Parallel Processor Configuration Design with Processing/Transmission Costs

Saravut Charcranoon, *Member*, *IEEE*,
Thomas G. Robertazzi, *Senior Member*, *IEEE*,
and Serge Luryi, *Fellow*, *IEEE*

Abstract—A computer configuration design problem where the objective is to configure a parallel processor to do processing in a cost effective manner is examined. The application envisioned is a specialized on-line service that rents time on its machine. The combinatorial optimization problem involved is examined analytically and a heuristic algorithm for its solution is provided. Lessons learned in this work appear in the conclusion.

Index Terms—Cost, divisible load, economics, heuristic algorithm, local search, single-level tree network, star network.

1 Introduction

Over the past several decades, a great deal of research has been performed on the performance evaluation of computer systems. Today, because of the declining cost of computer hardware and the interest in electronic commerce and on-line services, the economics or cost evaluation of networked computation is as deserving of attention as are performance issues. In this paper, we examine a computer configuration design problem that has implications for the leasing of networked computer time.

We envision a scenario where a specialized on-line service wishes to rent time on a high performance parallel machine to users. The question to be investigated is how the parallel processor configuration should be optimized so that, in some sense, the parallel processor can solve a submitted problem at minimal monetary cost to both the service, and by implication, to the user.

A number of deliberate choices were made regarding the features of this problem:

- single level tree (star) topology,
- divisible load,
- linear costs for communication and computation.

Generally, these choices were made for analytical tractability. The divisible load model in particular has, over the years, seen its tractability proven [1] and well models problems involving data parallelism. In spite of these innocuous choices, the related mathematics is substantive. A secondary reason for the choices is that they allow a comparison with earlier published work by some of the authors [4], considering computation costs only, in bus networks.

With these choices we seek, in this paper, to optimize the choice of which of a set of processors to connect to which of the tree network's links. This is a combinatorial optimization problem we call the "processor arrangement" problem. Unfortunately we have not been successful in devising a simple condition to implement an optimal processor arrangement profile (i.e., pairing of processors to links). Instead, expressions of moderate complexity for determining how to improve a given profile will be presented. A heuristic algorithm based on combinatorial local search principles will also be described.

One more choice regarding the problem discussed in this paper requires some explanation. In this work, there are two objective functions to be optimized: the finish (solution) time and the total processing and transmission cost. It is well-known that there are several approaches to solve such multiple objective function 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 [1]. That is, for each possible arrangement of processors, load is allocated so that all processors stop computing at the same time instant and finish time minimized for that specific arrangement profile. While other approaches are certainly possible, we believe that the proposed approach is a natural one for a first study.

The paper is organized as follows: The model and load distribution scheme are presented in Section 2. In Section 3, processor arrangement and monetary cost models are discussed. Adjacent processor swapping is also discussed in this section. Cost efficient processor arrangements and the necessary cost improvement conditions in a single-level tree network are developed in Section 4. The heuristic cost efficient processor arrangement algorithm and its performance evaluation are discussed in Section 5. Finally, the conclusion and lessons learned appear in Section 6.

2 Model, Notation, and Load Distribution

In this section, some necessary modeling, notation, and load distribution equations and their background are discussed.

2.1 Model Descriptions

A single-level tree network where the root processor is equipped with a front-end processor for communications off-loading is considered. The presence of the front-end processor means that the root can compute and communicate simultaneously. A single-level tree network with $(N+1)$ processors and (N) links is shown in Fig. 1. All the processors are connected to the root processor, p_0, via communication links. Associated with the links and processors are the associated linear cost coefficients $c_1^l, c_2^l, \ldots, c_N^l$ and $c_0^p, c_1^p, c_2^p, \ldots, c_N^p$, respectively. 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 $\alpha_1, \alpha_2, \ldots, \alpha_N$ to the children processors $p_1, p_2, \ldots, p_N$, respectively, and sequentially. We do not consider strategies of multiinstallment load distribution [1]. 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. It is assumed that, compared to the size of the data, the time to report solutions back to the root is negligible.

