計算機アーキテクチャ 70－3
（１９８８．３．１０）

ベンズINのための並列パーミュテーションアルゴリズムの下限値の導出

アィサム A. ハミド　　白鳥 則郎　　野口 正一

東北大学電気通信研究所
〒980 仙台市片平2-1-1

本論文では、ベンズIN (Interconnection Network) において、制限された並列性について議論し、並列パーミュテーションアルゴリズムの下限値として、 $O((\Phi/\mathbb{N})\log_{\varepsilon}\mathbb{N})$ を導出する。ここで、$\mathbb{N}$ は入力の数、Φ は計算モデルを構成する処理要素の総数、ε は直接に結合されている隣接する処理要素の数を示す。この下限値は、$\mathbb{N} \gg \Phi$ (これを制限された並列性と呼ぶ) の場合に有用である。また、$\mathbb{N}$ や Φ に加えて、ε が並列パーミュテーションアルゴリズムの複雑さに影響を与えることを証明する。

Lower Bound Derivation of Parallel Permutation Algorithm for Benes Network－ With Restricted Parallelism

Issam A. HAMID, Norio SHIRATORI, and Shoichi NOGUCHI
Research Institute of Electrical Communication,
Tohoku University, 1-1-2, Katahira, Sendai, 〒980.

This companion paper has discussed the limited restriction of parallelism, such that we have derived the lower bound of parallel permutation algorithms as $O((\Phi/\mathbb{N})\log_{\varepsilon}\mathbb{N})$, *where* ε *is the number of the neighboring Processing Elements(PEs) directly connected,* Φ *is the total number of PEs consisting the general nonshared memory computation model, and* $\mathbb{N}$ *is the number of data elements that represents the permutation. This bound is useful in case that,* $\mathbb{N} \gg \Phi$ *(restricted parallelisim) when the number of the data items are much bigger than the total number of PEs, therefore each PE will compute a subpermutation of* $\lceil \mathbb{N}/\Phi \rceil$ *data items. We have proven that* ε, *(in addition to* $\mathbb{N}$ *and* Φ*) affect the complexity of the (dynamic) parallel permutation algorithm which has been shown that it is not completely parallelizable on any* ε*-parallel computational model.*

I. Introduction

Parallel computers which have thousands of processors working cooperatively, are becoming more feasible and practical for many applications for instance, the Connection Machine[83]. Generally, there are two groups of models for the parallel architectures. The first group is concerned with the shared memory model to which all processors have access. The second model is concerned with the many processors with private local memories, each accessed exclusively by one processor. The second group should effectively be capable to permute data items among the Processor Elements; PEs by high permutability interconnection network. We have concentrated on the second group because, it is more practical to be constructed compared to the first group's model, which has many drawbacks such as; memory conflict and the $\mathbb{N}-1$ fanout per processors, (where $\mathbb{N}$ is the total number of PEs). But, in the second group's model the interconnection network becomes the most important part to synthesize powerful mappings.

We have concentrated in our research on a type of Interconnection Network (IN) named as Benes IN, because of its capability to realize arbitrary permutation at its input. In a companion paper[Ha88], we have presented a new computational model with efficient algorithm to control the SWitching elements; (SWs), in a setting time of $O(\lceil n/k \rceil \log_2 \mathbb{N})$, assuming $\mathbb{N}$ is the number of the input elements at the input of IN, and it is of base 2, (i.e., $n=\log_2\mathbb{N}$), k is the number of the Processor Elements; (PEs), per cycle of the Cyclic Cube Engine; (CCE). However, that bound is within the lower bound for setting Benes IN, for arbitrary permutation, i.e., $O(\log_2\mathbb{N})$, which also, represents the propagation time delay for such type of IN[Wa68]. These results could be concluded by assuming the number of processors of the computation model are large enough to hold one record; (record represents one data element of a permutation), therefore, $\Phi \geqq \mathbb{N}$, where Φ is the number of PEs which are constructed and connected according to the computation model. This means that the computational model should always be tolerable to the size of the problem (i.e., the number of data items to be permuted or processed), therefore according to $\mathbb{N}$ we can specify Φ. In this paper, we have changed the assumption concerned with $\Phi \geqq \mathbb{N}$, such that, Φ becomes constant, and $\mathbb{N}$ data items is so large, i.e., $\mathbb{N} \gg \Phi$. Therefore, we want to examine the parallel permutation algorithm presented in the companion paper, and see the complexity of the setting time for arbitrary permutation using the computational model of the Cyclic Cube Engine, (CCE) [Ha88]. However, we have proved that the permutation problem is not completely parallelizable on any degree parallel computational model. Specifically, permuting $\mathbb{N}$ data items on the parallel model of Φ total number of PEs, each with ε of direct connected neighbor PEs(i.e., for CCE model $\varepsilon=3$), is required a lower bound of $\Omega((\mathbb{N}/\Phi)\log_\varepsilon \Phi)$ data routings steps, when $\mathbb{N} \gg \Phi$. Please note that, if $\mathbb{N}=\Phi$, then we have the above bound to become as $\mathbb{N}\log_\varepsilon \Phi$, as were shown in the pervious paper[Ha88].

