FAST PARALLEL ALGORITHMS FOR FINDING HAMILTONIAN PATHS AND CYCLES IN A TOURNAMENT

Danny Soroker*

Computer Science Division
University of California
Berkeley, CA 94720

ABSTRACT.
A *tournament* is a digraph $T=(V,E)$ in which, for every pair of vertices, u & v, exactly one of (u,v) , (v,u) is in E. Two classical theorems about tournaments are that every tournament has a Hamiltonian path, and every strongly connected tournament has a Hamiltonian cycle. Furthermore, it is known how to find these in polynomial time. In this paper we discuss the parallel complexity of these problems. Our main result is that constructing a Hamiltonian path in a general tournament and a Hamiltonian cycle in a strongly connected tournament are both in NC. In addition, we give an NC algorithm for finding a Hamiltonian path with one fixed endpoint. In finding fast parallel algorithms, we also obtain new proofs for the theorems.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **OCT 1986**	2. REPORT TYPE	3. DATES COVERED **00-00-1986 to 00-00-1986**
4. TITLE AND SUBTITLE **Fast Parallel Algorithms for Finding Hamiltonian Paths and Cycles in a Tournament**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Berkeley,Department of Electrical Engineering and Computer Sciences,Berkeley,CA,94720**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		

14. ABSTRACT

A tournament is a digraph T=(V, E) in which, for every pair of vertices, u & v, exactly one of (u, v), (v, u) is in E. Two classical theorems about tournaments are that every tournament has a Hamiltonian path, and every strongly connected tournament has a Hamiltonian cycle. Furthermore, it is known how to find these in polynomial time. In this paper we discuss the parallel complexity of these problems. Our main result is that constructing a Hamiltonian path in a general tournament and a Hamiltonian cycle in a strongly connected tournament are both in NC. In addition, we give an NC algorithm for finding a Hamiltonian path with one fixed endpoint. In finding fast parallel algorithms, we also obtain new proofs for the theorems.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**12**	

FAST PARALLEL ALGORITHMS FOR FINDING HAMILTONIAN PATHS AND CYCLES IN A TOURNAMENT

Danny Soroker*

Computer Science Division
University of California
Berkeley, CA 94720

1. Introduction

A *tournament* is a directed graph $T=(V,E)$ in which, for any pair of vertices, u,v, either $(u,v)\epsilon E$ or $(v,u)\epsilon E$ but not both. This models a competition involving n players, where every player competes against every other one. A trivial, but useful, fact is that any induced subgraph of a tournament is also a tournament. If $(u,v)\epsilon E$ we will say that *u dominates v*, and denote this property by $u>v$. Note that since the directions of the arcs are arbitrary, the domination relation is not necessarily transitive. We extend the notion of domination to sets of vertices: let A,B be subsets of V. *A dominates B* ($A>B$) if every vertex in A dominates every vertex in B.
For a given vertex, v, we categorize the rest of the vertices according to their relation with v : $W(v)$ is the set of vertices that are dominated by v (i.e. vertices involved in matches which v *Won*) and $L(v)$ is the set of vertices that dominate v (matches which v *Lost*).

Much work has been done on tournaments (see, e.g. [1] chapter 7). In this paper we concentrate on two classical results: every tournament has a Hamiltonian path, and every strongly connected tournament has a Hamiltonian cycle. These theorems are in contrast with the fact that deciding if an arbitrary graph is Hamiltonian is NP-complete [4]. The proofs of these theorems in the literature imply efficient algorithms for finding these

*Research supported in part by Defense Advance Research Projects Agency (DoD) Arpa Order No. 4871 Monitored by Naval Electronic Systems Command under Contract No. N00039-84-C-0089

objects, but since the proofs are by induction, the algorithms seem inherently sequential. A natural question is - can Hamiltonian paths and cycles in tournaments be found quickly in parallel? We answer in the affirmative by giving *NC* algorithms for both problems (see [10] for a discussion of *NC* and fast parallel algorithms). In the process of giving the algorithms we demonstrate new proofs of the theorems.

In the first, simpler, part we show how to find a Hamiltonian path. A similar algorithm was discovered independently by Simeon and Joseph Naor [6]. Our solution uses an interesting technical lemma, which states that in every tournament there is a "mediocre" player - one that has both lost and won many matches.