Let:

α_i: The load fraction assigned to the ith link-processor pair.
w_i: The inverse of the computing speed of the ith processor.
z_i: The inverse of the link speed of the ith link.
T_{cp}: Computing intensity constant: The entire load is processed in $w_i T_{cp}$ seconds by the ith processor.
T_{cm}: Communication intensity constant: The entire load can be transmitted in $z_i T_{cm}$ seconds over the ith link.
T_f: The finish time: Time at which the last processor ceases computation.

Then, $\alpha_i w_i T_{cp}$ is the time to process the fraction α_i of the entire load on the ith processor. Note that the units of $\alpha_i w_i T_{cp}$ are [load] $\times$ [sec/load] $\times$ [dimensionless quantity] = [seconds]. Likewise,

- *S. Charcranoon is with Alcatel Corporate Research Center, 1201 E. Campbell Rd., Richardson, TX 75081-1936. E-mail: scharcra@usa.alcatel.com.*
- *T.G. Robertazzi and S. Luryi are with the Department of Electrical and Computer Engineering, State University of New York at Stony Brook, Stony Brook, NY 11794. E-mail: {tom, sluryi}@ece.sunysb.edu.*

Manuscript received 7 Aug. 1998; accepted 3 Apr. 2000.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 112526.

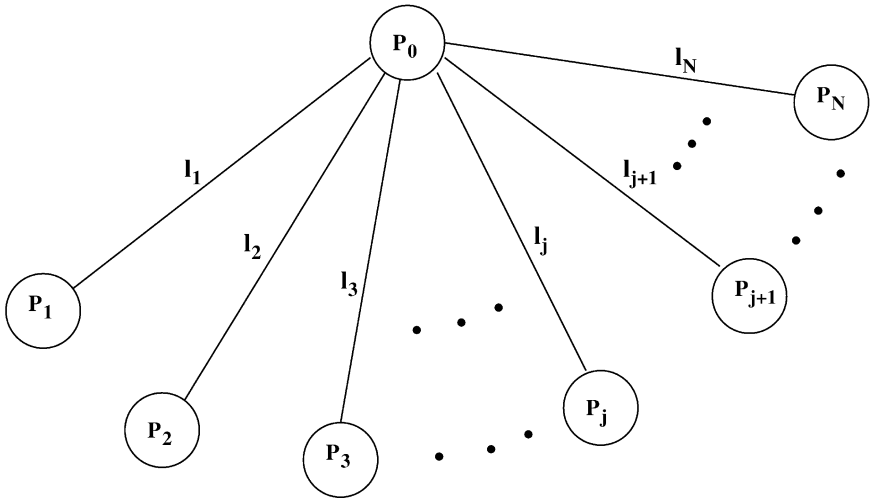

Fig. 1. Single level tree network: normal case.

$\alpha_i z_i T_{cm}$ is the time to transmit the fraction α_i of the entire load over the ith link. Note that the units of $\alpha_i z_i T_{cm}$ are [load] $\times$ [sec/load] $\times$ [dimensionless quantity] = [seconds].

2.2 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. Instead, it is intuitive that, to minimize the processing finish time, the load distribution should be such that all processors finish computing at the same time. Otherwise, the processing finish time could be reduced by transferring some fractions of load from busy processors to idle processors. Formal proofs of this argument in the case of linear, bus, and tree networks appear in [1]. 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 [1], 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 nonzero assigned load have to finish computing at the same time. In this paper, it is assumed that all processors in the network are utilized.

2.3 Fundamental Recursive Equations

The timing diagram of the process of load distribution in a single level tree network is given by Fig. 2. In this figure, each of the $N+1$ processors has a graph associated with it. Communication to and from each processor appears above the time axis and computation by each processor appears below the time axis. Fig. 2 shows the sequential distribution of load fractions, each processor commencing computation upon receiving its load fraction and all processors stopping at the same time. Again, it is assumed that solutions are small enough in comparison with the data that their transmission time back to the root is negligible.