This paper is organized to have, Chap. I , for the introduction, Chap. II, represents the assumptions and notations, with some realizations useful to describe our construction. Chap. III, represents the theortical construction for the parallel permutation algorithm by assuming $\mathbb{N} \gg \Phi$.

II. Assumptions and Notations

In this chapter we have given assumptions and notations with some useful characteristics related to this paper.

(1) We have represented here, every PE as a vertex of graph. The edges represent the connection between these PEs. Therefore, the structure of the CCE, can be represented by undirected graph such that every vertex represents a PE and the communication between them is represented by the edges that connect these vertices.

(2) Assume Φ; represents the set of the total number of the PEs consisting the computation model represented by the CCE [Ha88].

(3) In order to have more general results, we assume here to have a model whose PEs have a degree of ε. Therefore, degree ε PE is the ε number of the neighbor PEs. Please note that, $\varepsilon=3$ for our CCE model.

(4) Generally the time complexity of an algorithm denoted as; $\tau_\Phi(\mathbb{N})$ in a sequential model, (the sequential model is the model that has a single PEs and executes its operations serially, i.e., without any form of parallelism; $\tau_{(\Phi=1)}(\mathbb{N})$), is a function of the problem size; $\mathbb{N}$ which represents the number of records or data items for certain permutation. Whereas, for the parallel algorithms, their time complexity is a function of the problem size $\mathbb{N}$, and the parallel processor size; Φ. (The computations carried out locally at a PE is included here.)

(5) Generally there are different kinds of applications that are affected or not affected with the size of the computation model; i.e., Φ. This is related to how much parallelism we can offered using these Φ PEs. For instance, for the summing of the $\mathbb{N}$ numbers, there is a sequential algorithm of exact bound of $\Theta(\log \Phi)$. But if $\mathbb{N}$ becomes larger than Φ; $\mathbb{N} > \Phi$, we can let each PE computes $\lceil \mathbb{N}/\Phi \rceil$,

numbers, individually. Therefore, the total sum of these local sums, can be computed in time of, $\Theta((\mathbb{N}/\Phi)+\log\Phi)$, but if $\mathbb{N}\gg\Phi$, then $\tau_\Phi(\mathbb{N})\simeq\Theta(\mathbb{N}/\Phi)$.

We will have the following claim concerning with the time complexity of a problem. Claim: If the complexity of a problem on a sequential proccesor is $\tau_\Phi(\mathbb{N})$ then its complexity on a parallel processor of size Φ is $\Omega(\tau_\Phi(\mathbb{N})/\Phi)$.

(6) As well as the main considerations in our discussion is how to permute the N data items in parallel, between Φ PEs, assuming $\mathbb{N}\gg\Phi$. Therefore, the computation of Z output items denoted as ; $(z_0,z_1,....z_{\mathbb{N}-1})$, depends on N input items $(a_0,a_1,....a_{\mathbb{N}-1})$. If each a_i data item needs at least, one z_i then $f(a_0,a_1,....a_{\mathbb{N}-1})=z_0,z_1,....z_{\mathbb{N}-1}$, such that $f(a_i=x_1)\neq f(a_i=x_2)$ for every i, where f denotes a permutation function like for instance, the perfect shuffle permutation.

(7) Let $\Phi=\{PE_0,PE_1,....PE_{\Phi-1}\}$, represents the total parallel PEs consisting the parallel computation model, such that every PE has ε neighbors of PEs. The distance from a PE_i to PE_j, is represented by ∂ such that; $\partial(PE_i,PE_j)$ denotes an integer value. But for $\partial(PE_i,PE_j)=\infty$, if there is no connection or path between those PEs. Also, note that, the minimum or short distance between those processors denoted as Δ, represents the least integer of ∂, such that, the short distance from PE_i, to $PE_j=PE_0,PE_1,...PE_\Delta=PE_i$. Therefore PE_i is within distance Δ of PE_j if $\partial(PE_i,PE_j)\leqq\Delta$.

