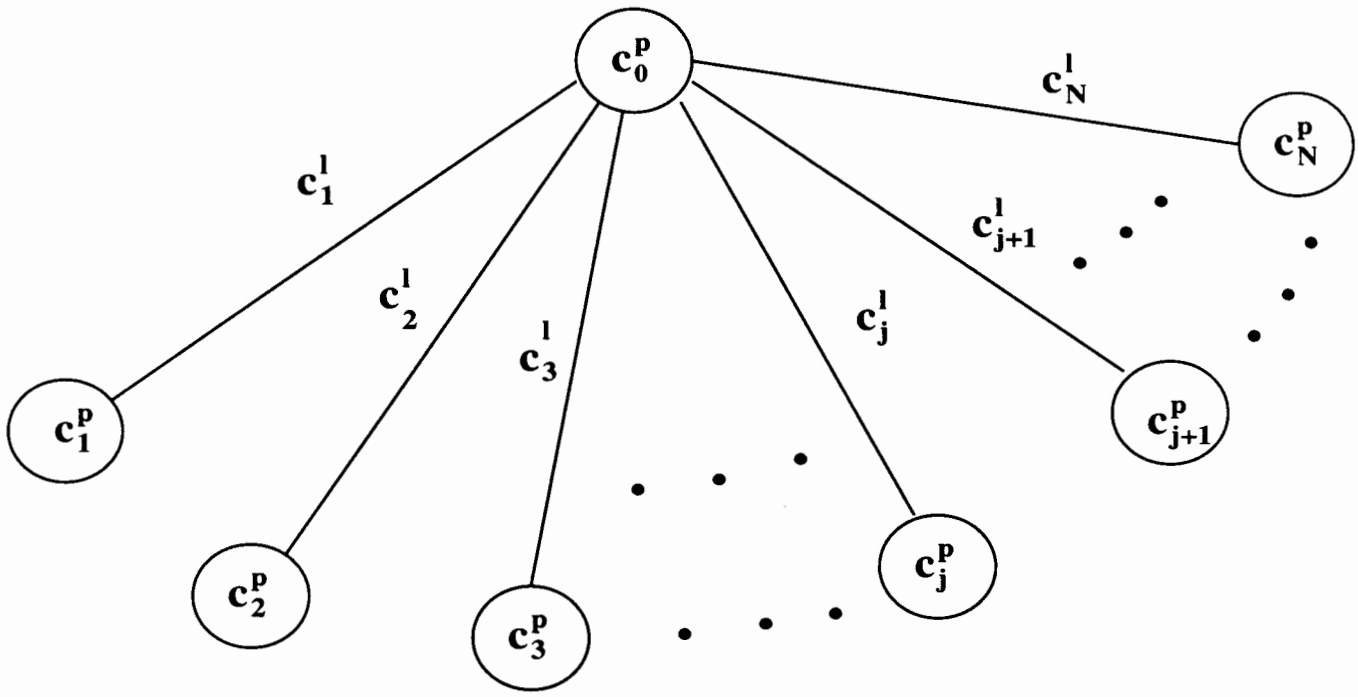


Figure 2: Single level tree network with associated cost coefficients

