NOTES

EDITED BY DENNIS DETURCK, DAVID J. HALLENBECK, AND RODICA SIMION

A Note on Finding a Strict Saddlepoint

DANIEL BIENSTOCK* FAN CHUNG* MICHAEL L. FREDMAN*
ALEJANDRO A. SCHÄFFER†‡ PETER W. SHOR† SUBHASH SURI*
*Bellcore, Morristown, NJ 07960
†AT & T Bell Laboratories, Murray Hill, NJ 07974
‡On leave from Department of Computer Science, Rice University

Given an $m \times n$ matrix A, where $m \geqslant n$, a (*strict*) *saddlepoint* (SP) of A is an entry that is (strictly) maximum in its row and (strictly) minimum in its column. Saddlepoints arise in the theory of two-person zero-sum games. Recently, Llewellyn et al. [2] showed that finding a non-strict SP requires querying all mn entries of the matrix in the worst case, while a strict SP can be found by querying just $O((m/n)n^{\log_2 3})$ entries. Assuming that querying an entry of the matrix takes constant time, their algorithm finds a strict SP in time $O((m/n)n^{\log_2 3})$. The purpose of this note is to describe an algorithm that finds a strict SP of A in time $O(m \log n)$, with just $O(m)$ queries of the entries of A.

For simplicity, we assume that A is a square matrix; a rectangular matrix of size $m \times n$ is handled by dividing it into $\lceil m/n \rceil$ matrices of size $n \times n$, as in [2]. The following observation is from [2].

LEMMA 1. *Given two entries of A, we can eliminate one of them as a possible candidate for a strict SP by querying one more entries of A and doing a constant amount of extra computation.*

It follows from this lemma that a matrix can have at most one strict SP. Let H be a collection of n triples of the form (*row*, *column*, *value*) satisfying the following properties:

P1. *H has at most one entry from each row or column of A.*
P2. *Any strict saddlepoint of A lies in a row and column represented in H.*

LEMMA 2. *If a and b are, respectively, the minimum and maximum values of entries in H, then any strict saddle point has a value c, $a \leqslant c \leqslant b$, with equality only if it is a member of H.*

Proof. A strict saddlepoint must either be the representative of H in its row or else exceed it. Similarly it must either be the representative of H in its column or else be less than it. ■

It follows immediately from Lemma 2 that:

LEMMA 3. *If A_{ij} is a minimum (maximum) element of H, then it is the only possible strict SP in column i (row j).*

We initially set H to be $\{(i, i, A_{ii}) | 1 \leqslant i \leqslant n\}$. The following lemma essentially gives an algorithm for finding the strict saddlepoint of A, if it exists.

LEMMA 4. *Let* (i, j, A_{ij}) *and* (k, l, A_{kl}) *be two distinct entries of* H *having minimum and maximum values, respectively. By querying* A_{il} *and doing a constant amount of extra computation, we can reduce the size of* H *by one, while preserving properties* $P1$ *and* $P2$.

Proof. By property P1, $i \neq k$ and $j \neq l$. We say a row or column is *remaining* if it has a representative in H. We divide the analysis into three cases depending on the value of A_{il}.

Case 1. $A_{il} \leqslant A_{ij} \leqslant A_{kl}$. Any strict saddlepoint in column l is no larger than A_{il}. However, $A_{il} \leqslant A_{ij}$, so by Lemma 2 column l cannot contain a strict SP, and we can eliminate this column entirely. By Lemma 3, the only possible strict SP in row k is A_{kl}, which we have already ruled out. Consequently, we can delete the entry (k, l, A_{kl}) from H.

Case 2. $A_{ij} \leqslant A_{kl} \leqslant A_{il}$. This case is symmetric to case 1; here we eliminate row i and column j and, consequently, delete the entry (i, j, A_{ij}) from H.

Case 3. $A_{ij} < A_{il} < A_{kl}$. By Lemma 3, the only possible strict saddlepoints in column j or row k are A_{ij} and A_{kl}. The first inequality rules out A_{ij}; the second rules out A_{kl}. Hence, the column j and the row k can be eliminated. The row i and the column l, however, cannot be eliminated yet. We, therefore, delete (i, j, A_{ij}) and (k, l, A_{kl}) from H but insert (i, l, A_{il}). This preserves the properties of H while decreasing its size by one.

This completes the proof. ■

Once we have eliminated all but one entry as a possible saddlepoint, then to test whether this last entry really is a saddlepoint, we make comparisons with all other entries in its row and column; this requires $2n - 2$ additional queries. Finally, we can store H as a *min-max heap* so that the operations Delete-Min, Delete-Max, Find-Min, Find-Max and Insert can be performed in worst-case time $O(\log n)$ [1]. This proves our main result.

THEOREM 5. *Given an* $m \times n$ *matrix* A, *where* $m \geqslant n$, *we can determine whether* A *has a strict saddlepoint, and report such an entry, by querying* $O(m)$ *matrix entries and doing* $O(m \log n)$ *additional computation.*

REFERENCES

1. M. D. Atkinson, J.-R. Sack, N. Santoro, and T. Strothotte, Min-max heaps and generalized priority queues, *Comm. ACM*, 29 (1986) 996–1000.
2. D. C. Llewellyn, C. Tovey, and M. Trick, Finding saddlepoints of two-person, zero sum games, *Amer. Math. Monthly*, 95 (1988) 912–918.

Circumscribed Circles

ROBERT OSSERMAN
Mathematics Department, Stanford University, Stanford, CA 94305

In a paper giving a new derivation of the four-vertex theorem [1], I stated without proof some elementary lemmas about circumscribed circles. The arguments needed are familiar to those who work in the field, but are not completely obvious. In rethinking those arguments, I noticed that the statements hold in far