JOURNAL OF ALGORITHMS 7, 270–276 (1986)

Conway's Parallel Sorting Algorithm

MAX L. WARSHAUER

Department of Mathematics and Computer Science, Southwest Texas State University, San Marcos, Texas 78666

Received September 1, 1984

We analyze a parallel processor composed of $(N - 1)$ finite state machines which is used to sort N keys. In one "cycle," comparisons and exchanges are made between pairs of adjacent keys. We show that the keys will be sorted after at most $(2N - 3)$ cycles. © 1986 Academic Press, Inc.

The parallel processor we will study was suggested by Conway and communicated to the author by Early. It consists of $(N - 1)$ finite state machines which are used to sort N keys, $K_1, K_2, \ldots, K_N$, stored as m-bit binary words. The ith finite state machine FSM_i (see Fig. 1) is responsible for comparing the ith and $(i + 1)$th words.

On a given cycle, FSM_i will do one of two things:

1. Swap Word_i and Word_{i+1}
2. Not swap Word_i and Word_{i+1}.

If either FSM_{i+1} or FSM_{i-1} assumes the *swap* state on a given cycle, then FSM_i must assume the *no swap* state because a given word can only swap with one neighbor on a given cycle.

A cycle is composed of m phases, one phase for each bit of the words being compared. On the first phase of a given cycle the largest bit of Word_i is sent to FSM_i and FSM_{i-1}. Word_N only sends a bit to FSM_{N-1} and Word_1 only sends a bit to FSM_1.

Let us begin by analyzing the ith finite state machine FSM_i which has received the leading bits from Word_{i+1} and Word_i. FSM_i can assume one of three states: the *no swap* state denoted N_i, the *swap* state denoted S_i, or the *undecided* state denoted U_i. Initially all FSM_i are in state U_i. During a phase of a cycle each FSM_i receives a new bit Bit_i from Word_i and Bit_{i+1}

0196-6774/86 $3.00

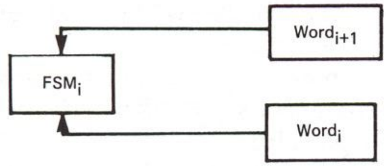

FIGURE 1

from Word$_{i+1}$, and then changes its state according to the following:

Old state	New state
N_i	N_i
S_i	S_i
U_i	(a) If S_{i-1} or S_{i+1}, or if $\mathrm{Bit}_{i+1} > \mathrm{Bit}_i$, then N_i
	else (b) If $\mathrm{Bit}_i > \mathrm{Bit}_{i+1}$, then S_i
	else (c) U_i

On each successive phase, FSM$_i$ will receive the next largest bits of words Word$_{i+1}$ and Word$_i$. As long as FSM$_i$ is in the undecided state U_i, all previous bits of words Word$_{i+1}$ and Word$_i$ are identical. Upon receiving the new bits, FSM$_i$ will assume a new state according to the criteria described above.

Thus, Word$_{i+1}$ and Word$_i$ will circulate in place as long as the finite state machines FSM$_{i+1}$, FSM$_i$, and FSM$_{i-1}$ are in the undecided state. If FSM$_i$ ever assumes the *swap* state S_i then successive bits of Word$_{i+1}$ and Word$_i$ are swapped. Swapping these remaining bits will interchange Word$_{i+1}$ and Word$_i$, since the earlier bits are all equal.

The reader should observe that if FSM$_i$ assumes the *swap* state on a given phase then it is not possible for either FSM$_{i+1}$ or FSM$_{i-1}$ to also assume the swap state on the same phase. This is because if FSM$_i$ assumes the swap state then Bit$_{i+1}$ from Word$_{i+1}$ in the higher position must be a 0, and Bit$_i$ from Word$_i$ in the lower position must be a 1. However, Word$_{i+1}$ is the word in the lower position for FSM$_{i+1}$ and Word$_i$ is the word in the higher position for FSM$_{i-1}$. Consequently neither FSM$_{i+1}$ nor FSM$_{i-1}$ could meet the criteria to assume the swap state on this phase. Further, if FSM$_i$ assumes the swap state S_i, then it sends this message to both FSM$_{i+1}$ and FSM$_{i-1}$ so that both of these finite state machines will assume the no swap state N_i in the next phase. This is because Word$_{i+1}$ and Word$_i$ can only be swapped by one finite state machine during a given cycle.

The question we examine is how many cycles this process requires to ensure that the keys are arranged in descending order. This provides a natural measure of the "time complexity" for a parallel processor which can make comparisons simultaneously.