(8) The speedup, denoted as $SP_\Phi(\mathbb{N})$, of a parallel algorithm is the ratio of the time complexity of the fastest known sequential algorithm to the time complexity of the parallel algorithm for the same problem. This is can be concluded because a parallel computational model can work as a sequential processor by using a single PE. Also, the sequential processor can works as a parallel processor at the cost of time as a factor of Φ.

(9) In this paper we have followed Knuth's terminologies which indicate the asymptotic orders of algorithms, represented as, O, Θ, Ω, which are representing the upper, exact and lower bounds respectively.

III. Lower Bound for the Parallel Permutation Algorithms

In this chapter we have studied the asymptotic lower bound of the complexity of the permutation algorithm for any Φ-parallel computational model. Specifically, we want to show that a permutation algorithm for a degree Φ parallel processor needs time of $\Omega((\mathbb{N}/\Phi)\log_\varepsilon\Phi)$. Therefore, any permutation algorithm for a bounded degree parallel computational model needs time $\Omega((\mathbb{N}/\Phi)\log_\varepsilon\Phi)$ for the case when $\mathbb{N}\gg\Phi$. Therefore, the parallel permutation algorithms are not parallelizable on any degree parallel model.

This chapter have been divided into three sections. In III.1, we have presented a general bounds for any parallel computational model which is completely parallelizable for any problem. While, in Sec.III.2, we have presented the lower bound for the parallel permutation algorithms taking in the consideration that $\mathbb{N}\gg\Phi$ for any parallel computational model in general and our CCE model in special. Sec.III.3, represents, how the parallel permutation algorithms are not completely parallelizable on the CCE model and other parallel computational models by analyzing the lower bounds of those algorithms due to its execution on the CCE model.

III.1. The General Computational Bounds

We have presented here, a general lower bounds for any parallel computational model which is completely parallelizable for any problem. We mean by the completely parallelizable problems as the problems which utilize optimally all the PEs of the parallel model. Hence, as well as we have Φ PEs, we can execute on our model Φ computations, simultaneously. These computations represent a sequence of time steps during which each PE can do trivial operations such as for instance; either, 1) do binary operation, or 2) do binary test, or 3) receive data, or 4) receive test result. Therefore, we want to find the lower bound for Φ computations on a parallel processor of ε-degree, in general and our CCE model of $\varepsilon=3$, in special.

Due to previous results concerning with the graph problems[Ge78], we can have the number of vertices or PEs, (recall that every PE corresponds to one vertex), within distance ∂ of PE_i which is $\leqq\varepsilon^{\partial+1}-1$. We have modified this result in the following proposition.

Proposition 1

If Υ is the number of PEs within distance ∂ of PE_i, then;

$$\Upsilon=1+\sum_{i=0}^{\partial-1}\varepsilon(\varepsilon-1)^i, \qquad \blacksquare$$

proof:

Within distance 0, only PE_i itself is within this distance.

Within distance 1, there are ε neighbors PEs. Within distance 2, there are $\varepsilon-1$ neighbors, and so on. Thus in general within distance ∂ the number of PEs is at most; $1+\varepsilon+\varepsilon(\varepsilon-1)+\varepsilon(\varepsilon-1)^2+.....\varepsilon(\varepsilon-1)^{\partial-1}$. Therefore, for instance for $\varepsilon=2$, $\Upsilon\leqq2\ \partial+1$, and for $\varepsilon\geqq3$, $\Upsilon\leqq(\varepsilon(\varepsilon-1)^\partial-2))/(\varepsilon-2)$. □

Due to the above proposition we can find the maximum distance between PE_i, and any other PE such as PE_j such that PE_j belongs to Υ. Such

maximum distance can be represented by the following proposition.

Proposition 2

If Υ is the number of PEs within distance ∂ of PE_i, such that $\Upsilon \subset \Phi$, and if $PE_i \in \Phi$, $PE_j \in \Upsilon$, then for $\varepsilon > 1$ we may have, $\log_\varepsilon(|\Upsilon|+1)-1 \leqq maximum(\partial(PE_i,PE_j))$ ■

proof:

Let $maximum(\partial(PE_i,PE_j)) = \partial_M$, hence all $PE_j \in \Upsilon$ are within ∂_M of PE_i. Therefore using the propos. 1, we may have, $|\Upsilon| \leqq \varepsilon^{(\partial_M+1)}-1$, which results, $\log(|\Upsilon|+1) \leqq (\partial_M+1)\log\varepsilon$, from which the results can follow. □

From Propos.2 we can conclude that a parallel computational model of degree ε, has maximum distance of at least $\log_\varepsilon(|\Phi|+1)-1$. This conclusion can be proved by putting Φ instead of Υ in Propos.2. Please note that the maximum distance between two vertices in a graph represents the graph diameter, in the graph terminology. Due to the above two propositions we can construct the following Lemma which gives the lower bound of any parallel computational model of degree of ε; $\varepsilon > 1$. Therefore the Φ computations for a single data item z_i of a permutation depends on $\mathbb{N}$ given data items and requires a time of $\Omega(\log_\varepsilon \mathbb{N} + (\mathbb{N}/\Phi))$, for any ε such that, $1 < \varepsilon < \Phi$.