The second part discusses how to find a Hamiltonian cycle, which turns out to be quite a bit more complicated. The main idea in the solution is defining a new problem - that of finding a Hamiltonian path with one fixed endpoint - and solving it simultaneously with finding a Hamiltonian cycle, using a "cut and paste" technique.

The notation we use is mostly standard (see e.g. [3]). We deal only with directed graphs. For a graph, G, $V(G)$ denotes the vertex set and $E(G)$ denotes the arc set. For a set of vertices, U, $G(U)$ denotes the induced subgraph on U. For graphs A, B, $\{A, B\}$ means the union of the two graphs. For paths P, Q, (P, Q) means the concatenation of the two paths.

2. Hamiltonian Path

We start by stating the theorem due to Redei [7] and its textbook proof ([8] page 487).

Theorem 1: Every tournament contains a Hamiltonian path.

Proof: By induction on the number, n, of vertices. The result is clear for $n=2$. Assume it holds for tournaments on n vertices. Consider a tournament, T, on $n+1$ vertices. Let v be an arbitrary vertex of $V(T)$. By assumption $G(V-\{v\})$ has a Hamiltonian path $v_1, v_2, \ldots, v_n$. If $v > v_1$ then $v, v_1, \ldots, v_n$ is a Hamiltonian path of T. Otherwise let i be the largest index such that $v_i > v$. If $i=n$ then $v_1, \ldots, v_n, v$ is a Hamiltonian path. If not, $v_1, \ldots, v_i, v, v_{i+1}, \ldots, v_n$ is the desired Hamiltonian path. $\square$

It is not hard to see that the proof yields an efficient sequential algorithm for finding a Hamiltonian path in a tournament. In order to obtain a parallel algorithm a different method seems to be required. The approach we take is divide and conquer.

A simple-minded way is the following: (i) Split the tournament into two subgraphs, T_1, T_2, of roughly equal order. (ii) In parallel, find Hamiltonian paths H_1 in T_1 and H_2 in T_2. (iii) Connect H_1 and H_2 to form a Hamiltonian path of T.
The problem with this approach is that step (iii) is not guaranteed to succeed, since we have no control over what the endpoints of H_1 and H_2 are.

It turns out that a slightly modified approach does work. The key observation is the following: let v be a vertex of T. Consider Hamiltonian paths $l_1, \ldots, l_k$ of $L(v)$ and $w_1, \ldots, w_m$ of $W(v)$. Since $l_k > v$ and $v > w_1$ we have the following Hamiltonian path of T: $l_1, \ldots, l_k, v, w_1, \ldots, w_l$. Note that this provides an alternative, simpler proof of theorem 1.

In order to derive an NC algorithm from the above idea, we need the following technical lemma:

Lemma 1 (Mediocre player lemma): In a tournament, T, on n vertices there exists a vertex, v, for which both $L(v)$ and $W(v)$ have at least $\lfloor \frac{n}{4} \rfloor$ vertices.

Proof: Let

$$I = \{u \mid d_{in}(u) \geq d_{out}(u)\}$$
$$O = V - I.$$

Assume w.l.o.g that $|I| \geq |O|$. By the pigeonhole principle there exists a vertex, v, whose out-degree in $G(I)$ is no less that its in-degree in $G(I)$. Thus

$$d_{out}(v) \geq \lfloor \frac{|I|}{2} \rfloor \geq \lfloor \frac{n}{4} \rfloor$$

and $d_{in}(v) \geq d_{out}(v) \geq \lfloor \frac{n}{4} \rfloor$ by definition. []

Remark: A simple construction shows that for every n there are tournaments on n vertices for which each vertex has either in-degree or out-degree $\lfloor n/4 \rfloor$.

Using lemma 1 we obtain our algorithm:

```
procedure PATH(T)
    (1) Let $n$ = order of $T$.
    (2) If $n=1$ then return the unique vertex of $T$.
    (3) Find a vertex, $v \epsilon T$, whose in-degree and out-degree in $T$ are both at least $\lfloor n/4 \rfloor$.
    (4) In parallel find $H_1 = PATH(L(v))$ and $H_2 = PATH(W(v))$.
    (5) Return the path $(H_1, v, H_2)$.
end PATH.
```

By lemma 1 only $O(\log n)$ levels of recursive calls (step (4)) are required. The time required for one level is $O(\log n)$, So the total running time is $O(\log^2 n)$. With careful processor allocation the algorithm can be implemented to run using $O(n^2/\log n)$ processors on an *EREW PRAM*. We will not go into details here. Note that this is quite efficient since the size of the input is $\Theta(n^2)$.

