NC Algorithms for Antidirected Hamiltonian Paths and Cycles in Tournaments

E. Bampis[*] Y. Manoussakis and I. Milis[†]
LRI, Bât 490
Université de Paris Sud
91405 Orsay Cedex, France

ABSTRACT. Two classical theorems about tournaments state that a tournament with no less than eight vertices admits an antidirected Hamiltonian path and an even cardinality tournament with no less than sixteen vertices admits an antidirected Hamiltonian cycle. Sequential algorithms for finding such a path as well as a cycle follow directly from the proofs of the theorems. Unfortunately, these proofs are inherently sequential and can not be exploited in a parallel context. In this paper we propose new proofs leading to efficient parallel algorithms.

1 Introduction

A simple path in a directed graph is said to be antidirected (ADP) if every two adjacent arcs of the path have opposite orientations, in other words if no two adjacent arcs of the path form a directed path. An antidirected Hamiltonian path (ADHP) in a digraph is a simple antidirected path containing all the vertices. In a similar way, we define an antidirected Hamiltonian cycle (ADHC). The problem of finding an ADHP in an arbitrary digraph is trivially NP-complete. To see that, consider any symmetric digraph (i.e. a digraph in which every arc is in a directed cycle of length two) with an even number of vertices. Clearly, this digraph has an ADHC if and only if it has a (directed) Hamiltonian cycle and it is well-known that this last problem is NP-complete.

A tournament is an orientation of a complete graph, i.e. a digraph such that between each pair of vertices there is exactly one arc. For this particu-

[*]LaMI, Université d'Evry, Bd des Coquibus, 91025 Evry Cedex, France
[†]Fellow of the E.E.C. program Human Capital and Mobility.

lar class of digraphs much work has been done in respect with ADHPs and ADHCs. In [7] and [11] it has been shown that all tournaments have an ADHP, except exactly three tournaments of sizes 3, 5 and 7, respectively. It has been, also, shown that each tournament with no less than sixteen vertices has an ADHC [9] [12] [14]. The proofs of these results imply efficient sequential algorithms for finding ADHPs and ADHCs of complexities $O(n)$ and $O(n^2)$, respectively (n is the number of vertices of the tournament). Such an algorithm for ADHPs is presented in the more general context of [8].

Although many results have recently appeared in the literature on NC algorithms for (directed) Hamiltonian paths and cycles in tournaments [1] [4] [13], there are no analogous results for the antidirected case. Unfortunately, the existing efficient algorithms for constructing ADHPs and ADHCs are implied by inductive proofs and are inherently sequential. A natural question, therefore, is: can ADHPs and ADHCs in tournaments be found by NC algorithms?

In this paper we give a positive answer to the above question by providing NC algorithms for both problems. The development of these algorithms is based on new proofs for the existence of ADHPs and ADHCs. Our algorithms are in the well known CRCW PRAM model, where simultaneous reading is allowed while for simultaneous writing processors are required to write the same value. The complexity of the algorithms presented are $O(\log n)$ time, $O(n/\log n)$ processors for finding ADHPs and $O(\log^2 n)$ time, $O(n^2/\log n)$ processors for finding ADHCs. Thus the first algorithm is optimal and the second one is optimal up to a factor of $\log n$, in respect with the sequential complexities of the problems.

2 Definitions and Notation

Throughout this paper, T_n denotes a tournament with n vertices. A trivial, but useful fact is that any induced subgraph, of a tournament is also a tournament. If v, w are vertices of a tournament T_n then we say that v dominates w if the arc (v, w) exists and denote this relation by $v \rightarrow w$. Note that since the directions of the arcs are arbitrary the domination relation is not necessarily transitive. By $\Gamma^-(v)$ and $\Gamma^+(v)$ we denote the sets of vertices which, respectively, dominate and are dominated by the vertex v. The indegree and the outdegree of v are defined as $|\Gamma^-(v)|$ and $|\Gamma^+(v)|$ and are denoted by $d^-(v)$ and $d^+(v)$, respectively.

