Saddle Points in Random Matrices: Analysis of Knuth Search Algorithms

Micha Hofri
Dept. of Computer Science
Rice University
Houston TX 77005
e-mail: hofri@cs.rice.edu

Philippe Jacquet
INRIA, Domaine de Voluceau -
Rocquencourt - B.P. 105
78153 Le Chesnay Cedex, France
e-mail: Philippe.Jacquet@inria.fr

Abstract

We present an analysis of algorithms for finding saddle points in a random matrix, presented by Donald E. Knuth as exercise 1.3.2-12 in *The Art of Computer Programming*. We estimate the average computing costs of three saddle point search algorithms. Amusingly, the asymptotic results in this analysis about matrix saddle points uses the same approach that leads to the celebrated saddle point method in complex analysis.

1. Introduction

A matrix saddle point is defined in [3] as an element of the matrix which is the smallest in its row, and the largest in its column. Exercise 1.3.2-10 there shows two algorithms to find such saddle points, and suggests a third.

In this paper we respond to exercise 1.3.2.12 there, and analyze the average cost of these three saddle point search algorithms. Some of the results are only given explicit form asymptotically, as the numbers of both rows and columns tend to infinity.

Note: while we selected here a direct approach to derive the asymptotics, by a curious coincidence they also follow from recent depoissonization results of the second author and Wojciech Szpankowski [2], results which were obtained via the saddle point method of complex analysis. This is shown in an Appendix.

The calculations below concern the random variables one needs in order to compute expected processing times. In particular, unlike the custom in [3], we do not count the instructions and their execution times in a MIX program, to determine the multipliers of these variables.

We consider matrices with distinct values. The scenario where values may be repeated introduces additional difficulties, which are not addressed here, except in the last section. We do not quite follow the way the problem is presented—a matrix with distinct elements with all its permutations equally likely—since one can avoid looking at all possible rearrangements of a fixed set of elements by using the following result:

Theorem 1: Let an algorithm operate on an $m \times n$ numerical matrix in such a way that only order-relations between the terms affect its operation (never the actual values). Then the same probabilistic behavior is observed under the following two scenarios:
(1) All the matrix elements are distinct, and all $(mn)!$ possible rearrangements of the elements are equally likely.
(2) The matrix elements are drawn independently using the same continuous, atom-less distribution.

The proof is direct: since a matrix can be associated with a sequence in a consistent way (e.g. by catenating successive rows), and each sequence constructed by method (2) can be naturally related to a permutation from the symmetric group S_{mn} (using the relative order of its elements), we only need to show inductively on the size of the sequence that each such member is obtained in probability $1/(mn)!$ – an immediate calculation. $\square$

2. Notation and Assumptions

We denote the number of rows and columns by m and n respectively. Rows are numbered 1 through m etc. The values of the minimum element of row i and the maximum element of column j are denoted by R_i and C_j, respectively. In our model they are random variables.

The operations of finding row-minima or column-maxima in each of the algorithms are independent of each other, and each takes a time as discussed in [3, §1.2.10], over n or m entries. These durations are denoted by T_n and T_m respectively. It is known that $E[T_n] = \alpha + \beta n + \nu \ln n$, where the Greek letters represent instruction counts and times that depend on the implementation.

We use the abbreviations *pdf* for probability distribution function and *df* for such a density function.

Let the *pdf* from which the elements are drawn be $F(\cdot)$ with the corresponding density $f(\cdot)$. Since any continuous atom-less *pdf* would do, we could pick a particular one, even the ever-so-convenient $U(0,1)$. Keeping the symbol F makes some of the calculations more transparent. An arbitrary matrix term is denoted by X.