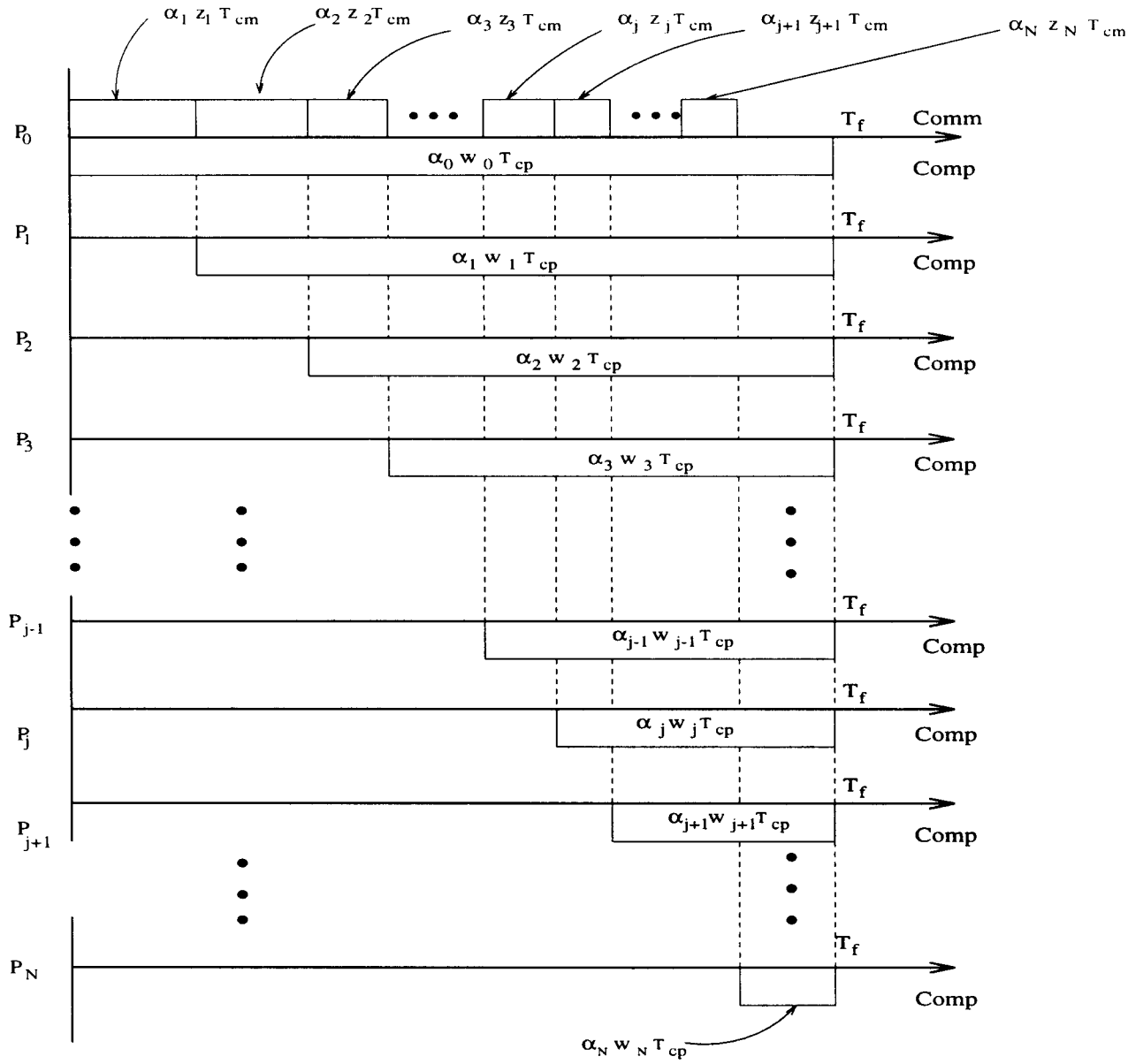


Figure 3: Timing Diagram: Normal Case

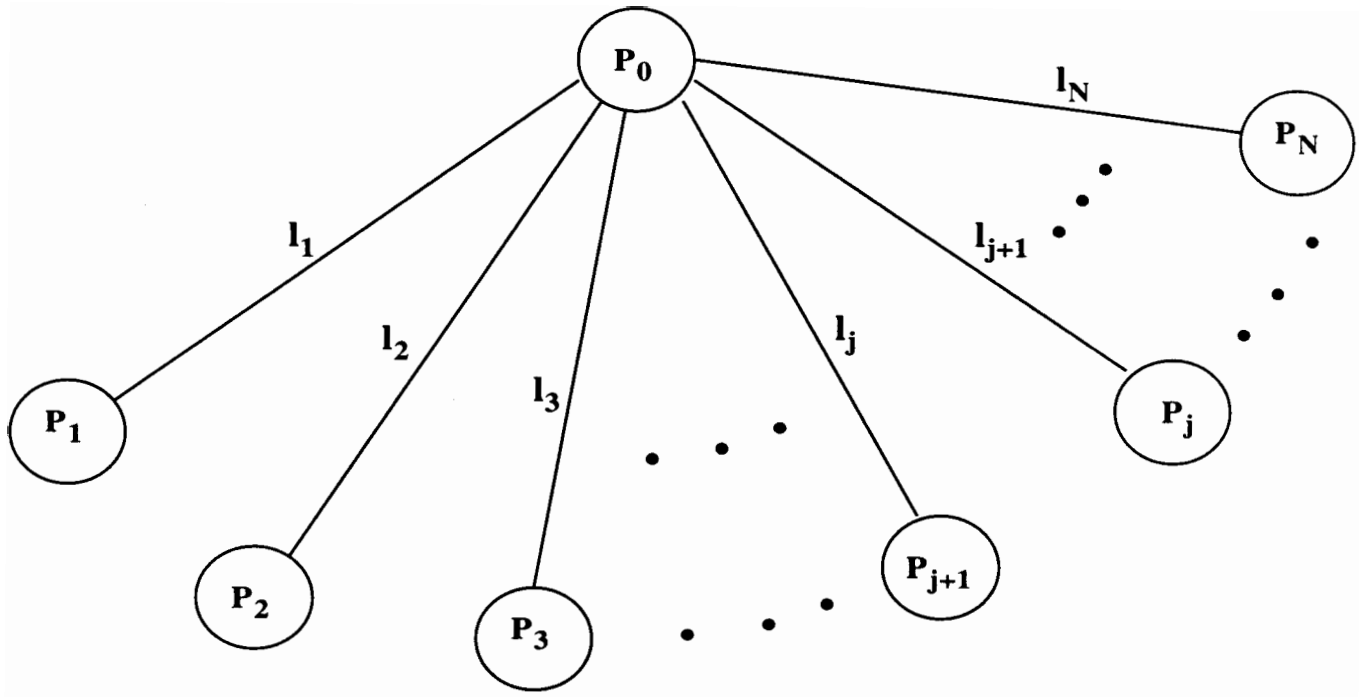


Figure 4: Single level