Following [11], we say that a vertex v is a starting vertex (resp. an ending vertex) of an antidirected path, if the path is of the form $v \rightarrow v_1 \leftarrow \dots$ (resp. $v \leftarrow v_1 \rightarrow \dots$). If v is both a starting and an ending point we say that v is a **double point**. It follows directly from this definition that one of the extremities of an antidirected path with odd number of vertices,

$v_1 \rightarrow v_2 \leftarrow \cdots \leftarrow v_{2r+1}$, is a double point. To see that, it is enough to examine the arc between its extremities, i.e. if $v_{2r+1} \rightarrow v_1$ (resp. $v_{2r+1} \leftarrow v_1$) then the double point is the vertex v_1 (resp. v_{2r+1}).

We say that a tournament is *transitive* if the domination relation is transitive. Clearly the vertices of a transitive tournament is linearly ordered by the domination relation and there is a unique such order, in which $i \rightarrow j$ if and only if $i < j$. In the following a transitive tournament, with n vertices is denoted by TT_n and its ordered vertex set by $\{1, 2, \ldots, n\}$.

We define, fianlly, a special type of transitive subtournaments that will be used in the sequel.

Definition 1: *A transitive subtournament $TT_p \subset T_n$ is said to be* **nice** *if for each vertex $x \in T_n - TT_p$ there is at least one arc $x \rightarrow i$ for some $i \in TT_p$.*

3 Specified Transitive Subtournaments

Transitive subtournaments of a tournament T_n will be very helpful for the problems examined in this paper because of their following interesting property [11] :

Lemma 1. [11]. *Let TT_n be a transitive tournament with vertex set $\{1, 2, ..., n\}$. (i) If n is even, then TT_n contains an ADHP starting from i and ending in j unless either $i = n$ or $j = 1$ or $\{i, j\} = \{1, 2\}$ or $\{i, j\} = \{n-1, n\}$. (ii) If n is odd, then TT_n contains an ADHP with starting (ending) vertices i, j unless either $i = n$ $(= 1)$ or $j = n$ $(= 1)$ or $\{i, j\} = \{n-2, n-1\}$ $(= \{2, 3\})$.*

Although M. Rosenfeld in [11] was not interested on the complexity of finding the ADHPs in Lemma 1, it is easy to show that, given the order of the vertices in a transitive tournament, no searching is needed to find these paths, i.e. it takes $O(1)$ sequential time, while the rank of the vertices in the ADHPs can be computed in parallel in $O(\log n)$ time using $O(n)$ processors [6].

It is known that the problem of finding a maximum transitive subtournament of a tournament is NP-complete [3]. On the other hand, a maximal transitive subtournament can be found trivially by a greedy sequential algorithm of complexity $O(n^2)$. Unfortunately, we do not know if this last problem is in NC. In return we present in this section an NC algorithm for finding a transitive subtournament TT_p with size at least $\lfloor \log n \rfloor + 1$. This can be done by the following procedure.

```
procedure FIND-$TT_p$ ($T_n$)
    (1) Find the vertex $x_0 \in T_n$ with the maximum degree.
    (2) $TT_1 = \{x_0\}$.
    (3) $T_{n_1} = \Gamma^+(x_0)$.
    (4) For $i = 1, 2, ..., \lfloor \log n \rfloor$ do
        (4.1) Find the vertex $x_i \in T_{n_i}$ with the maximum degree.
        (4.2) $TT_{i+1} = TT_i \cup \{x_i\}$.
        (4.3) $T_{n_{i+1}} = \Gamma^+(x_i)$.
    (5) return a $TT_p$ of $T_n$.
end FIND-$TT_p$.
```

Lemma 2. *For every tournament T_n procedure FIND-TT_p obtains a TT_p, $p \geq \lfloor \log n \rfloor + 1$, in $O(\log^2 n)$ time using $O(n^2/\log n)$ processors.*

Proof: Let x be the vertex of maximum out-degree in a tournament T_n. Clearly, $d^+(x) \geq \frac{n}{2}$. This fact guarantees that the procedure above finds a TT_p with size at least $\lfloor \log n \rfloor + 1$. The main loop of the procedure is repeated $\log n$ times and each time the vertex with the maximum degree must be found. This can be done in $O(\log n)$ time using $O(n^2)$ processors. The rank of the vertices in TT_p can be computed in $O(\log n)$ time using $O(n)$ processors [6]. Therefore, the complexity of the FIND-TT_p procedure becomes $O(\log^2 n)$ time using $O(n^2)$ processors. By applying the well known Brent's principle [5] we can reduce the number of processors to $O(n^2/\log n)$. □

Next, we give an NC algorithm for finding a **nice** transitive subtournament of T_n. In order to obtain such an algorithm the next lemma of D. Soroker [13] is used.