Lemma 1

The Φ computations of a single data item z_i of a permutation depends on $\mathbb{N}$ given data items and requires a time of $\Omega(\log_\varepsilon \mathbb{N} + (\mathbb{N}/\Phi))$, for any ε such that, $1 < \varepsilon < \Phi$. ■

proof:

If we can be able to partition or divide the data items for the permutation into the available number of PEs, (recall that $\Phi \ll \mathbb{N}$), such that, each of $\lceil \mathbb{N}/\Phi \rceil$ data items is in one PE. Therefore any PE examines $\Omega(\log_\varepsilon \mathbb{N})$ data items and test results, hence, $\Omega(\mathbb{N}/\log_\varepsilon \mathbb{N})$ PEs must communicate in a direct way or indirect way with the PE that has the output value z. As well as $\log(\mathbb{N}/\log_\varepsilon \mathbb{N}) = \Omega(\log \mathbb{N})$, hence the results follows from the propos.2. □

Due to this result we can compute a single item (z), (depending on $\mathbb{N}$ data items), by the parallel computational model whose PEs are much less than the number of these data items, in time of $\Omega(\log \Phi + (\mathbb{N}/\Phi))$. This can be correct for the summing of the $\mathbb{N}$ numbers. The summing of the $\mathbb{N}$ numbers in parallel algorithm using $\mathbb{N} = \Phi$ PEs, can take $\Theta(\log \Phi)$. But as well as $\mathbb{N} \gg \Phi$, we can let each PE computes $\lceil \mathbb{N}/\Phi \rceil$ numbers individually. Therefore the total time for summing $\mathbb{N}$ numbers is $\Theta((\mathbb{N}/\Phi) + \log \Phi)$.

In fact Lemma-1 represents a very general lower bound for computing an output items from $\mathbb{N}$ input items. But in our work we have concentrated on the parallel permutation algorithms. Therefore, from now on we will discuss the lower bound concerning with the parallel permutation algorithm on a parallel computational model of any ε, such that, $1 < \varepsilon < \Phi$, and $\mathbb{N} \gg \Phi$.

III.2. The General Bound for the Parallel Permutation Algorithm

In this section we have presented the lower bound for the parallel permutation algorithms taking in the consideration that $\mathbb{N} \gg \Phi$ for any parallel computational model in general and for our CCE model in special, such that $1 < \varepsilon < \Phi$, and $\mathbb{N} \gg \Phi$. Therefore, in order to achieve this, let us prove at first through the following proposition, that most pairs of PEs are separated by a distance of at least logarithm in terms of the number of PEs.

Proposition 3

Assume that the following set denoted as; Λ of PEs includes the PEs such that; $\Lambda \equiv \{PE_j \in \Gamma \mid \partial(PE_i,PE_j) > \xi \log|\Gamma|\}$, where Γ represents a set of PEs such that $\Gamma \subset \Phi$, and $PE_i \in \Gamma$. Also assume, for each $\xi < 1$ there exists an $\sigma > 0$, such that, the set Λ has cardinality at least equal to $\lfloor \xi |\Gamma| \rfloor + 1$, when $\sigma = 1/(2(\log\varepsilon - \log(1-\xi)))$ ■

proof:

By Propos.1 we may have, $\lambda - \gamma \leqq (\varepsilon^{\lceil \sigma \log \lambda \rceil} - 1)$, where $\lambda = |\Lambda|$, and $\gamma = |\Gamma|$. Hence, $\gamma \geqq \lambda - (\varepsilon^{\lceil \sigma \log \lambda \rceil} - 1 = (\lfloor \xi \lambda \rfloor + 1) + \lceil (1-\xi) \rceil - \varepsilon^{\lceil \sigma \log \lambda \rceil}$. Therefore it is suitable to select σ such that; $\varepsilon^{\lceil \sigma \log \lambda \rceil} \leqq \lceil (1-\xi)\lambda \rceil$, therefore, $(\sigma \log \lambda + 1)\log\varepsilon \leqq \log\lambda + \log(1-\xi)$, and solve for σ in terms of other known variables, therefore ,