2.1 The minmax and maxmin inequality

The following theorem is shown in [3] by elementary considerations:

Theorem 2: For an arbitrary numerical matrix let *minmax* be the minimum value of all columns maxima, and similarly, let *maxmin* be the maximum value of all row minima. Then

- *maxmin* $\leq$ *minmax;*
- The matrix has a saddle point if and only if *minmax* = *maxmin*, and then this is the value of the saddle point.
- A matrix with distinct terms can have one saddle point at most. $\square$

2.2 Saddle point probability

The likelihood that a saddle point occurs in a distinct-values matrix is computed in [3], and is shown there to be quite small:

$$P_{mn} \equiv \Pr[\text{An } m \times n \text{ matrix has a saddle point}] = \frac{m+n}{\binom{m+n}{n}}. \tag{1}$$

This quantity is nm times the probability of occurrence of a saddle point at any given position in the matrix.

The last observation allows us to rederive the result in [3] from the equivalent model, by theorem 1. Let x be the value of a specific element. The probability that the other $n-1$ terms of its row are greater than x is $(1-F(x))^{n-1}$. Similarly, the probability that the other $m-1$ terms in its column are smaller than x is $F(x)^{m-1}$. Therefore the probability of occurrence of saddle point in the specified position is $\int_{-\infty}^{\infty} F(x)^{m-1}(1-F(x))^{n-1} f(x)dx = \frac{(m-1)(m-2)\cdots 1}{n(n+1)\cdots(n+m-1)}$. Multiplying by nm reproduces the above result.

3. Knuth's saddle point search algorithms

These algorithms come with no claim of being optimal or even particularly good. They are used as tutorial programming examples in *The Art of Computer Programming*. Correspondingly, the following should be considered an exercise in analysis of algorithms, in the sense that although the analyses point to various possible improvements in the algorithms, we rarely follow these issues.

Note: We discuss some of these aspects briefly in the last section. A more sophisticated approach to searching for saddle points is described in [1].

3.1 Algorithm I: row-oriented

At the level of detail needed here, this algorithm does the following:

1. For rows $i = 1$ through m,
2. Locate minima R_i, (only one is found in the current scenario)

3. Compare R_i against elements in its column:

4. If a larger element is found, continue to next row,

5. Else return the minimum and its location: a saddle point has been found. Exit.

3.2 Algorithm II: two-phase algorithm

Phase one: scan columns.

1. For columns $j = 1$ through n,

2. Locate maxima C_j, (only one is found in the current scenario)

3. Compare C_j against the smallest found so far, and keep the record in μ.

At the end of this phase the variable μ holds the value of *minmax*.

Phase two: scan rows.

4. For row $i = 1$ through m,

5. Search the row for an element smaller than μ.

6. If the search succeeds, continue to the next row.

7. If the search fails, μ is the value of the saddle point; by virtue of theorem 2 μ itself is found in that row. Exit.

3.3 Algorithm III: algorithm II + theorem 2

The third algorithm has two phases like algorithm II, and the first phase is nearly identical with the first one there: the only difference is that in addition to recording the smallest column maximum found so far in μ, it also records the row in which each new record is found, denoted by ρ.

By the comment at the end of the previous subsection, it suffices, during phase 2, to check that μ is indeed the smallest term of row ρ.

4. Average case analysis of the search algorithms

4.1 Analysis of the first algorithm

Locating each row minimum R_i is independent of all other operations and requires the duration T_n. Now we examine the number of comparisons of R_i with elements in its column, denoted by D_i.

Consider row 1. Since R_1 is a minimum of n *iid* F-random variables, its *pdf* is $1-(1-F(x))^n$. Since the first comparison is of R_1 with itself, $D_1 \geq 2$. We compute the distribution of D_1. This is easy to do exactly. We shall then make an approximation, and claim that up to a detail we treat explicitly, the same distribution applies for all the other R_i. The detail is that while each R_i is compared with the terms in its column, if a larger value is not found in the first $i-1$ trials, the next comparison is "dummy" – when R_i is compared with itself – since it never terminates the search.

Concerning the need for approximation: If in any of the first $i-1$ rows the local row-minimum was found in the same column as R_i, we know that it might have been compared with R_i before. This introduces a dependence between the comparisons. We rid ourselves of this complicated dependence by a simple device: we modify the algorithm. We assume that it does *not* terminate when a saddle-point is found, as line 5 in §2.1 now says; instead, the algorithm always examines all m row-minima. This accomplishes the goal, since now the fact that we reached row i provides no information about the outcomes of the comparisons done in the previous searches, making the above claim that the D_i have (essentially the same) independent distributions precise.