3. Hamiltonian Cycle and Restricted Path

The following theorem, due to Camion [2] (see [1] page 173), states exactly when a tournament has a Hamiltonian cycle:

Theorem 2: A tournament is Hamiltonian if it is strongly connected.

Note that the converse is trivially true. The theorem is proven by induction, but note that a similar proof to that of theorem 1 will not work here, since removal of a vertex from a strongly connected tournament might result in a tournament which is not strongly connected. A classical proof, due to Moon [5] (see [1] page 173), proves a stronger claim: a strongly connected tournament on n vertices has a cycle of length k, for $k=3,4,...,n$. We omit the proof.

Again, the proof yields an efficient algorithm, which seems sequential in nature. For our parallel solution we introduce a new notion - a restricted Hamiltonian path.

Definition: A *restricted Hamiltonian path* is a Hamiltonian path with a specified endpoint (either the first or the last vertex, not both).

A natural question is - when does there exist a Hamiltonian path starting (ending) at a given vertex, v? The next theorem gives the precise condition.

Definition: Let T be a tournament and v be a vertex in T. v is a *source* (*sink*) of T if all vertices of T have directed paths from (to) v.

Theorem 3: A tournament, T, has a Hamiltonian path starting (ending) at vertex v if v is a source (sink) of T.

Proof: We prove the theorem for a source. The proof is symmetrical for a sink. The proof is by induction on the n, the order of T. For $n=1$ the claim holds trivially. Assume the claim for tournaments of n vertices. Let T be a tournament of order $n+1$, and let v be a source of T. Using the inductive claim we need only show that $W(v)$ contains a source of $G(V-\{v\})$. By theorem 1, $W(v)$ contains a Hamiltonian path starting at, say, u. Thus u is a source of $W(v)$. Furthermore, by assumption every vertex in $L(v)$ can be reached from some vertex in $W(v)$. Thus u is a source of $G(V-\{v\})$. $\square$

Once again the proof implies a sequential algorithm. Note that here, as in theorem 2, the converse obviously holds.

The key idea for an NC algorithm for finding a Hamiltonian cycle in a strongly connected tournament is to tie it to the problem of finding restricted Hamiltonian paths. The idea is that each problem will be solved by recursive calls to the other. We start by giving an alternative proof for theorem 3, this time using theorem 2. But first, a small lemma.

Lemma 2: Let T be a tournament and let $C_1, C_2, \cdots, C_k$ be its strongly connected components. Then for all i, j either $C_i > C_j$ or $C_j > C_i$.

Proof: By definition of strongly connected components all arcs between C_i and C_j go in the same direction. Since T is a tournament all such arcs exist. $\square$

Second Proof of theorem 3: Let $C_1, C_2, \cdots, C_k$ be the strongly connected components of T such that $C_1 > C_2 > \cdots > C_k$. Since v is a source of T, it must lie in C_1.

Since C_1 is strongly connected, it contains a Hamiltonian cycle, H_1. Let H_2 be the path obtained by deleting from H_1 the unique arc entering v. We note that H_2 is a Hamiltonian path of C_1 starting at v. Let H_3 be a Hamiltonian path of $\{C_2, C_3, \ldots, C_k\}$. By construction, the last vertex of H_2 dominates the first vertex of H_3, so (H_2, H_3) is a Hamiltonian path of T starting at v. $\square$

Now we return to theorem 2 and prove it using theorem 3.