$$\sigma \leqq \frac{\log\lambda + \log(1-\xi) - \log\varepsilon}{\log\lambda \log\varepsilon}; \text{(by dividing by } \log\varepsilon\text{)};$$

$$\sigma \leqq \frac{1 + (\log(1-\xi) - \log\varepsilon)/\log\lambda}{\log\varepsilon}; \text{(by dividing by } \log\lambda\text{)}$$

$$\sigma \leqq \frac{1 + \log(\frac{1-\xi}{\varepsilon})/\log\lambda}{\log\varepsilon}, \text{then } \log(\frac{1-\xi}{\varepsilon} - \lambda) = -1/2$$

hence,we have $\sigma \leqq 1/(2\log\varepsilon)$ and this can be true if, $\lambda \geqq ((1-\xi)/\varepsilon) - 2$.

However, for any λ, if $\sigma \geqq (1/\log\lambda)$, leads to satisfy this proposition, therefore, $\sigma \leqq (1/\log((1-\xi)/\varepsilon) - 2) = 1/(2(\log\varepsilon - \log(1-\xi)))$, and this is will be true if $\lambda < ((1-\xi)/\varepsilon) - 2$, this will lead us to select σ, such that, $\sigma \leqq minimum(1/(2\log\varepsilon), 1/(2(\log\varepsilon - \log(1-\xi)))) = 1/(2\log\varepsilon - \log(1-\xi))$. □

Assume we have a set of data items, such that, $Z = \{z_0, z_1, \ldots z_{\mathbb{N}-1}\}$, distributed among the PEs, according to an arbitrary permutation Π of the input set, $A = \{a_0, a_1, \ldots a_{\mathbb{N}-1}\}$, such that $\Pi(a_i) = z_i$. Hence, Z represents the permutation set of the input set A. Let us assume a subpermutation; X such that $X \subset Z$. Therefore, we want to permute (i.e., have the bijection mapping) X on the PEs subset

represented by $\Gamma \subset \Phi$, such that each $PE \in \Gamma$ has one data element; $z \in X$. Hence, according to arbitrary permutation Π, the set of data items corresponding to that permutation is permuted by the PEs, which assigns the data item, located in PE_i, to $\Pi(PE_i)$. Hence, according to this permutation a data item in $PE_i \in \Gamma$, moves a distance Δ, if $\partial(PE_i, \Pi(PE_i)) = \Delta$, (recall that Δ represents the minimum data transfer necessary to transfer the one item from PE_i to $\Pi(PE_i)$). Please note that, we are specifying the distance between two PEs in the subset Γ, but generally, the sequence of PEs used to measure the distance, may be chosen from all elements of the set Φ.

The next step in our derivation is to find out the Hamilitonian cycles, (i.e., each PE_i there is at least PE_j, which is a logarithmic distance away.). We have used the result of the following theorem, from the graph theory[Di52] concerning with; how to detect the Hamiltonian cycles.

Theorem[Di52]

A certain graph contains a Hamiltonian cycle, if all vertices of that graph have a size of $\Phi \geqq 3$, and every vertex has a degree of at least $\varepsilon = (\Phi/2)$. ■

According to the above theorem we can have a Hamiltonian cycles in a certain graph if the number of the vertices of that graph is $\geqq 3$. Therefore, this theorem helps us to construct propos-4, which detects that, for each PE_i there is at least PE_j, which is a logarithmic distance away.

Proposition 4

For any set of PEs such as $\Gamma \subset \Phi$, there is a subpermutation on Γ; denoted as Π, such that each $PE_i \in \Gamma$ transfer its data item a distance Δ, to $\Pi(PE_i)$, where $\Delta > \sigma \log \Gamma$. This can be true if there is σ such that $\sigma = 1/(2(\log \varepsilon + \log 2))$. ■

proof:

Using propos.3 , by letting $\xi = 1/2$, we can find $\sigma = 1/(2(\log \varepsilon + \log 2))$. If we consider the graph whose vertices is the set Γ, and its edges' set is denoted as E, such that, the edge $(PE_i, PE_j) \in E$, if $\partial(PE_i, PE_j) > \sigma \log \Gamma$. Then according to the propo.3, every vertex has a degree of Δ, such that $\Delta \geqq \lfloor \Gamma/2 + 1 \rfloor \geqq \Gamma/2$. Therefore, according the above theorem, we can detect there is Hamiltonian cycle. On this cycle we can find $\Pi(PE_i)$ to be connected to PE_i. □