Lemma 3. **[13].** *Every tournament T_n contains a vertex u which dominates, and is dominated by at least $\lfloor \frac{n}{4} \rfloor$ vertices.*

The following procedure uses this Lemma to find out a nice $TT_p \subset T_n$.

```
procedure FIND-NICE-$TT_p$ ($T_n$)
    (1) $TT_p = \emptyset$.
    (2) While $|T_n| \geq 1$ do
        (2.1) Find a vertex $u \in T_n$ whose in-degree and out-degree are
              at least $\lfloor \frac{n}{4} \rfloor$
        (2.2) $TT_p = TT_p \cup \{u\}$.
        (2.3) $T_n = \Gamma^+(u)$.
    (3) return a nice $TT_p$ of $T_n$.
end FIND-NICE-$TT_p$.
```

Lemma 4. *For every tournament T_n, procedure FIND-NICE-TT_p obtains a $TT_p \subset T_n$, with $p \geq \lfloor \frac{\log n}{2} \rfloor + 1$ vertices in $O(\log^2 n)$ time using $O(n^2/\log n)$ processors.*

Proof: It is clear that procedure FIND-NICE-TT_p returns a transitive subtournament of T_n. We know that $\Gamma^+(u) \geq \lfloor \frac{n}{4} \rfloor$ and $\Gamma^+(u) \leq \lfloor \frac{3n}{4} \rfloor$, since, by Lemma 3, both $\Gamma^+(u)$ and $\Gamma^-(u)$ have at least $\lfloor \frac{n}{4} \rfloor$ vertices. Each call of the procedure adds a vertex to TT_p, and we obtain that finally $p \geq \lfloor \frac{\log n}{2} \rfloor + 1$. Furthermore for each vertex $x \in T_n - TT_p$ we know that $x \in \Gamma^-(i)$ for some $i \in TT_p$ and thus the obtained TT_p is a nice one.

The procedure is invoked $O(\log n)$ times and at each time Step (2.1) must be executed. This step can be implemented in $O(\log n)$ time using $O(n^2)$ processors. Thus, similarly with Lemma 2, the parallel complexity of FIND-NICE-TT_p procedure is $O(\log^2 n)$ on $O(n^2/\log n)$ processors. □

4 Antidirected Hamiltonian Path

In this section we prove an NC algorithm for finding an ADHP in a tournament T_n. In order to find such a path the divide and conquer approach is exploited. A simple way to apply this approach is the following:

(i) Split the tournament T_n into two subtournaments T_{n_1} and T_{n_2} of roughly equal order.

(ii) In parallel, find ADHPs X_1 in T_{n_1} and X_2 in T_{n_2}.

(iii) Connect appropriately X_1 and X_2 to form an ADHP in T.

Following this approach Step (iii) does not seem clear, since it is not obvious how to connect X_1 and X_2. Fortunately, the following lemma helps to overcome this difficulty.

Lemma 5. *The following operations on ADPs in a tournament T_n can be done in constant time.*

(i) Add a vertex to an ADP with odd number of vertices.

(ii) Add two vertices to an ADP with even number of vertices.

(iii) Connect two vertex-disjoint ADPs X and Y, each of odd number of vertices, in a new one with vertex set $V(X) \cup X(Y)$.