In [4], simple conditions were found for cost optimization for a single level tree network when only computation costs were taken into account. One might think that when communication costs are included that the communication and computation costs on each branch might be collapsed into a single equivalent cost. In fact, though, there are dependencies between the timing of events that make the actual situation more complex. For instance, the second link cannot receive load until the transport of load over the first link is complete. However, all of these dependencies can be taken into account through a series of chained linear equations. As discussed in Section 2.2, since all processors must stop computation at the same instant in order to achieve a minimum finish time, one can set up a series of chained linear equations reflecting this and all other timing relationships. These equations can be solved for the fractions of load, α_is, to be assigned to the processors:

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

However, rather than solving a set of linear equations, it is simpler to chain the equations together recursively to yield:

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

where

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

The normalization equation is given as,

$$\alpha_0 + \alpha_1 + \ldots + \alpha_N = 1. \quad (3)$$

With the normalization equation, one can then resolve the recursive equations (2) to obtain the closed-form expression of all αs as given below. In the following, $w_i T_{cp}$ is the time for the ith processor to process the entire load. Likewise, $z_i T_{cm}$ is the time to communicate the entire load over the ith link. Naturally, $(z_i T_{cm} + w_i T_{cp})$ is the time for the ith processor to receive and process the entire load. For $1 \le n \le N$, one finds:

$$\alpha_n = \frac{1}{D} \prod_{i=0}^{n-1} (w_i T_{cp}) \prod_{i=n+1}^{N} (z_i T_{cm} + w_i T_{cp}), \quad (4)$$

where

$$D = \prod_{i=1}^{N} (z_i T_{cm} + w_i T_{cp}) + \sum_{n=1}^{N} \left(\prod_{i=0}^{n-1} (w_i T_{cp}) \prod_{i=n+1}^{N} (z_i T_{cm} + w_i T_{cp}) \right). \quad (5)$$

3 Processor Arrangement and Monetary Cost

The nature of processor arrangement and a set of linear equations to model processing and transmission monetary costs are presented below. Also discussed are basic swapping relations that could be used in a heuristic algorithm.