We have mentioned that the above proposition shows that, any subsets of PEs can be permuted such that each PE's items is transferred to another PE at logarithmic distance. However, if the data items located on Φ processors, are evenly distributed such that every item in one PE, i.e., $\Phi = \Gamma$. Therefore, Prop. 4, can be useful by setting $\Phi = \Gamma$, such that all data items in PE_i are transferred to $\Pi(PE_i)$, according to the permutation Π. But in case we have $\Phi > \Gamma$, (i.e., $\mathbb{N} \gg \Phi$) then we at first, partition the set of the total permutation's elements of the set of the $\mathbb{N}$ input data items located initially in the all PEs of Φ. Therefore, this set which is denoted as; A, is partitioned into different subsets such as, $A_1, A_2, \ldots A_\mu$, where each $|A_i| = \lceil \mathbb{N}/\Phi \rceil$, consists of one item from each PE in $\Gamma_i \subset \Phi$, such that, $1 \leqq i \leqq \mu$ (μ is the maximum number of items in any group of PEs), and $A_i \subset A$. Please note this is also, the same for the output set Z where $Z = \Pi(A)$, according to the permutation function Π. Hence, every subpermutation is assigned to a group of PEs such as Γ_i, where, $\Gamma_i \subset \Phi$. Therefore, for every group of PEs such as Γ_i, we have a set of A_i, which is permuted according to the permutation function Π, to have the output set of Z_i. Then, we apply propos. 4, to each subgroup of PEs; Γ_i, individually. Please note that, we have assigned the large possible set of data items (i.e., greedy algorithm) to Γ_i, and A_i, in condition that, the set A_i or Z_i, has at most one data item for any PE in Γ_i. Therefore, according to such partitioning, and the applying of propos.4, to these different partitions, we should compute then how much time (the least time) is needed to execute the resulting data permutation. Hence, the following proposition is constructed for this purpose.

Proposition 5

The number of times needed to apply prop.4, to have the permutation Π, is at least, $\sigma(\lceil \mathbb{N}/\Phi \rceil) \log (\lceil \mathbb{N}/\mu \rceil)$, where σ has the same value resulted from props.4. ■

proof:

According to propos.4, any data item, a_i, which is needed to be transferred according to the permutation, Π, such that, $\Pi(a_i) = z_i$, needs at least, $\sigma \log |A_i|$, or $\sigma \log |Z_i|$, where, a_i, z_i belongs to A_i, $Z_i = \lceil \mathbb{N}/\Phi \rceil$, respectively. Recall that, $A_i \subset A$ and $Z_i \subset Z$, for $1 \leqq i \leqq \mu$. However, we have Φ data transfers for the first application for propos.4, therefore the number of times to apply propos.4, necessary to have the permutation Π is (at least),

$$\frac{1}{\Phi} \sum_{i=1}^{\lceil \mathbb{N}/\Phi \rceil} \sigma |A_j| \log |A_i| \; ;$$

but in order to mimimize the summation of $|A_i|$ or $|Z_i|$ for all i, where, $1 \leqq i \leqq \lceil \mathbb{N}/\Phi \rceil$, we have applied Lagrange multiplier argument such as:

$$\sum_{i=1}^{\lceil \mathbb{N}/\Phi \rceil} |A_i| \; or \; \sum_{i=1}^{\lceil \mathbb{N}/\Phi \rceil} |Z_i| = \mathbb{N}$$

therefore the number of time becomes, $\sigma(\mathbb{N}/\Phi) \log \lceil \mathbb{N}/\mu \rceil$. □

From these propositions we can have the following lemma which gives us the lower bound needed to have the parallel permutation algorithm executed on any degree ε of the parallel computational model.

Lemma 2

The number of times needed to apply propos.4, to permute the data items among Φ PEs of degree ε is at least, $\sigma(\mathbb{N}/\Phi)\log\Phi$, where σ here is, $1/(3(\log\varepsilon+\log 2))$. ■

proof:

By the application of prop.5, the subpermutation, Π_1, for instance, requires, (by applying prop. 5), at least $(\sigma_1(\lceil\mathbb{N}/\Phi\rceil)\log(\Phi/\mu))$ times, in order to achieve Π permutation, where $\sigma_1=1/(2(\log\varepsilon+\log 2))$, and μ, is the maximum number of items initially located in any one PE. Then, let us have the subpermutation Π_2, such that it needs μ times by prop.4, in order to be represented by PEs. Of course, the time needed to have the permutation Π is more than the time needed for Π_1, and Π_2. Therefore the number of cycles needed to have the permutation Π is at least, $max((\sigma_1(\lceil\mathbb{N}/\Phi\rceil)\log\Phi/\mu),\mu)$. After some calculations, we can find that $(\sigma_1(\lceil\mathbb{N}/\Phi\rceil)\log\Phi/\mu)>\mu$, when $\mu=(2/3)\sigma_1(\mathbb{N}/\Phi)\log\Phi$. Therefore, setting $\sigma=(2/3)\sigma_1$, we can have the lemma to be satisfied. □