New Proof of theorem 2: Let T be a strongly connected tournament and let $v \epsilon V(T)$. Let $L_1 > L_2 > \cdots > L_q$ be the strongly connected components of $L(v)$ and $W_1 < W_2 < \cdots < W_p$ be the strongly connected components of $W(v)$. Since T is strongly connected there must be some arc leaving W_1. Every such arc must go to a vertex in $L(v)$ (Since, by definition, it cannot go to a vertex in W_i, $i > 1$, or to v). Let

$$m = \min\{i \mid a > b \textit{ for some } a \epsilon W_1, b \epsilon L_i\},$$

and let $w_1 \epsilon W_1$, $l_1 \epsilon L_m$ be such that $w_1 > l_1$. Symmetrically, there must be an arc entering L_1 and let

$$k = \min\{i \mid a > b \textit{ for some } a \epsilon W_i, b \epsilon L_1\},$$

$w_2 \epsilon W_k$, $l_2 \epsilon L_1$ and $w_2 > l_2$. The construction is shown in figure 1.

The existence of a Hamiltonian cycle of T is shown by demonstrating several paths and the connections between their endpoints. These paths are shown in figure 2. The paths are the following:

(1) A Hamiltonian path of W_1 ending at w_1.
(2) A Hamiltonian path of $\{L_m, L_{m+1}, \ldots, L_q\}$ starting at l_1.
(3) The vertex v.
(4) A Hamiltonian path of $\{W_k, W_{k+1}, \ldots, W_p\}$ ending at w_2.
(5) A Hamiltonian path of L_1 starting at l_2.
(6) A Hamiltonian path of $\{W_2, W_3, \ldots, W_{k-1}, L_2, L_3, \ldots, L_{m-1}\}$

We claim that the concatenation of the paths above in the order (1),(2),(3),(4),(5),(6) forms a Hamiltonian cycle of T. First we notice that each of the paths specified does, in fact, exist. For the restricted paths ((1), (2), (4) and (5)) this is a consequence of theorem 3. The only other fact we need to verify is that the arcs between endpoints of the paths are in the desired direction. The only non-obvious cases are the connections from path (5) to path (6) and from path (6) to path (1). For showing this we recall that we chose L_m and W_k in a way that implies that $L_2, L_3, \ldots, L_{m-1}$ all dominate W_1 and $W_2, W_3, \ldots, W_{k-1}$ are all dominated by L_1. Thus the last vertex of path (5) must dominate the first vertex of path (6). Similarly, the last vertex of path (6) must dominate the first vertex of path (1). Notice that both endpoints of path (6) may be either in $L(v)$ or in $W(v)$. $\square$