Proof:

(i) Let $X = x_1 \rightarrow x_2 \leftarrow ... \rightarrow x_{2r} \leftarrow x_{2r+1}$ be an ADP with odd number of vertices in T_n and a vertex $u \in T_n - X$. Without loss of generality assume that x_1 is the double point of X, that is, $x_{2r+1} \rightarrow x_1$. We examine the orientation of the arc between x_1 and u. If $x_1 \rightarrow u$, then the desired path is $x_{2r+1} \rightarrow x_{2r} \leftarrow ... \rightarrow x_2 \leftarrow x_1 \rightarrow u$, else the path is $x_2 \leftarrow ... \rightarrow x_{2r} \leftarrow x_{2r+1} \rightarrow x_1 \leftarrow u$.

(ii) Let $X = x_1 \rightarrow x_2 \leftarrow ... \leftarrow x_{2r-1} \rightarrow x_{2r}$ be an ADP with even number of vertices in T_n and two vertices $u_1, u_2 \in T_n - X$. If u_1 (or u_2) $\leftarrow x_1$ (resp. u_1 (or u_2) $\rightarrow x_{2r}$), then replace X by $u_1(u_2) \leftarrow x_1 \rightarrow x_2 \leftarrow ... \leftarrow x_{2r-1} \rightarrow x_{2r}$ (resp. $x_1 \rightarrow x_2 \leftarrow ... \leftarrow x_{2r-1} \rightarrow x_{2r} \leftarrow u_1(u_2)$). Moreover, if u_1 (or u_2) $\rightarrow x_2$ (resp. u_1 (or u_2) $\leftarrow x_{2r-1}$), then replace X by $x_1 \leftarrow u_1(u_2) \rightarrow x_2 \leftarrow ... \leftarrow x_{2r-1} \rightarrow x_{2r}$ (resp. $x_1 \rightarrow x_2 \leftarrow ... \leftarrow x_{2r-1} \rightarrow u_1(u_2) \leftarrow x_{2r}$). If only one of the vertices u_1, u_2 is added so far to X, then the path formed is of odd number of vertices, and the other vertex can be added by case (i). Otherwise (if no vertex is added to X so far), assume without loss of generality, that $u_1 \rightarrow u_2$. If $x_1 \rightarrow x_{2r-1}$, then replace X by $x_{2r} \rightarrow u_2 \leftarrow u_1 \rightarrow x_{2r-1} \leftarrow x_1 \rightarrow x_2 \leftarrow ... \rightarrow x_{2r-2}$, else replace X by $x_{2r} \rightarrow u_2 \leftarrow u_1 \rightarrow x_1 \leftarrow x_{2r-1} \rightarrow ... \rightarrow x_2$.

(iii) Let $X = x_1 \rightarrow x_2 \leftarrow ... \rightarrow x_{2r} \leftarrow x_{2r+1}$ and $Y = y_1 \rightarrow y_2 \leftarrow ... \rightarrow y_{2s} \leftarrow y_{2s+1}$. Without loss of generality assume that x_1 and y_1 are the double points of X and Y respectively, that is $x_{2r+1} \rightarrow x_1$ and $y_{2s+1} \rightarrow y_1$. We examine the orientation of the arc between x_1 and y_1. If $x_1 \rightarrow y_1$, then the desired path is $x_{2r+1} \rightarrow x_{2r} \leftarrow ... \rightarrow x_2 \leftarrow x_1 \rightarrow y_1 \leftarrow y_{2s+1} \rightarrow y_{2s} \leftarrow ... \rightarrow y_2$, else the path is $x_2 \leftarrow ... \rightarrow x_{2r} \leftarrow x_{2r+1} \rightarrow x_1 \leftarrow y_1 \rightarrow y_2 \leftarrow ... \rightarrow y_{2s} \leftarrow y_{2s+1}$.

