Discrete Applied Mathematics 36 (1992) 199–201
North-Holland

Note

A linear-time algorithm for finding Hamiltonian cycles in tournaments

Y. Manoussakis*

Université Paris-XI, L.R.I., bât. 490, 91405 Orsay cedex, France

Received 2 April 1990
Revised 7 August 1990

Abstract

Manoussakis, Y., A linear-time algorithm for finding Hamiltonian cycles in tournaments, Discrete Applied Mathematics 36 (1992) 199–201.

We give a simple algorithm to transform a Hamiltonian path in a Hamiltonian cycle, if one exists, in a tournament T of order n. Our algorithm is linear in the number of arcs, i.e., of complexity $O(m) = O(n^2)$ and when combined with the $O(n \log n)$ algorithm of [2] to find a Hamiltonian path in T, it yields an $O(n^2)$ algorithm for searching a Hamiltonian cycle in a tournament. Up to now, algorithms for searching Hamiltonian cycles in tournaments were of order $O(n^3)$ [3], or $O(n^2 \log n)$ [5].

We consider the problem of finding a Hamiltonian cycle in a tournament T with n vertices and $m(=n(n-1)/2)$ arcs. By Camion's [1] well-known theorem, a tournament is Hamiltonian if and only if it is strong. Camion's standard proof leads to an obvious $O(n^3)$ algorithm for finding a Hamiltonian cycle in tournaments.

Here we give a very simple algorithm to transform a Hamiltonian path into a Hamiltonian cycle (if one exists) in our tournament T. Our algorithm is linear in the number of arcs, i.e., of complexity $O(m) = O(n^2)$. When it is combined with the $O(n \log n)$ algorithm of [2] to find a Hamiltonian path in T, it yields an $O(n^2)$

* Research partially supported by PRC Math-Info.

algorithm for finding a Hamiltonian cycle in tournaments. We notice that the algorithm of [2] can be implemented in time $O(n \log n)$ with balanced trees or in time $O(n^2)$ with very simple data structures. Since the latter is sufficient for our purposes, our algorithm is very easy to implement. Previous algorithms for searching Hamiltonian cycles in tournaments were of order $O(n^3)$ [3], or $O(n^2 \log n)$ [5] (which was a parallel algorithm with time $O(\log^2 n)$ using $O(n^2/\log n)$ processors in an EREW PRAM).

In what follows, $T(V, A)$ denotes a tournament of order n with vertex set $V(T)$ and arc set $A(T)$. If x and y are vertices of T, then we say that x dominates y if the arc (x, y) is in $A(T)$.

Algorithm

Input: A tournament T on n vertices.

Output: A Hamiltonian cycle, if any, in T.

(1) We find a Hamiltonian path $x_1, x_2, \ldots, x_n$ of T by using the algorithm of [2]. Next, by backward search of this path, we determine the largest integer k (if any), $3 \leq k \leq n$, such that the arc (x_k, x_1) is in $A(T)$. If k does not exist, then we stop since x_1 has indegree 0 and therefore T is not Hamiltonian. If $k = n$, then we stop since $x_1, x_2, \ldots, x_n, x_1$ is the desired cycle. Otherwise, we initialize $j \leftarrow k$ (j is an auxiliary variable) and we let C denote the cycle $x_1, x_2, \ldots, x_j, x_1$ and P denote the path $x_{j+1}, \ldots, x_n$.

(2) We distinguish two cases (a) and (b) depending on whether x_{j+1} dominates a vertex of C or not.

(a) Suppose first that there exists a vertex on C which is dominated by x_{j+1} and let x_i denote the vertex of C with smallest index having this property (i is defined modulo j). Since x_{i-1} dominates x_{j+1}, we define $C \leftarrow x_1, x_2, \ldots, x_{i-1}, x_{j+1}, x_i, \ldots, x_1$. Furthermore, we set $j \leftarrow j+1$, $P \leftarrow x_{j+1}, \ldots, x_n$, we rename the vertices of C so that x_1 is the successor of x_j on C, and we return to (2).

(b) Suppose next that each vertex of C dominates x_{j+1}. By forward search on P beginning from x_{j+2}, we determine the smallest integer p (if any), $j+2 \leq p \leq n$, that x_p dominates a vertex in C. If there is no such p, then we stop since in this case T has no Hamiltonian cycle. Otherwise, we let x_r (r is defined modulo j) be any vertex of C which is dominated by x_p. This gives the cycle $C \leftarrow x_1, x_2, \ldots, x_{r-1}, x_{j+1}, x_{j+2}, \ldots, x_p, x_r, \ldots, x_1$. If $p < n$, then we set $j \leftarrow p$, $P \leftarrow x_{j+1}, \ldots, x_n$, we rename the vertices of C so that x_1 is the successor of x_j on C, and we return to (2).

End

Time analysis. A Hamiltonian path of T can be found with $O(n \log n)$ operations [2]. Clearly, at most $O(n)$ operations are needed to determine the cycle C in step (1). In order to complete step (2), we check once the arcs between either x_j and C (case (a)), or $\{x_j, \ldots, x_p\}$ and C (case (b)) and consequently this can be done with $O(m)$ operations. It follows that the whole algorithm terminates within at most

$O(m)$ operations, as claimed. We notice that this running time is optimal, i.e., at least $O(m)$ time is required for this problem, as it has been observed in [4].

Implementation. This algorithm was implemented on an IBM-PC computer by my students A. Moreau and C. Leroux, whose assistance was appreciated. The results for transforming a Hamiltonian path into a Hamiltonian cycle suggest an average time performance of $O(n \log n)$ for n substantially large.

Remarks. (1) We can find *either* a Hamiltonian cycle *or* the strong components and their Hamiltonian cycles in T by lightly modifying the above algorithm as follows: In case (1), when x_1 has indegree 0, we restart applying our algorithm on the path $x_2, x_3, \ldots, x_n$. In case (2b), if there is no vertex x_p on P which dominates a vertex of C, then we restart applying the algorithm on the path $x_{j+1}, \ldots, x_n$. We notice that even in this case the time complexity remains $O(m)$.

(2) The above algorithm can also be used for searching a Hamiltonian cycle in semicomplete digraphs, i.e., directed graphs with no nonadjacent vertices.

References

[1] P. Camion, Chemins et circuits hamiltoniens des graphs complets, C.R. Acad. Sci. Paris (A) 249 (1959) 2151-2152.

[2] P. Hell and M. Rosenfeld, The complexity of finding generalized paths in tournaments, J. Algorithms 4 (1982) 303-309.

[3] C. Morrow and S. Goodman, An efficient algorithm for finding a longest cycle in a tournament, in: Proceedings 7th Southeastern Conference on Combinatorics, Graph Theory and Computing (Utilitas Math., Winnipeg, 1976) 453-462.

[4] D. Soroker, Fast parallel algorithms for graphs and networks, Ph.D. Thesis, Rept. UCB/CSD 87/390, Computer Science Division, University of California, Berkeley, CA (1987).

[5] D. Soroker, Fast parallel algorithms for finding Hamiltonian paths and cycles in a tournament, J. Algorithms 9 (1988) 276-286.