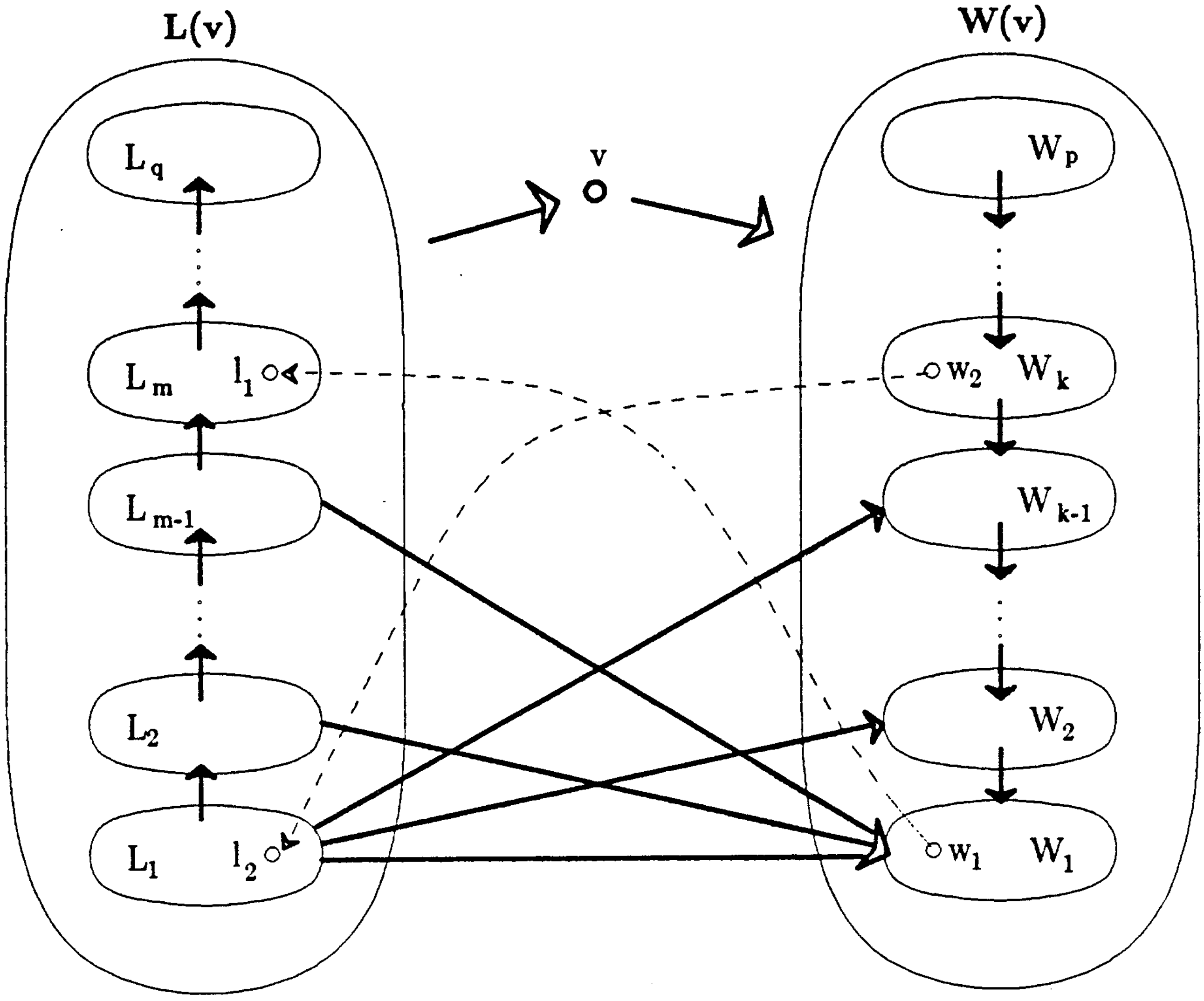

Figure 1 : The construction used in the second proof of theorem 2.

Bold arrows denote all arcs from one component to another (domination of sets).