What is the effect of the modification? It only changes the operation of the algorithm when a saddle point is found. This event is so rare, that the relative error it introduces, on the order of the probability given in (1), is sufficiently small that it can be safely disregarded. We illustrate this claim by an example below.

No calculations are needed to see by symmetry that $u_1 \equiv \Pr[D_1 = 2] = \Pr[R_1 < X] = n/(n+1)$. If R_1 survives the comparison, in probability $v_1 \equiv 1 - u_1 = 1/(n+1)$, it is the second order-statistic among $n+1$, and has the density $f^{(2)}_{n+1}(x) = n(n+1)F(x)f(x)\,(1-F(x))^{n-1}$. We proceed in this way to find successive $f^{(j+1)}_{n+j}(x)$, the conditional *df* of R_1 following $j+1$ comparisons (including the "dummy" one), given by

$$f^{(j+1)}_{n+j}(x) = \binom{n+j}{j,1,n-1} f(x)F^j(x)\,(1-F(x))^{n-1}, \tag{2}$$

and v_{j+1}, the probability that a variable with this distribution is larger than X, is given by

$$v_{j+1} = \frac{j+1}{n+j+1}, \qquad (v_0 \equiv 1). \tag{3}$$

With the natural definition $u_{j+1} \equiv 1 - v_{j+1}$ we have

$$p_{r-1} \equiv \Pr[D_1 = r] = v_1 v_2 \ldots v_{r-2} u_{r-1} = \frac{n}{r-1}\binom{n+r-1}{n}^{-1}, \qquad 2 \le r \le m. \tag{4}$$

And our assumption is that for row i, $2 \le i \le m$ the following holds:

$$\Pr[D_i = r] = \begin{cases} p_r & 1 \le r < i \\ 0 & r = i \\ p_{r-1} & i < r \le m \end{cases}. \tag{5}$$

These probabilities do not sum to one: if $D_i > m$ we declare that R_i is a saddle point

We use this approach to compute once again the probability of such a matrix having a saddle point. It provides an example for our claim about the error our modification of the algorithm introduces. We compute the complementary probability, of the event $\{\text{all } D_i \le m\}$, none overflows. For any row i we have $\Pr[D_i \le m] = 1 - v_1 \ldots v_{m-1} = 1 - \frac{(m-1)!}{(n+m-1)!/n!}$, hence

$$\Pr[\text{no saddle point}] \approx \left(1 - \frac{(m-1)!n!}{(n+m-1)!}\right)^m = 1 - P_{mn}\left(1 + O(mP_{mn})\right), \tag{6}$$

where P_{mn} is the small probability given in (1).

One can define a random variable, R, the number of rows the unmodified algorithm uses, but again, unless the matrix is very small, it equals m with a probability so close to 1 that it is not very useful.

The expected values of the D_i are directly available from equations (4) and (5):

$$E[D_i] = \sum_{r=1}^{i-1} r p_r + \sum_{r>i} r p_{r-1} = \frac{n}{n-1} + \frac{1}{\binom{n+i-1}{i-1}}. \tag{7}$$

Over the execution of the algorithm i varies from 1 through m, and the expected overall (modified) number of comparisons is then

$$E[D] = \sum_{i=1}^{m} E[D_i] = \frac{n(m+1)}{n-1} - \frac{n}{n-1}\frac{1}{\binom{m+n-1}{m}}. \tag{8}$$

Except for very small m and n, the second term above is negligible.