From the above results we can conclude that for any permutation algorithm works on a parallel computaional model of Φ PEs where every PE has degree of ε, we have a lower time bound of $\Omega((\mathbb{N}/\Phi)\log_\varepsilon\Phi)$. This bound is because the permutation problem is not completely parallelizable on any degree of parallel computational model, this is because, if $\log\varepsilon=o(1)\log\Phi$ then $\varepsilon=\Phi^{o(1)}$.

We could see the lower bound for the parallel permutation algorithm is as a function of the number of data items needed to be permuted; $\mathbb{N}$, the number of processor consisting the computational model; Φ, and the degree of each processor ε. Therefore, the computational model has a big role to specify how much fast the parallel permutation can be.

We could see the lower bound for the parallel permutation algorithm is as a function of the number of data items needed to be permuted; $\mathbb{N}$, the number of processor consisting the computational model; Φ, and the degree of each processor ε. Therefore, the computational model has a big role to specify how much fast the parallel permutation can be.

From now on, we will discuss the lower bound given in the above result, on the parallel computational model represented by the Cyclic Cube Engine(CCE) given in previous paper[Ha88]. In that paper we have discussed a fast parallel algorithm for the parallel permutation problem in $\Theta(\log N)$ time, assuming $\mathbb{N}\leqq\Phi$, (i.e., one item is per PE). But for the CCE model which has $\varepsilon=3$, we can permute in parallel these $\mathbb{N}$ data items, when we have $\mathbb{N}\gg\Phi$, such that each PE has x data items, for $\mathbb{N}=(x\ \Phi)$. Therefore we want to implement the parallel permutation algorithm which has the lower bound given in the above results, on the CCE model. Hence, we want to show that the CCE, can permute in time of $\Theta((\mathbb{N}/\Phi)\log\mathbb{N})$, when $\mathbb{N}\gg\Phi$ and $\varepsilon\geqq 3$. Because it is clear that for parallel computational model of $\varepsilon=1$ or 2 and $\mathbb{N}=\Omega(\Phi\varepsilon)$, there is no algorithm to permute evenly distributed data items in time of $O((\mathbb{N}/\Phi)\log_\varepsilon\mathbb{N})$. (Please note that, we have discussed previously, the parallel permutation algorithm on CCE when $x=1$.)

Therefore, in order to permute the data items according to the permutation Π, each data item of the set A such as a_i in PE_i, should be transferred or sent to the PE which has the value z_i, as its destination, i.e., $PE(\Pi(a_i))=PE(z_i)$. In order to achieve this, we have represented the permutation Π by two dimensional matrix as $\Phi\times x$, such that the row represents the PEs of the computational model, and the columns represents the number of data items per each PE. Therefore, every PE has x data items from the total items $\mathbb{N}$, where $\mathbb{N}=\Phi x$. Hence, we can do the permutation Π, such that, ① In each PE_i,(of each row in the representation matrix) do the sub permutation on the local data x. ② To each column apply the parallel permutation algorithm given in previous paper x times, for $\mathbb{N}=\Phi$. ③ the same as ①.

The steps; ① and ③ are sequential permutations and need $\Theta(x)$ time. While step ② has to apply the parallel permutation algorithm x times on Φ PEs. Thus the total bound to implement this algorithm is $\Theta(x\log\Phi)$. To the authors information, we do not know of a dynamic permutation algorithm for $\mathbb{N}>\Phi$ and has complexity of $\Theta(x\log\Phi)$.

III.3. The Lower Bound Algorithms on the CCE Model

This section represents, how the parallel permutation algorithms are not completely parallelizable on the CCE model and other parallel computational models. Therefore, here we have analyzed the lower bounds of those algorithms due to its execution on the CCE model using the results of the previous sections, to show that the algorithms are optimal if their computational bounds are within the bounds given in the previous sections.

We have shown that, the CCE model is an $\mathbb{N}$-cycles cubic topology of 2^k-PEs per cycle[Ha88]. We have shown there a control algorithm for our objective network; Benes IN, when $x=1$, (recall that $x=\mathbb{N}/\Phi$; represents the number of data items per processor). Therefore, an arbitrary permutation can be represented or realized in $O(\log\mathbb{N})$ communication steps consisting of cube, inverse cube, and transpositions which are supported by the CCE model, and executed in time of O(1)

because of the direct connection. We have researched here the case when $x>1$, such as when $\mathbb{N} \gg \Phi$. Therefore, how much parallelism can be gotten by Φ PEs configured as CCE.