Let us establish some notation. We denote the keys by $K_1, K_2, \ldots, K_N$. Let $K_i(s)$ denote the position of key K_i after cycle s. Our sorting procedure is finished after cycle s provided

$$K_i(s) > K_j(s) \quad \text{whenever } K_i > K_j.$$

We thus think of the positions as "going up," pictorially:

Position N
Position $N - 1$
⋮
Position 1.

The sorting places higher values in higher numbered positions. We shall make the convention that positions $N + 1, N + 2, \ldots$ are filled by $+\infty$, and positions $0, -1, \ldots$ are filled by $-\infty$. With this convention, we can now make a definition:

DEFINITION 1. We say that key K_i is *attractive up* after cycle s if either of the following conditions is satisfied:

Condition 1. The key in position $K_i(s) + 1$ is larger than K_i.

Condition 2. The key in position $K_i(s) + 2$ is larger than K_i.

Following our convention, it follows that if key K_i is in either position N or position $(N - 1)$ then K_i is attractive up. Similarly we define:

DEFINITION 2. We say that key K_i is *attractive down* after cycle s if either of the following conditions is satisfied:

Condition 1. The key in position $K_i(s) - 1$ is smaller than K_i.

Condition 2. The key in position $K_i(s) - 2$ is smaller than K_i.

LEMMA 1. *Suppose key K_i is attractive up (resp. down) after cycle s. Then key K_i is attractive up (resp. down) after each cycle $t > s$.*

Proof. There are three types of moves key K_i can make on cycle t:

(1) *Move down.* K_i swaps with the key immediately below itself, so $K_i(t) = K_i(t - 1) - 1$.

(2) *Move up.* K_i swaps with the key immediately above itself, so $K_i(t) = K_i(t - 1) + 1$.

(3) *Stationary.* K_i does not move on cycle t, so $K_i(t) = K_i(t - 1)$.

We examine each of these cases. If key K_i makes a move down on cycle t, then the key in position $K_i(t-1)-1$ is larger than K_i. After cycle t this key will be immediately above K_i in position $K_i(t)+1$ after cycle t. Consequently Condition 1 of Definition 1 will be satisfied and K_i will be attractive up after cycle t.

Next assume that K_i is attractive up after cycle $(t-1)$ and makes a move up on cycle t. Then clearly Condition 2 of Definition 1 must have been satisfied after cycle $(t-1)$. Now examine the key in position $K_i(t-1)+2$ (if any). This key is larger than K_i. It cannot move down on cycle t, since K_i makes a move up. If this key makes a move up on cycle t, then Condition 2 will still be satisfied after cycle t. If this key remains stationary on cycle t, then Condition 1 will be satisfied after cycle t. In any case, K_i will remain attractive up after cycle t.

Finally assume that K_i is attractive up after cycle $(t-1)$ and makes a stationary move on cycle t. If Condition 1 is satisfied after cycle $(t-1)$ then clearly K_i will remain attractive up after cycle t. Thus we examine the case that Condition 2 is satisfied and Condition 1 is not satisfied. The keys are then arranged as follows:

The key K_j in position $K_i(t-1)+1$ is smaller than K_i.
The key K_r in position $K_i(t-1)+2$ is larger than K_i.

With this arrangement of keys, K_i will necessarily swap with K_j on cycle t, i.e., K_i will not remain stationary on cycle t, a contradiction. Hence, in any case, if K_i is attractive up on cycle $(t-1)$, then K_i will remain attractive up after cycle t.

Observe that we have also shown:

COROLLARY 2. *If key K_i moves down on cycle t, then key K_i will be attractive up at each cycle $s > t$.*

Since the same proof works for attractive down, we may also state

COROLLARY 3. *If a key K_i moves up on cycle t, then key K_i will be attractive down at each cycle $s > t$.*

Let $[j/2]$ denote the greatest integer less than or equal to $j/2$. We now may state:

LEMMA 4. *Every key whose position is less than or equal to $[j/2]$ is attractive up after j cycles (where $j < 2N$).*

Proof. Suppose key K_i is not attractive up after j cycles. Then by Corollary 2 at each cycle $t \le j$ K_i moves up or remains stationary. Suppose that key K_i makes two successive stationary moves at cycles t and $(t+1)$.

Then one of the following must be true:

(a) The key in position $K_i(t+1)+1$ is larger than K_i and K_i is attractive up, a contradiction; or

(b) $K_i(t+1) = N$, and K_i is attractive up, contradiction.

Thus key K_i will make at least one move up every other cycle, unless it reaches the top position. Hence after j cycles,

$$K_i(j) > [j/2] \qquad \text{for } j < 2N.$$

The lemma follows.