It is clear, that all these operations take constant time, since the orientation of a constant number of arcs (2, 10 and 3 in cases (i), (ii) and (iii) respectively) must be examined. □

However, is not obvious yet how Lemma 5 can be exploited, into the framework of the divide and conquer approach, to divide T_n and to combine the obtained in each part ADHPs. In the case of even cardinality tournaments we can handle this problem as it is shown in the following procedure:

```
procedure FIND-EVEN-ADHP($T_n$, $n$ even)
  (1) If $|T_n| \leq 2$ then return an ADHP of $T_n$.
  (2) Split $T$ into two subtournaments $T_{n_1}$ and $T_{n_2}$
          of roughly equal even cardinalities such that $n_1 + n_2 = n$.
  (3) In parallel, find ADHPs $X_1$=FIND-EVEN-ADHP($T_{n_1}$, $n_1$ even)
```

```
                and $X_2$=FIND-EVEN-ADHP($T_{n_2}$, $n_2$ even).
    (4) $X_1 = X_1 - \{u_1\}$, $X_2 = X_2 - \{u_2\}$. {$u_1$ and $u_2$ are extremities of
            the ADHPs $X_1$ and $X_2$ respectively }
    (5) Use Lemma 5(iii) to connect paths $X_1$ and $X_2$ in a new path $X$.
    (6) Add, by Lemma 5(ii), vertices $u_1, u_2$ to $X$ and return an ADHP
            of $T_n$.
end FIND-EVEN-ADHP.
```

Lemma 6. *For every tournament T_n, n even, procedure FIND-EVEN-ADHP obtains an ADHP in $O(\log n)$ time using $O(n/\log n)$ processors.*

Proof: The depth of the recursion tree of the above procedure is $O(\log n)$ (Step (3)) and it can be implemented using $O(n)$ processors. Steps (5) and (6) can be implemented in constant time, by Lemma 5, and the same is obvious for Step (4). The rank of the vertices in the ADHPs can be computed in $O(\log n)$ time using $O(n)$ processors [6]. Thus, by using Brent's principle [5], the parallel complexity of FIND-EVEN-ADHP procedure becomes $O(\log n)$ on $O(n/\log n)$ processors. □

Unfortunately, in the case of odd cardinality tournaments a similar approach does not work. This is due to parity reasons and we proceed in a different way. First, we find a transitive subtournament of T_n consisting of 5 vertices, i.e. a TT_5. Such a transitive subtournament can be found in each tournament T_n, $n \geq 16$ in $O(1)$ sequential time, by applying procedure FIND-TT_p on a subtournament of T_n induced by 16 of its vertices. Next, we consider the subtournament $T_k = T_n - TT_5$, in which we can find an ADHP using FIND-EVEN-ADHP, since k is even. Now, using Lemma 1, we can prove the following:

Lemma 7. *Let T_n, $n > 16$, be a tournament of odd cardinality. Given a TT_5 of T_n and an ADHP in $T_k = T_n - TT_5$, an ADHP in T_n can be constructed in $O(1)$ time.*

Proof: Let $\{1,2,3,4,5\}$ be a TT_5 in T_n and $x_1 \rightarrow x_2 \leftarrow \ldots \leftarrow x_{2s-1} \rightarrow x_{2s}$ be an ADHP in $T_n - TT_5$. We test if one of the extremities of this path, x_1 or x_{2s}, can be inserted into TT_5 in such a way that preserves the transitivity, i.e. to form a TT_6. Each of these vertices can not be inserted in TT_5 if and only if there are two vertices i and j in TT_5, $i < j$, such that $j \rightarrow x_1(x_{2s})$ and $x_1(x_{2s}) \rightarrow i$. It is clear that this test can be done in constant time, since TT_5 is a constant size transitive subtournament. Next, we consider two cases:

Case (i): No extremity can be inserted in TT_5.

If for some vertex i of TT_5, $1 \leq i \leq 4$, $i \rightarrow x_{2s}$, then, by Lemma 1, there is an antidirected path in TT_5 starting from i and the desired path is $x_1 \rightarrow$