It is interesting to see if this analysis can suggest whether this algorithm, when used on non-square matrices, should follow the above outline, or "rotate" the matrix and exchange the roles of rows and columns, assuming no other considerations (such as virtual storage management) intervene. Surely the algorithm logic is invariant under such a rotation (this holds, *mutatis mutandis*, for all three algorithms). The total times in both cases can be written as $T_{rows} = mT_n + T(D_{rows})$ and $T_{columns} = nT_m + T(D_{columns})$. Adding more detail for the expectations, with appropriate multipliers (α' represents all the fixed overhead needed at each row/column, but we disregard the setup costs that would be the same in each orientation),

$$E[T_{rows}] = m(\alpha' + \beta n + \nu \ln n) + \delta\left(\frac{n(m+1)}{n-1} - \frac{n}{n-1}\frac{1}{\binom{m+n-1}{m}}\right), \tag{9}$$

compared with

$$E[T_{columns}] = n(\alpha' + \beta m + \nu \ln m) + \delta\left(\frac{m(n+1)}{m-1} - \frac{m}{m-1}\frac{1}{\binom{m+n-1}{n}}\right). \tag{10}$$

The leading term is the same. The last term is negligible, and otherwise the difference can be written as

$$E[T_{rows}] - E[T_{columns}] = (m-n)\left(\alpha' - \delta\frac{mn-1}{(n-1)(m-1)}\right) + \nu \ln\frac{m^n}{n^m}. \tag{11}$$

Typically $\alpha' > \delta$; also the multiplier of δ is nearly 1, and that of ν is positive when $n > m$; since we also expect α' to exceed ν, it seems the algorithm should proceed along the shorter dimension (i.e. as above, if $n > m$) – but actual estimates of these parameters would be needed to make a definite call.

4.2 Analysis of the second algorithm

The first phase requires time of $nT_m + T_n$ and it finds the *minmax* value, μ. The T_n accounts for selecting μ from the successively determined C_j. Since this is not the same as a direct search over n entries, the instruction counts in this T_n would differ, but the same variables appear.

In the second phase the scan of any given row stops as soon as a term strictly smaller than μ is found. We now compute the distributions of lengths of these scans. The analysis requires the distributions of the relevant random variables: the *minmax* with value μ, the column maxima C_j with values $(x_1, \ldots, x_n)$, which we condition on μ, and row elements. The same algorithmic modification as before, making the algorithm search m rows even when a saddle point is discovered, makes the distribution of the inspected row elements independent of their row index, but once we condition them on $(C_1, \ldots, C_n) = (x_1, \ldots, x_n)$, the distributions depend on their column indices. The unconditional *pdf* of any C_j is $\Pr[C_j \le x] = F^m(x)$, and that of μ is $\Pr[\mu \le x] = 1 - (1 - F^m(x))^n$.

The distribution function of C_j conditioned on the value of *minmax* being μ, and on its occurring in another column, equals

$$c_j(x \,|\, \mu) \equiv \Pr[C_j \le x \,|\, C_j > \mu] = \frac{F^m(x) - F^m(\mu)}{1 - F^m(\mu)}, \qquad (x \ge \mu). \tag{12}$$

Finally, the conditional distribution of a matrix term in position (i, j) is given by

$$\Pr[A_{ij} \le x \,|\, C_j = x_j] = F(x)/F(x_j).$$

We consider now the scan of an arbitrary row. Let $q_i(\mu)$ be the probability that the scan overflows column i, *i.e.,* at least the first i scanned terms there are all greater than or equal to μ; clearly $q_0(\mu) = 1$. Let us look at $q_1(\mu)$; it equals $\Pr[A_{.1} \ge \mu]$. Computing it is easy using the above conditional distribution of $A_{.1}$ given $C_1 = x_1$, and the distribution of C_1 given μ. There are additional considerations due to the following possibilities:

(a) – in probability $1/m$ the first term in the row is C_1, then $A_{.1} = C_1 \ge \mu$ and the scan continues, and

(b) – in probability $1/n$ the *minmax* falls in the first column, then $C_1 = \mu > A_{.1}$ and the scan ends, except in the case covered in part (a) (and then $C_1 = \mu = A_{.1}$).