We may similarly state:

LEMMA 5. *Every key whose position is greater than or equal to $N + 1 - [j/2]$ is attractive down after j cycles.*

Let $\lceil N/2 \rceil$ denote the ceiling of $N/2$, i.e.,

$$\begin{aligned} \lceil N/2 \rceil &= N/2 && \text{if } N \text{ is even,} \\ &= (N+1)/2 && \text{if } N \text{ is odd.} \end{aligned}$$

LEMMA 6. *After $(N + \lceil N/2 \rceil - 3)$ cycles, every key is attractive up and down.*

Proof. Suppose that there is a key K_i which is not attractive up. Then we can say the following about K_i:

(1) By Lemma 4, after $2\lceil N/2 \rceil - 2$ cycles, the position of K_i is greater than $\lceil N/2 \rceil - 1$, i.e., $K_i(2\lceil N/2 \rceil - 2) > \lceil N/2 \rceil - 1$.

(2) By Corollary 2, every move K_i makes is either a move up or a stationary move.

(3) *We claim:* On each cycle $t > 2\lceil N/2 \rceil - 2$, K_i makes a move up.

To verify this claim, suppose to the contrary that key K_i makes a stationary move on cycle $t > 2\lceil N/2 \rceil - 2$. Then since K_i is not attractive up, there must be keys in locations $K_i(t-1)+1$ and $K_i(t-1)+2$ which are both smaller than K_i. In order for K_i to be stationary on cycle t, these two keys above K_i must swap positions. It follows that the key in position $K_i(t-1)+2$ before cycle t must be smaller than both K_i and the key in position $K_i(t-1)+1$, and its position is greater than or equal to

$$\lceil N/2 \rceil + 2 \geq N + 1 - (\lceil N/2 \rceil - 1).$$

But the key in position $K_i(t-1)+2$ is not attractive down after $2\lceil N/2 \rceil - 2$ cycles, contradicting Lemma 5. Hence K_i must move up on each cycle $t > 2\lceil N/2 \rceil - 2$.

It follows that after $(2\lceil N/2\rceil - 2) + ([N/2] - 1) = N + \lceil N/2\rceil - 3$ cycles, the position of key K_i will be greater than $(\lceil N/2\rceil - 1) + ([N/2] - 1) = N - 2$, and K_i is attractive up as desired. A similar argument shows that K_i is attractive down after $(N + \lceil N/2\rceil - 3)$ cycles, which completes the proof. We may now state

THEOREM 7. *After at most* $(2N - 3)$ *cycles, all keys will be sorted.*

Proof. After $(N + \lceil N/2\rceil - 3)$ cycles all keys are attractive up and down by Lemma 6.

Claim. On each cycle $t > N + \lceil N/2\rceil - 3$ at least two more keys, one "large" and one "small" will reach their final positions, until done.

To verify the claim, let K_i be the largest key not in its final position after cycle j, where $j \geq N + \lceil N/2\rceil - 3$. K_i is attractive up so K_i must be one place below its final position. Observe that the key in position $K_i(j) + 1$ must be less than K_i else K_i would be in its final position. Further the key in position $K_i(j) + 2$ (if any) must be larger than K_i else K_i would not be attractive up. It follows that K_i will move up on cycle $(j + 1)$ and be in its final position after $(j + 1)$ cycles.

Similarly, the smallest key not in its final position after cycle j is attractive down, and will reach its final position after $(j + 1)$ cycles.

It follows that on each cycle $t > N + \lceil N/2\rceil - 3$ at least two move keys will reach their final positions until all keys are sorted. Thus after at most $[N/2]$ more cycles, $(N + \lceil N/2\rceil - 3) + ([N/2]) = 2N - 3$ cycles total, all keys will be in their final positions.

Observe that although this establishes an upper bound for the number of cycles required to sort N keys, it is by no means clear that this is best possible.

Consider the following example: Place keys in positions $1 - N$ as follows:

Position	Key
N	00
$N - 1$	10
$N - 2$	10
$N - 3$	00
$N - 4$	10
$N - 5$	10
⋰	⋮
3	00
2	10
1	11

The order of keys 00, 10, 10 is repeated until the last three keys which are 00, 10, 11. It easily follows that the largest key, 11, in position 1 requires

$(\frac{4}{3})N - 1$ cycles to reach the top. Thus the best possible result is between $(\frac{4}{3})N - 1$ and $2N - 3$. It seems reasonable to conjecture that after $N + \lceil N/2 \rceil - 2$ cycles (see Lemma 6) all keys will be sorted, but no proof is known.

It would be interesting to know about the average behavior of this algorithm, although little is presently known about this.

Acknowledgment

The author wishes to express his appreciation to the referee for his insightful suggestions.

Reference

1. D. E. Knuth, Sorting and searching, "The Art of Computer Programming," Vol. 3, Addison–Wesley, Reading, Mass. (1973).