Let us compute and realize the computational steps needed for permuting $\mathbb{N}$ data items between Φ PEs, such that $\mathbb{N} \gg \Phi$, according to the arbitrary permutation; Π.

The first step in such realization is to set within each PE the x data items of the pair (a_i, z_i) of permutation Π, recall that a; is the input element, $a, z \in \mathbb{N}$, and $z = \Pi(a)$. Therefore, as were given in pervious paper[Ha88], we sort or permute the records of those pairs on each PE according to the field z. As well as we have x data items in each PE, and the destinations of these items are the Φ PEs, therefore, the execution time for such a step is $\Theta(x+\Phi)$.

The next step is to choose the data items which have distinct z. As well as every PE is interpreted as a vertex in a bipartite graph, therefore, the choices of distinct z from every group represents the complete matching in the graph terminology[Di52]. It is known that, the maximal matching in the bipartite graph between the input vertices; $a_{\{0,\ldots x-1\}},\ldots\ldots,a_{\{\Phi-x,\ldots\Phi-1\}}$, and the output vertices, $z_{\{0,\ldots x-1\}},\ldots\ldots z_{\{\Phi-x,\ldots\Phi-1\}}$ represented by the edges' set according to the permutation Π can be executed in time of $\Theta(\Phi^{2.5})$[Di52]. As well as among all PEs there are at most Φ^2 data items which are needed to be relocated according to z. Therefore the total execution time is $\Theta(\Phi^{2.5}(\Phi^2))$.

During these Φ^2 executions of the data items, a total of x permutations are affected, which take a time of $\Theta(x \log \Phi)$. Each PE then takes time of $\Theta(x)$ to permute its local subpermutation. Therefore the total time for the parallel permutation algorithm takes a time of; $\Theta(x+\Phi)+\Theta(\Phi^{4.5})+\Theta(x \log \Phi)+\Theta(x) = \Theta(\Phi^{4.5})+\Theta(x \log \Phi)$, which is optimal.if $\Phi^{4.5}+x \log \Phi=0$, then $x = \Phi^{4.5}/\log \Phi$. This bound computed here, is within the results given in the previous sections.

IV. Conclusions

We have shown that the lower bound for any parallel computational model of degree of ε can permute its N data items with time of $O(\mathbb{N}/\Phi) \log_\varepsilon \Phi$)) in condition that these data items are evenly distributed, $\varepsilon \geqq 3$, and $\mathbb{N} = \Omega(\Phi^\varepsilon)$. Hence, for CCE model of $\varepsilon = 3$, we can permute in parallel, according to arbitrary permutation, $\mathbb{N}$ data items of $\mathbb{N} \gg \Phi$, such that each PE has x data items where $x = \mathbb{N}/\Phi$. However, we have discussed in a previous paper[Ha88], a fast algorithm for the same problem when $\mathbb{N} = \Phi$, such that each item is assigned per PE. That algorithm has a bound of $\Theta(\log \mathbb{N})$ when $\Phi = 2^{n+k}$ PEs where $n = \log_2 \mathbb{N}$, and 2^k is the number of PEs per cycle of the CCE model.

REFERENCES:

[Di52] Dirac, G.A., "Some theorems on abstract graph," Proc. London Math. Soc., 69-81, 1952.

[Ge78] Gentleman, W.M., "Some complexity results for matrix computations on parallel processors," J. ACM, vol. 25, pp. 112-115, Jan., 1978.

[Ha87] Hamid, I.A., et.al.,"A new fast control mechanism for rearrangeable interconnection network useful for supersystems," Trans. IEICE, vol. E70, No. 10, pp. 997-1008, Oct. 1987.

[Ha88] Hamid, I.A., et.al.,"A new Controlling algorithm for Benes interconnection network without symmetry-Construction part," Trans. IEICE, submittd to be published in this issue.

[Hi86] Hillis, D., The connection Machine, MIT press, Camdridge, Massachusetts, 1986.

[Kn72] Knuth, D.E., The art of computer programming-Volume 1/ Fundamental Algorithms, Addison-Wesley Publishing company, 1972, (pp.104-108).

[Op71] Opferman, D.C., et.al., "On a class of rearrangeable switching networks," Bell System Tech. J. vol. 50, pp. 1579-1600, May-June 1971.