$x_2 \leftarrow ... \leftarrow x_{2s-1} \rightarrow x_{2s} \leftarrow [i \rightarrow \cdots$ (ADHP in TT_5 starting from i)], for some i such that $1 \leq i \leq 4$. Similarly, if for some vertex i of TT_5, $2 \leq i \leq 5$, $i \leftarrow x_1$, then, by Lemma 1, there is an antidirected path in TT_5 ending at i and the desired path is $x_{2s} \leftarrow x_{2s-1} \rightarrow ... \rightarrow x_2 \leftarrow x_1 \rightarrow [i \leftarrow \cdots$ (ADHP in TT_5 ending in i)], for some i such that $2 \leq i \leq 5$. If none of these is the case, then $x_1 \rightarrow 1$, for otherwise x_1 can be inserted into TT_5, a contradiction. Then the desired path is $x_{2s-1} \rightarrow ... \rightarrow x_2 \leftarrow x_1 \rightarrow 1 \leftarrow x_{2s} \rightarrow [i \leftarrow \cdots$ (ADHP in $TT_5 - \{1\}$ ending in i)], $3 \leq i \leq 5$.

Case (ii): An extremity can be inserted in TT_5.

Assume that x_{2s} can be inserted into TT_5, the proof being similar in the case of x_1. Now, we consider a TT_6 and the antidirected path $x_1 \rightarrow x_2 \leftarrow ... \leftarrow x_{2s-1}$ in $T_n - TT_6$, which now contains an odd number of vertices. Let us assume, without loss of generality, that x_1 is the double point of this path. It is enough to consider an arc between x_1 and some vertex i of TT_6, $2 \leq i \leq 5$. If $x_1 \rightarrow i$ then the path is [(ADHP in TT_6 ending in i) $\cdots \rightarrow i] \leftarrow x_1 \rightarrow x_2 \leftarrow ... \leftarrow x_{2s-1}$ else the path is $x_2 \leftarrow ... \leftarrow x_{2s-1} \rightarrow x_1 \leftarrow [i \rightarrow \cdots$ (ADHP in TT_6 starting from i)].

Since we consider a constant size transitive subtournament of T_n, i.e. a TT_5, in both cases the direction of a constant number of arcs is examined and the corresponding ADHPs can be found in constant time. □

We can, now, summarize the procedure for finding an ADHP in tournaments of odd cardinality as following:

```
procedure FIND-ODD-ADHP($T_n$, $n$ odd)
  (1) Find a $TT_5$, in $T_n$.
  (2) Find an ADHP in $T_n - TT_5$ using FIND-EVEN-ADHP($T_n$, $n$ even).
  (3) Return an ADHP of $T_n$ using Lemma 7.
end FIND-ODD-ADHP.
```

Steps (1) and (3) can be implemented in constant time, and thus the complexity of the procedure FIND-ODD-ADHP is determined by the complexity of the procedure FIND-EVEN-ADHP. Therefore, next theorem follows from Lemma 6.

Theorem 1. *For every tournament T_n, $n \geq 16$, an ADHP can be found in $O(\log n)$ time using $O(n/\log n)$ processors.*

Note that the complexity of our parallel algorithm for finding ADHPs is optimal with respect to the best known sequential one of complexity $O(n)$.

5 Antidirected Hamiltonian Cycle

A restricted antidirected Hamiltonian path of a tournament T_n, is an ADHP with a specified extremity, either the first or the last vertex, not both. Given a specified extremity, say x, we denote such an ADHP as x-ADHP. Notice that we are not interested in whether the specified extremity is a starting or an ending point.

In [11] M. Rosenfeld has proved that for every vertex x of a tournament T_n, $n \geq 9$, there is an x-ADHP. From a careful reading of the proof of Theorem 3 in [11] it is not hard to see that its arguments can be easily implemented in parallel. In particular, if n is even, then it is enough to find an ADHP in $T_n - \{x\}$. Since this ADHP has an odd number of vertices it is trivial to construct a x-ADHP. If n is odd then we test first if x is an internal vertex in some $TT_4 \subset T_n$. Such a test can be done in $O(\log n)$ time using $O(n^2/\log n)$ processors. If this is the case we find an ADHP in $T_n - TT_4$ and then we can easily construct a x-ADHP. Otherwise, T_n has a special structure which implies directly a x-ADHP. Hence, taking into account Theorem 1, we have the following corollary.