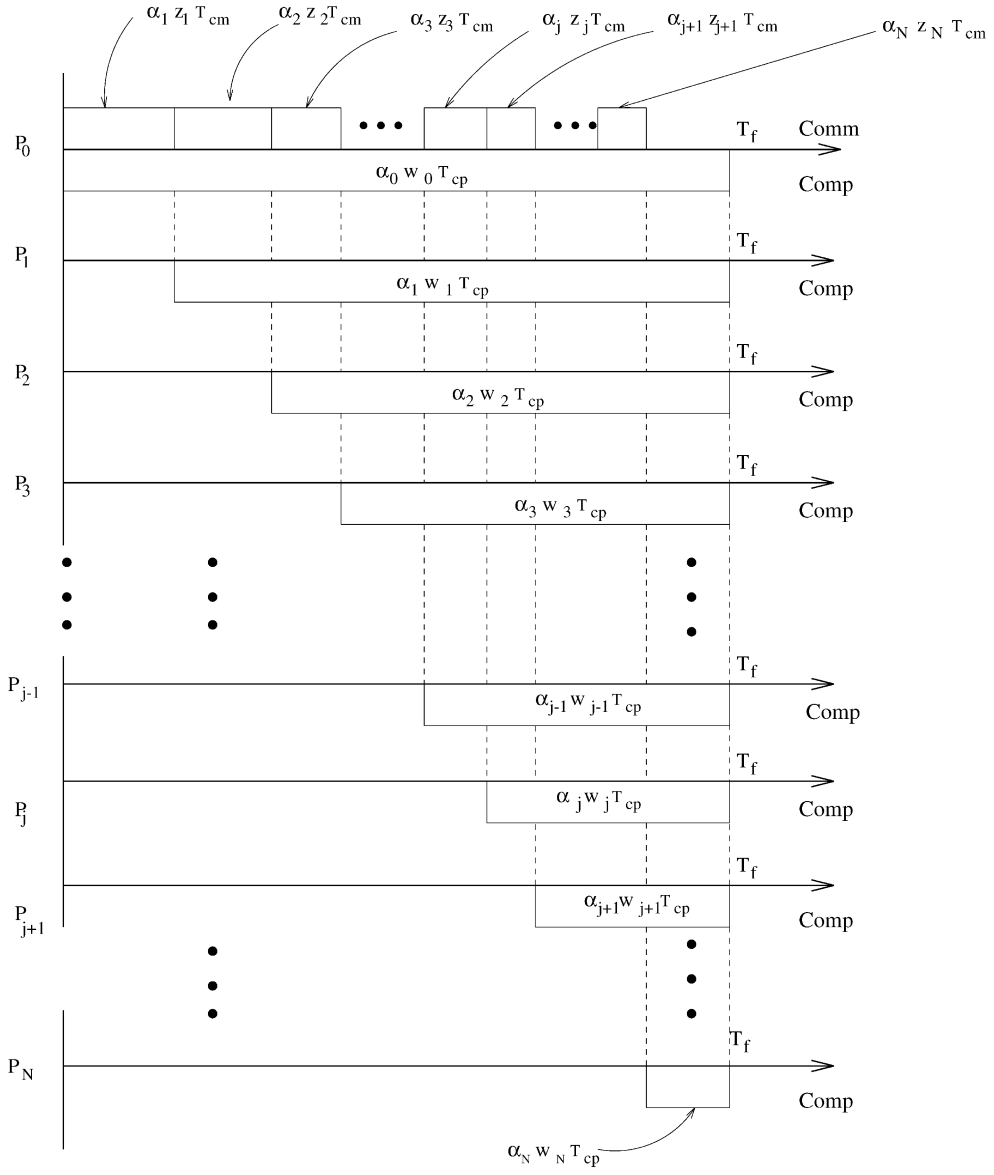

Fig. 2. Timing diagram: normal case.

3.1 Processor Arrangement

Processor arrangement in general involves a permutation of the order of processors (in a single level tree network, here) 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. A processor arrangement determines which processor is connected to $l_1, l_2, \ldots, l_N$. It does not change the order $(l_1, l_2, \ldots, l_N)$ of dispatching fractions of load from the root processor to links. In this paper, a single level tree network can be represented by an ordered set as follows:

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

This ordered set is 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. We seek a processor arrangement profile to minimize the sum of the processing and transmission costs given that, for each possible processor arrangement profile, finish time is minimized using the methodology of Section 2.2 and [1]. Our goal in this and the next section is to determine if there are simple conditions for creating an efficient or optimal arrangement profile of the processors in this sense.

3.2 Link-Processor Monetary Cost

The link-processor monetary cost for processing a fraction of load at any processor is defined as the monetary cost incurred from utilizing the processor and its corresponding link in order to process the associated 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, it is assumed 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 monetary cost is defined only in terms of accounting for the duration during which the resource is busy serving 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. For $0 \leq n \leq N$, let:

c_n^p: the computing cost per second of utilizing the nth processor.

c_n^l: the communication cost per second of utilizing the nth link.

$c_n^p w_n$: the computing cost per load of utilizing the nth processor.
$c_n^l z_n$: the communication cost per load of utilizing the nth link.
$(c_n^p w_n + c_n^l z_n)$: the processing cost per load of the nth link-processor pair.