Dashed arrows denote single arcs (domination of vertices).

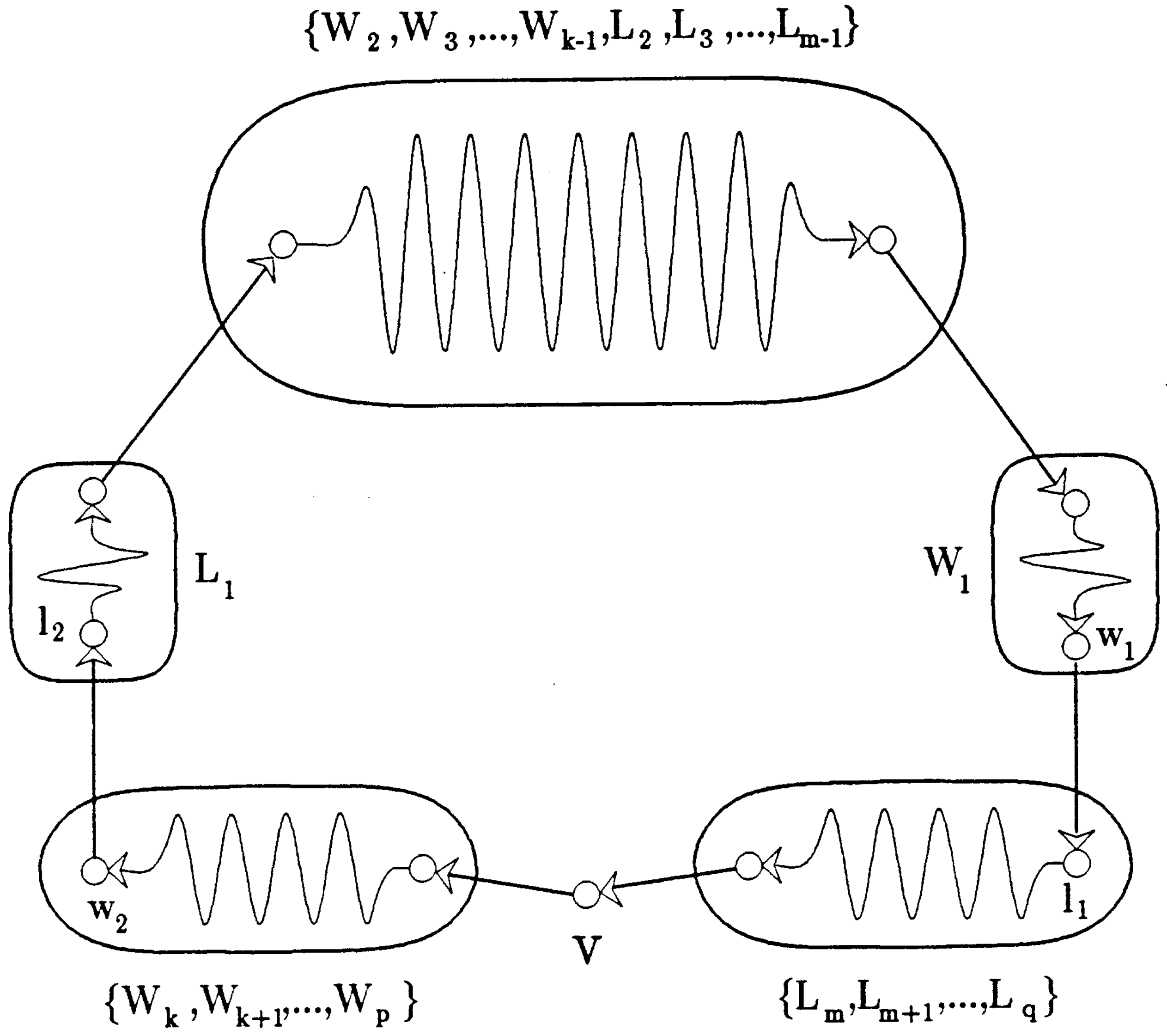
$\{W_2, W_3, \ldots, W_{k-1}, L_2, L_3, \ldots, L_{m-1}\}$
L_1
l_2
W_1
w_1
w_2
l_1
V
$\{W_k, W_{k+1}, \ldots, W_p\}$
$\{L_m, L_{m+1}, \ldots, L_q\}$