Corollary 1. *For every tournament T_n, $n \geq 9$, and x-ADHP can be found in $O(\log n)$ time using $O(n^2/\log n)$ processors.*

Let us now consider the problem of finding an ADHC in tournaments. Obviously, such a cycle exists only in even cardinality tournaments. It is known that every tournament T_n, n even, $n \geq 16$, has an ADHC. This result was initially proved in [14] for $n \geq 50$. For $n \geq 16$, if T_n contains a TT_6, then a proof can be found in [12], otherwise the proof is given in [9]. We point out, here, that all these proofs use a maximal transitive subtournament of T_n. Unfortunately, as far as we know the problem of finding efficiently in parallel a maximal transitive subtournament of a tournament is open. Consequently, a parallel algorithm for the ADHC problem can not be based on these proofs. In what follows we give a new proof for the existence of ADHCs, using a nice transitive subtournament instead of a maximal one. This new proof is exploited into an efficient parallel algorithm for tournaments with no less than 256 vertices, i.e. a constant number of vertices. For tournaments with less vertices we can use the maximal transitive subtournament approach in constant time.

Lemma 8. *Let T_n be a tournament, $n \geq 256$, n even. Given a nice transitive subtournament TT_p of T_n and an ADHP in $T_k = T_n - TT_p$, an ADHC in T_n can be constructed in $O(\log n)$ time using $O(n^2/\log n)$ processors.*

Proof: Let $TT_p = \{1, 2, ..., p\}$ be a nice transitive subtournament of T_n. We distinguish between two cases depending on the parity of p.

(i) If p is odd, let $x_1 \leftarrow x_2 \rightarrow ... \leftarrow x_{2r} \rightarrow x_{2r+1}$ be an ADHP of $T_n - TT_p$ and let x_1 be its double point. We assume without loss of generality that there are at least two vertices, say i, j in TT_p dominating x_1 (if this is not the case we inverse each arc of T_n and study the new tournament). If for some vertex $u \in \{1,2,...,p-1\}$, $u \rightarrow x_{2r+1}$ then the desired cycle is $i \rightarrow x_1 \leftarrow x_2 \rightarrow ... \rightarrow x_{2r+1} \leftarrow [u \rightarrow \cdots \leftarrow i$ (ADHP in TT_p with starting vertices u and i)]. Otherwise, we consider the vertex x_{2r}. If for some vertex $v \in \{1,2,...,p-1\}$, $x_{2r} \rightarrow v$ then the desired cycle is $i \rightarrow x_1 \leftarrow x_2 \rightarrow ... \leftarrow x_{2r} \rightarrow v \leftarrow x_{2r+1} \rightarrow [u \leftarrow \cdots \leftarrow i$ (ADHP in $TT_p - v$ ending at u and starting from i)]. If this is not the case, then $x_{2r} \rightarrow p$, since TT_p is nice. Then, we consider the path $x_1 \leftarrow x_2 \rightarrow ... \leftarrow x_{2r} \rightarrow p$ and the transitive subtournament $TT'_p = \{x_{2r+1}, 1, 2, ..., p-1\}$. The cycle now is $i \rightarrow x_1 \leftarrow x_2 \rightarrow ... \leftarrow x_{2r} \rightarrow p \leftarrow [t \rightarrow \cdots \leftarrow i$ (ADHP in TT'_p with starting vertices t and i)], where $t \in TT'_p$ and $t \neq p-1$.

The above arguments are valid for $p \geq 5$. By Lemma 4, $p \geq \lfloor \frac{\log n}{2} \rfloor + 1$, that is $n \geq 256$. The implementation can be done in $O(1)$ time using $O(\log n)$ processors, since the cardinality of TT_p is $O(\log n)$.

(ii) If p is even, we examine the vertices in $T_n - TT_p$. If there is a vertex, y, which can be inserted in TT_p, then we consider the transitive subtournament $TT_{p+1} = TT_p \cup \{y\}$, which remain nice and therefore we are in the case (i). If no such a vertex exist, then TT_p is maximal and the proof is given by M. Rosenfeld in [12]. For his proof a restricted x-ADHP must be constructed and this dominates the implementation complexity, that is, by Corollary 1, $O(\log n)$ time using $O(n^2/\log n)$ processors.