tree network: Adjacent Pairwise Processor Swap

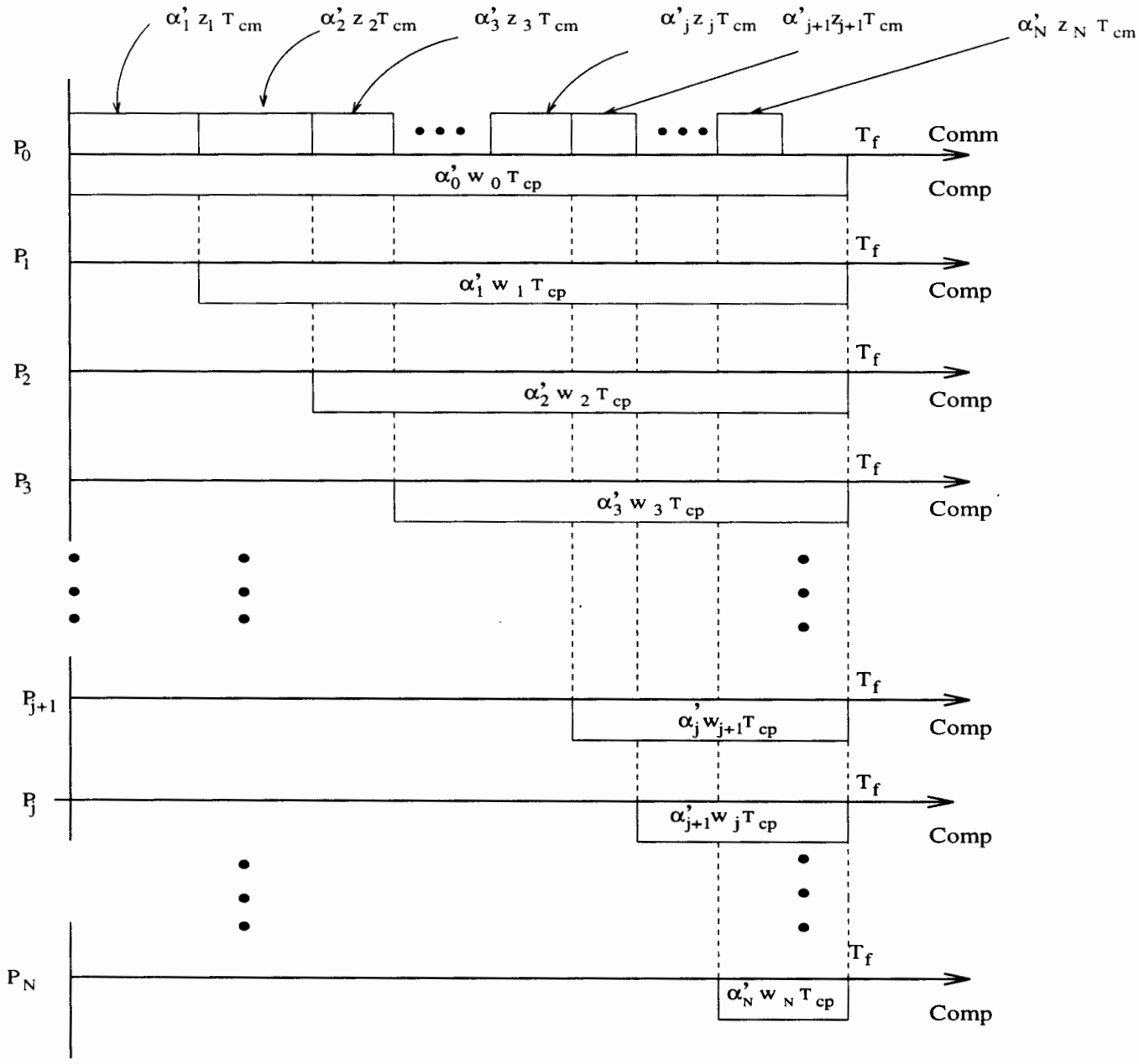


Figure 5: Timing Diagram: Swapping Case