3.3 Total Monetary Cost

Total monetary 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," we mean a specific arrangement of processors). Therefore, this total cost depends on the processor arrangement profile. Expressions for the cost of processing and transmission for each link-processor pair are given by (6) and (7) and an expression for the total processing cost is given by (8).

$$C_0 = c_0^p w_0 T_{cp} \tag{6}$$

$$C_n = c_n^l z_n T_{cm} + c_n^p w_n T_{cp} \qquad , n = 1, \ldots, N \tag{7}$$

$$C_{total} = \alpha_0 C_0 + \sum_{n=1}^{N} \alpha_n C_n. \tag{8}$$

By substituting α_0 and all α_n from the previous section into (8), one obtains the expression for total cost that explicitly shows the jth and the $(j+1)$st link-processor pairs as:

$$\begin{aligned} C_{total} = \frac{1}{D}\Bigg\{ & \prod_{i=1}^{N}(z_i T_{cm} + w_i T_{cp})(c_0^p w_0 T_{cp}) \\ & + \sum_{n=1}^{j-1}\left[\prod_{i=0}^{n-1}(w_i T_{cp}) \prod_{i=n+1}^{N}(z_i T_{cm} + w_i T_{cp})(c_n^l z_n T_{cm} + c_n^p w_n T_{cp})\right] \\ & + \prod_{i=0}^{j-1}(w_i T_{cp}) \prod_{i=j+1}^{N}(z_i T_{cm} + w_i T_{cp})(c_j^l z_j T_{cm} + c_j^p w_j T_{cp}) \\ & + \prod_{i=0}^{j}(w_i T_{cp}) \prod_{i=j+2}^{N}(z_i T_{cm} + w_i T_{cp})(c_{j+1}^l z_{j+1} T_{cm} + c_{j+1}^p w_{j+1} T_{cp}) \\ & + \sum_{n=j+2}^{N}\left[\prod_{i=0}^{n-1}(w_i T_{cp}) \prod_{i=n+1}^{N}(z_i T_{cm} + w_i T_{cp})(c_n^l z_n T_{cm} + c_n^p w_n T_{cp})\right]\Bigg\}. \end{aligned} \tag{9}$$

The total cost can be put in a relatively simple form as $C_{total} = \frac{N}{D}$. Thus, the corresponding numerator, N, is the collection of terms within the curly brace. The corresponding denominator, D, with the terms due to the jth and the $(j+1)$st link-processor pairs explicitly shown, is

$$\begin{aligned} D = & \prod_{i=1}^{j-1}(z_i T_{cm} + w_i T_{cp})(z_j T_{cm} + w_j T_{cp})(z_{j+1} T_{cm} + w_{j+1} T_{cp}) \\ & \prod_{i=j+2}^{N}(z_i T_{cm} + w_i T_{cp}) \\ & + \sum_{n=1}^{j-1}\Bigg(\prod_{i=0}^{n-1}(w_i T_{cp}) \prod_{i=n+1}^{j-1}(z_i T_{cm} + w_i T_{cp})(z_j T_{cm} + w_j T_{cp}) \\ & (z_{j+1} T_{cm} + w_{j+1} T_{cp}) \cdot \prod_{i=j+2}^{N}(z_i T_{cm} + w_i T_{cp})\Bigg) \\ & + \prod_{i=0}^{j-2}(w_i T_{cp})(w_{j-1} T_{cp})(z_{j+1} T_{cm} + w_{j+1} T_{cp}) \prod_{i=j+2}^{N}(z_i T_{cm} + w_i T_{cp}) \\ & + \prod_{i=0}^{j-2}(w_i T_{cp})(w_{j-1} T_{cp})(w_j T_{cp}) \prod_{i=j+2}^{N}(z_i T_{cm} + w_i T_{cp}) \\ & + \sum_{n=j+2}^{N}\left(\prod_{i=0}^{n-1}(w_i T_{cp}) \prod_{i=n+1}^{N}(z_i T_{cm} + w_i T_{cp})\right). \end{aligned} \tag{10}$$

3.4 Adjacent Pairwise Processor Swapping

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 Fig. 1. A swapped processor arrangement profile is a profile obtained by implementing a single adjacent pairwise processor swap of one of the adjacent processor pairs of the current profile, a swap of p_j and p_{j+1}. Pairwise swapping is of interest as a building block for a heuristic algorithm for this problem (see Section 5).