Combining the two cases we obtain that this Lemma can be implemented in $O(\log n)$ time using $O(n^2/\log n)$ processors. □

Hence, next Theorem follows from Lemma 8, Lemma 4 (construction of a nice TT_p) and Theorem 1 (construction of an ADHP).

Theorem 2. *For any tournament T_n, $n \geq 256$, an ADHC can be found in $O(\log^2 n)$ time using $O(n^2/\log n)$ processors.*

6 Concluding Remarks

We have shown that the problems of finding an ADHP and an ADHC in tournaments are both in NC. By this work two interesting unsolved questions are addressed:

1) Does the problem of finding a maximal transitive subtournament of a tournament belong to NC? (Recall that the problem of finding a maximum one is NP-complete). If this is true, then employing our results for ADHPs and the proofs given by M. Rosenfeld in [12] we can find ADHCs in tournaments with no less than 16 vertices. An

interesting generalization of this problem is that of finding a maximal acyclic subdigraph of a given digraph or equivalently a minimal feedback vertex set in a digraph.

2) What is the complexity of finding, if one exists, a doubly restricted ADHP in a tournament, i.e. an ADHP where both extremities are specified? In other words, what is the complexity (both sequential and parallel) of testing the antidirected Hamiltonian connectedness of a tournament? A similar question for the (directed) doubly restricted Hamiltonian path has been stated by D. Soroker in [13]. Recently, J. Bang-Jensen, Y. Manoussakis and C.Thomassen [2], answered Soroker's question in the affirmative by presenting a polynomial sequential algorithm, while it is not known if the problem is in NC.

Acknowledgment. The authors wish to express their thanks to an anonymous referee for suggesting various improvements on an earlier version of this paper.

References

[1] E. Bampis, M. El Haddad, Y. Manoussakis and M. Santha, A parallel reduction of Hamiltonian cycle to Hamiltonian path in tournaments, PARLE '93, *Lect. Notes in Comp. Sc.* **694** (1993), 553–560.

[2] J. Bang-Jensen, Y. Manoussakis and C. Thomassen, A polynomial algorithm for Hamiltonian-connectedness in semicomplete graphs, *Journal of Algorithms* **13** (1992), 114–127.

[3] J. Bang-Jensen and C. Thomassen, A polynomial algorithm for the 2-path problem for semicomplete digraphs, *SIAM J. Discr. Math.* (1992), 366–376.

[4] A. Bar-Noy and J. Naor, Sorting, minimal feedback sets and Hamiltonian paths in tournaments, *SIAM J. Discr. Math.* **3** (1990), 7–20.

[5] R. Brent, The parallel evaluation of general arithmetic expressions, *J. ACM* **21** (1974), 201–206.

[6] R. Cole and U. Vishkin, Approximate and exact parallel scheduling with applications to list tree and graph problems, In *Proc. 27th FOCS* (1986), 478–491.

[7] B. Grünbaum, Antidirected Hamiltonian paths in tournaments, *J. Combin. Theory (B)* **11** (1971), 249–257.

[8] P. Hell and M. Rosenfeld, The complexity of finding generalized paths in tournaments, *Journal of Algorithms* **4** (1983), 303–309.

[9] V. Petrovic, Antidirected Hamiltonian circuits in tournaments, In *Proc. 4th Yugoslav Seminar of Graph Theory*, Novi Sad, 1983.

[10] K. B. Reid and E. T. Parker, Disproof of a conjecture of Erdös and Moser, *J. Combin. Theory (B)* **9** (1970), 93–99.

[11] M. Rosenfeld, Antidirected Hamiltonian paths in tournaments, *J. Combin. Theory (B)* **12** (1972), 93–99.

[12] M. Rosenfeld, Antidirected Hamiltonian circuits in tournaments, *J. Combin. Theory (B)* **16** (1974), 234–242.

[13] D. Soroker, Fast parallel algorithms for finding Hamiltonian paths and cycles in a tournament, *Journal of Algorithms* **9** (1988), 276–286.

[14] C. Thomassen, Antidirected Hamiltonian circuits and paths in tournaments, *Math. Ann.* **201** (1973), 231–238.