Figure 2 : Demonstration of the Hamiltonian cycle described in the second proof of theorem 2. Wiggly arrows denote Hamiltonian paths of the various components.

This new proof gives an approach for an *NC* algorithm - by selecting v to be a "mediocre" vertex we break the problem into several subproblems of bounded size: subgraphs (1), (2), (3), (4) and (5) all have at most $\frac{3}{4}n$ vertices. However, subgraph (6) (the union of components $W_2, \ldots, W_{k-1}$ and $L_2, \ldots, L_{m-1}$) may be very large. In fact it may contain all but five vertices of T, since v, w_1, w_2, l_1 and l_2 are the only vertices guaranteed to be outside of this subgraph.

It turns out that this apparent obstacle is non-existent! The critical observation is that the Hamiltonian path we need to find in (6) is *not restricted*. Therefor we can use procedure *PATH* for finding this path, and need not worry about the size of this subproblem. Thus the problem of finding a Hamiltonian cycle (or restricted Hamiltonian path) on n vertices breaks down into several similar problems, each on no more than $\frac{3}{4}n$ vertices, and one easier problem on at most n vertices.

The algorithms for Hamiltonian cycle and restricted path follow. Note that the solution to the Hamiltonian cycle problem is very symmetrical, as demonstrated in figure 2.

```
procedure RESTRICTED_PATH(T,endpoint,u)
    (1) Let $n$ = order of $T$.
    (2) If $n=1$ then return the unique vertex of $T$.
    (3) Find strongly connected components $C_1>C_2> \cdots >C_k$ of $T$.
    (4) If endpoint='start' then
        (4.1) In parallel find
            $H_1=$ CYCLE($C_1$)
            $H_2=$ PATH($\{C_2, \ldots ,C_k\}$).
        (4.2) Let  $H_1= H_1-\{$unique arc into $u\}$.
    (5) If endpoint='end' then
        (5.1) In parallel find
            $H_1=$ PATH($\{C_1, \ldots ,C_{k-1}\}$)
            $H_2=$ CYCLE($C_k$)
        (5.2) Let  $H_2= H_2-\{$unique arc out of $u\}$.
    (6) Return the path ($H_1$, $H_2$).
end RESTRICTED_PATH.
```

```
procedure CYCLE(T)
    (1) Let $n$ = order of $T$.
    (2) If $n=1$ then return the unique vertex of $T$.
    (3) Find a vertex, $v \epsilon T$, whose in-degree and out-degree in $T$ are both at least $\lfloor n/4 \rfloor$.
    (4) Find strongly connected components  $L_1>...>L_q$ of $L(v)$ and $W_1<...<W_p$ of $W(v)$.
    (5) In parallel find
            $m=\min\{i \mid a>b$ for some $a \epsilon W_1, b \epsilon L_i\}$,
            $k=\min\{i \mid a>b$ for some $a \epsilon W_i, b \epsilon L_1\}$,
        and $w_1 \epsilon W_1$, $l_1 \epsilon L_m$, $w_2 \epsilon W_k$, $l_2 \epsilon L_1$ such that $w_1>l_1$ and $w_2>l_2$.
```

```
(6) In parallel find
        $H_1$=$RESTRICTED\_PATH(W_1, 'end', w_1)$
        $H_2$=$RESTRICTED\_PATH(\{L_m, \ldots, L_q\}, 'start', l_1)$
        $H_3$=$RESTRICTED\_PATH(\{W_k, \ldots, W_p\}, 'end', w_2)$
        $H_4$=$RESTRICTED\_PATH(L_1, 'start', l_2)$
        $H_5$=$PATH(\{W_2, \ldots, W_{k-1}, L_2, \ldots, L_{m-1}\})$
(7) Return the cycle $(v, H_3, H_4, H_5, H_1, H_2, v)$
end CYCLE.
```

The main computation required in one level of the recursion is finding strongly connected components and related operations, which can be done in $O(\log^2 n)$ time using $O(n^3)$ processors (fast matrix multiplication techniques can be applied to reduce the number of processors, e.g. $O(n^{2.81})$ using Strassen's method [9]). Again, there are $O(\log n)$ levels, so the total time is $O(\log^3 n)$ on an *EREW PRAM*.

4. Further Research

We have shown that finding a Hamiltonian path and a restricted Hamiltonian path in a tournament are both in *NC*. A natural question is: what is the complexity of finding a *doubly restricted Hamiltonian path*, i.e a Hamiltonian path from a specified point, a, to another specified point, b. We know how to solve this problem in *NC* if either of the graphs T, $T-\{a\}$, $T-\{b\}$ or $T-\{a,b\}$ is not strongly connected. If all these graphs *are* strongly connected, we do not know if the problem is solvable in polynomial time.

Acknowledgements
I would like to thank Prof. Richard Karp for suggesting the problem and for valuable discussions. Also thanks to Phil Gibbons for finding typing mistakes and for pointing out that *PATH* can be implemented a sub-quadratic number of processors without increasing the running time.

References

[1] Beineke, L.W. and Wilson, R.S. eds. , *Selected Topics in Graph Theory* Academic Press, 1978.

[2] Camion, P. , *Chemins et Circuits Hamiltoniens des Graphes Complets* C.R Acad. Sci. Paris (A) 249 pp. 2151-2152 , 1959.

[3] Chartrand, G. and Lesniak, L. , *Graphs & Digraphs* 2nd ed. Wadsworth & Brooks/Cole , 1986.

[4] Garey, M.R. and Johnson, D.S. , *Computers* and *Intractability* W.H Freeman and Company , 1979.

[5] Moon, J.W. , *Topics on Tournaments* Holt, Reinhart & Winston , 1968.

[6] Naor, J. , *Two Parallel Algorithms in Graph Theory* Hebrew University , Technical Report CS-86-6

[7] Redei, L. , *Ein Kombinatorischer Satz* Acta Litt. Sci. Szeged 7 pp. 39-43 , 1934.

[8] Roberts, F.S. , *Applied Combinatorics* Prentice Hall , 1984.

[9] Strassen, V., *Gaussian Elimination is Not Optimal* Numerische Mathematik 13 pp. 354-356, 1969.

[10] Vishkin, U., *Synchronous Parallel Communication — a Survey* TR 71, Dept. of Computer Science, Courant Institute, NYU, 1983.