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

$$\pi = \{p_0, (l_1, p_1), \ldots, \underline{(l_j, p_j)}, \underline{(l_{j+1}, p_{j+1})}, (l_{j+2}, p_{j+2}), \ldots, (l_N, p_N)\} \tag{11}$$

$$\pi' = \{p_0, (l_1, p_1), \ldots, \underline{(l_j, p_{j+1})}, \underline{(l_{j+1}, p_j)}, (l_{j+2}, p_{j+2}), \ldots, (l_N, p_N)\}. \tag{12}$$

4 Cost Efficient Processor Arrangement

In an attempt to aid algorithm development, conditions are sought under which swapping adjacent processors lowers the total monetary cost. For brevity, we summarize the theorem's development (see [2], [3] for an extended development). One can develop expressions for C'_{total}, the total monetary cost under an adjacent processor swapping. As with C_{total}, C'_{total} can be expressed as a rational function. Then:

$$C'_{total} - C_{total} = \frac{N_{\pi'}}{D_{\pi'}} - \frac{N_\pi}{D_\pi} \tag{13}$$

$$= \frac{N_{\pi'} D_\pi - N_\pi D_{\pi'}}{D_\pi D_{\pi'}}. \tag{14}$$

The conditions under which N_π and $N_{\pi'}$, and D_π and $D_{\pi'}$ are greater than, less than, or equal to each other can be developed. This leads to conditions under which the difference in C_{total} and C'_{total} can be determined from a number of conditions. Finally, one has:

Theorem 1. *In a single-level tree network, if one of the following conditions is satisfied, then the total cost of the adjacent pairwise*

swapped processor arrangement profile $C'_{total}(\pi'_{(p_1,\ldots,\underline{p_{j+1},p_j},\ldots,p_N)})$ *is less than the total cost of the current processor arrangement profile* $C_{total}(\pi_{(p_1,\ldots,\underline{p_j,p_{j+1}},\ldots,p_N)})$ *for* $1 \leq j < N$. *Otherwise,*

$$C'_{total}(\pi'_{(p_1,\ldots,\underline{p_{j+1},p_j},\ldots,p_N)})$$

is greater than or equal to

$$C_{total}(\pi_{(p_1,\ldots,\underline{p_j,p_{j+1}},\ldots,p_N)}).$$

1. $(z_{j+1}-z_j)(w_j-w_{j+1})=0 \quad and \quad (w_j-w_{j+1})(z_{j+1}c^l_{j+1} - z_jc^l_j) > z_{j+1}(w_{j+1}c^p_{j+1} - w_jc^p_j)$
2. $(z_{j+1}-z_j)(w_j-w_{j+1}) \neq 0 \quad and \quad (w_j-w_{j+1})(z_{j+1}c^l_{j+1} - z_jc^l_j) \geq z_{j+1}(w_{j+1}c^p_{j+1} - w_jc^p_j) + \Delta$
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* $C'_{total} < \frac{(N_\pi - N_{\pi'})}{(D_\pi - D_{\pi'})}$

Note that, in (2):

$$\Delta = \frac{1}{d}\left\{ \left|(z_j - z_{j+1})(w_j - w_{j+1})\right| \left[aC_0 + \sum_{n=1}^{j-1} k_n C_n \right] \right\}. \tag{15}$$

While it is always possible that additional work may result in simpler conditions than these for deciding when making a processor swap is cost effective, our feeling is that it would be difficult to simplify the analytical conditions above.

We now turn our attention to a heuristic algorithm for solving this problem.

5 Heuristic Processor Arrangement Algorithm

In this section, a heuristic algorithm to solve the processor arrangement problem efficiently is described.

One can use the conditions of Theorem 1 as the basis of a simple greedy algorithm implementing adjacent pairwise swapping for the processor arrangement problem. However, a simple greedy implementation has a fairly large tendency (as high as 18 percent in our runs, see [2], [3]) to converge to locally optimal solutions.