Hence $q_1(\mu) = 1/m + (1 - 1/n)(1 - 1/m)\Pr[A_{.1} \ge \mu | V]$, where V denotes the event $\{$*minmax* is not in the first column, and C_1 is not in the scanned row$\}$. This last probability equals

$$\int_{x_1 \ge \mu} \left(1 - \frac{F(\mu)}{F(x_1)}\right) m \frac{F^{m-1}(x_1)}{1 - F^m(\mu)} f(x_1)\, dx_1,$$

and we find

$$q_1(\mu) = \frac{1}{m} + (1 - \frac{1}{n}) \left(\frac{1 - F(\mu)}{1 - F^m(\mu)} - \frac{1}{m}\right). \tag{13}$$

For $q_j(\mu)$, $1 \le j \le n$, we have a generalization of this result by noticing that the only way computations for different columns interact is via the above consideration (b). We use in (14) below the unusual notation $(1-\frac{1}{n})^*$, where its power $(1-\frac{1}{n})^{*k}$ stands for $(1-\frac{1}{n})(1-\frac{1}{n-1})\cdots(1-\frac{1}{n-k}) = 1-\frac{k}{n}$. This is the probability[1] that a given subset of k columns does not contain *minmax*. A consideration of the possible configurations of events that lead the scan to go beyond the first j columns allows us to write, even without the need to integrate over the maxima $x_1,\ldots,x_j$ explicitly,

$$\begin{aligned} q_j(\mu) &= \left(\frac{1}{m}+\left(\frac{1-F(\mu)}{1-F^m(\mu)}-\frac{1}{m}\right)(1-\frac{1}{n})^*\right)^j && (14)\\ &= \left(\frac{1-F(\mu)}{1-F^m(\mu)}\right)^j\left(1-\frac{j}{n}+\frac{j}{nm}\frac{1-F^m(\mu)}{1-F(\mu)}\right). && (15)\end{aligned}$$

The average number of columns inspected in a row, after removing the conditioning on the *minmax*, is $\int_{\mu=-\infty}^{\infty}(q_0(\mu)+\cdots+q_n(\mu))nmF^{m-1}(\mu)(1-F^m(\mu))^{n-1}f(\mu)d\mu$. It is possible to obtain a closed form for the sum under the integral, but there is no advantage in doing so.

The expressions simplify when we make the change of variable under the integral of $x \leftarrow F(\mu)$, and we can then write

$$\begin{aligned} E[q_j] &\equiv \int_{\mu=-\infty}^{\infty} q_j(\mu)nmF^{m-1}(\mu)(1-F^m(\mu))^{n-1}f(\mu)d\mu && (16)\\ &= \int_{x=0}^{1}\left(\frac{1-x}{1-x^m}\right)^j\left(1-\frac{j}{n}+\frac{j}{nm}\frac{1-x^m}{1-x}\right)nmx^{m-1}(1-x^m)^{n-1}dx. \end{aligned}$$

We determine an asymptotic expansion of this mean. In the process we shall discover that we need only few of these, since $E[q_{j+1}] = o(E[q_j])$.

The basic expression we consider[2] is $I_{k,n} \equiv \int_0^1 m(1-x)^k x^{m-1}(1-x^m)^{n-1}dx$. Distributing the first binomial and using the substitution $t \leftarrow x^m$ produces an immediate integral:

$$I_{k,n} = \sum_r \binom{k}{r}(-1)^r\Gamma(1+\frac{r}{m})\frac{\Gamma(n)}{\Gamma(n+1+\frac{r}{m})}. \qquad (17)$$

The dependence of this value on the matrix size parameters is not obvious, and we intend to clarify the issue via an asymptotic estimate. We assume that both n and $m \to \infty$. In addition, we assume that their rates of growth are comparable; specifically, whenever we truncate expansions below we assume that $m = o(n\ln n)$ and $n = o(m\ln m)$ (but not necessarily as close as $m = \Theta(n)$). The values of k above, and hence r, may be assumed small (the largest value we actually use below is 2).

We replace $\Gamma(1+r/m)$ by its Taylor development at 1, noting that it is at the same time a power series in r and and an asymptotic series in m^{-1}: $\Gamma(1+r/m) = \sum_{j\ge 0}\gamma_j(r/m)^j$, where $\gamma_j = \Gamma^{(j)}(1)/j!$.