In order to improve the performance of a processor arrangement algorithm, four main actions were taken. First, several initial processor arrangement profiles, which serve as starting points, are generated. Second, the extent of a neighborhood is enlarged to cover neighborhoods other than that obtained by adjacent processor pairwise swapping. In this case, the cost of a profile is evaluated using (9) and (10). Third, a greedy strategy is applied to select the best processor arrangement profile. Finally, a restart searching strategy is used to determine a pair of processors to start over with after a new processor profile has been obtained.

Experimental results [2], [3] demonstrate that the probability of the heuristic greedy algorithm converging to a suboptimal solution is extremely small (on the order of 1 in 1,000 in our runs). Moreover, the few suboptimal solutions found are very close to the optimal solution in terms of cost. They are mostly within 1 percent of an optimal solution with the maximum observed deviation being about 5 percent from an optimal solution.

One problem we did encounter was an inability to solve problems with more than 20 or so child processors due to numerical difficulties (the costs of the current and swapped profiles tended to be equal to the limits of machine precision, making it not possible to make swapping decisions).

Finally, the computational efficiency of our algorithm can be empirically bounded by $O(N^{1.6})$ for N up to 20. This is consistent with the results in [5].

6 Conclusions and Lessons Learned

A number of lessons can be drawn from this work:

1. *Complex mathematics:* Setting up a cost-based computer configuration design problem, even with generic assumptions, of linearity, load divisibility, and star topology, leads to a fairly complex exercise in algebra.
2. *No simple conditions:* In an earlier work for a bus network with only computation costs (not communication costs) [4], relatively simple conditions for optimal load distribution were found. The expressions in this paper, where communication costs are considered, lack that absolute simplicity. It is always possible that additional work may result in simpler optimal load distribution conditions. Based on our experience on this problem though, we think this would be difficult.
3. *Combinatorial algorithms possible:* Our experience indicates that efficient, if not always optimal, solutions can be produced using combinatorial optimization algorithms. While a special purpose heuristic algorithm was presented in this paper, the use of other combinatorial approaches, such as tabu search, simulated annealing, and genetic algorithms, is possible. One problem we did encounter was an inability to solve problems with more than 20 or so child processors due to numerical difficulties.
4. *Computer Configuration Possible:* Using combinatorial optimization techniques creating efficient cost-based computer configuration designs is feasible for small N.
5. *Open Question:* An open question is efficient configuration design for large networks.

All in all this is an interesting problem area as it involves an integration of cost and performance issues.

Acknowledgments

This work was performed while S. Charcranoon was a PhD student at the State University of New York at Stony Brook.

References

[1] V. Bharadwaj, D. Ghose, V. Mani, and T.G. Robertazzi, *Scheduling Divisible Loads in Parallel and Distributed Systems*. Los Alamitos, Calif.: IEEE CS Press, 1996.

[2] S. Charcranoon, T.G. Robertazzi, and S. Luryi, "Cost Efficient Processor Arrangement in Single Level Tree Networks," State Univ. of New York at Stony Brook College of Eng. and Applied Science Technical Report 757, 30 Mar. 1998, available from T. Robertazzi.

[3] S. Charcranoon, T.G. Robertazzi, and S. Luryi, "Parallel Processor Configuration Design with Processor/Transmission Costs," State Univ. of New York at Stony Brook College of Eng. and Applied Science Technical Report 782, 5 July 2000, available from T. Robertazzi.

[4] J. Sohn, T.G. Robertazzi, and S. Luryi, "Optimizing Computing Costs Using Divisible Load Analysis," *IEEE Trans. Parallel and Distributed Systems*, vol. 9, no. 3, pp. 225-234, Mar. 1998. Related: US Patent 5,889,989 *Load Sharing Controller for Optimizing Monetary Cost*, 30 Mar. 1999.

[5] C.A. Tovey, "On the Number of Iterations of Local Improvement Algorithms," *Operations Research Letters*, vol. 2, no. 5, pp. 231-238, Dec. 1983.