[1]This expression may also be written as $(1-\frac{1}{n})^{*k} = (1+\frac{\partial}{\partial v})(1-\frac{v}{n})^k|_{v=0}$.

[2]We provide here a direct asymptotic evaluation of this integral. The appendix shows how to obtain for it an expansion using depoissonization, according to the results in [2], which are based on the *complex* saddle-point method.

The ratio $\Gamma(n)/\Gamma(n+1+r/m)$ is developed asymptotically as in [4, §4.5]. It is a direct calculation from the standard representation of the gamma function at x by the integral $\int_{t\geq 0} e^{-t}t^{x-1}dt$, and to third order in $1/n$ and $1/m$ produces

$$\frac{\Gamma(n)}{\Gamma(n+1+r/m)} = n^{-1-\frac{r}{m}}\left(1-\frac{r}{2nm}-\frac{r^2}{2m^2n}+\frac{r}{12n^2m}+O(m^{-2}n^{-2})\right) \tag{18}$$

The salient feature of this is again that in the parenthesized factor all powers of r must appear with like powers of m^{-1}. In addition, except the first, they *all* come with additional powers of n^{-1} sprinkled along. For $n^{-r/m}$ we write the natural Taylor development $\sum_{i\geq 0}(-r\ln n/m)^i/i!$.

Note that $\sum_r \binom{k}{r}(-1)^r r^s$ with s a non-negative integer equals $k!(-1)^k \left\langle {s \atop k} \right\rangle$ (where $\left\langle {s \atop k} \right\rangle$ is the subset coefficient, or Stirling number of the second kind). Therefore it vanishes, unless $s \geq k$. Hence only terms of order m^{-k} and higher (i.e., smaller) survive the summation in (17). In particular, terms with r^k produce the leading term. Inserting all these into equation (17), and only keeping the leading term, we find

$$\begin{aligned} I_{k,n} &= \frac{1}{n}\sum_r \binom{k}{r}(-1)^r \sum_j \gamma_j \left(\frac{r}{m}\right)^j \sum_i \left(\frac{-r}{m}\right)^i \frac{\ln^i n}{i!}\left(1+O\left(\frac{r}{mn}\right)\right) \\ &= \frac{1}{n}\sum_{t=0}^{k}\frac{\gamma_t}{m^t}(-1)^{k-t}\left(\frac{\ln n}{m}\right)^{k-t}\frac{1}{(k-t)!}\times k!(-1)^k\left(1+O\left(\frac{\ln n}{n}\right)\right) \\ &= \frac{1}{nm^k}\sum_t \binom{k}{t} b_t \ln^{k-t} n\left(1+O\left(\frac{\ln n}{n}\right)\right), \end{aligned} \tag{19}$$

where we use the notation $b_t \equiv (-1)^t t!\gamma_t = (-1)^t t!\Gamma^{(j)}(1)$. The next order term comes from admitting the next two terms in the parenthesized factor of equation (18), as well as including those from the collection $[r^{k+1}]\{\Gamma(1+r/m)n^{-r/m}\}$, which we do not use below.

In terms of these integrals, $E[q_j] = (n-j)I_{j,n-j} + \frac{j}{m}I_{j-1,n-j+1}$. The second term is here of lower order than the first, and we find

$$E[q_j] = \left(\frac{\ln n}{m}\right)^j \sum_t \binom{j}{t} b_t \ln^{k-t} n\left(1+O\left(\frac{\ln n}{n}\right)\right). \tag{20}$$

The above claim, that $E[q_{j+1}] = o(E[q_j])$, clearly holds. The replacement of $\sum_{j\geq 0}E[q_j]$ by the first three terms is then adequate at the order we maintain. We evaluate them and find

$$E[Q] \approx 1+\frac{\ln n+\gamma}{m}+O\left(\frac{\ln n}{mn}\right) \tag{21}$$

The total expected computation time for the second algorithm would then be, for some suitable coefficient δ, given by $nT_m + T_n + \delta m(1+(\gamma+\ln n)/m)$. Note that the dominant term is in the nT_m part, similar to what we had for the first algorithm.

4.3 Analysis of the third algorithm

The first phase of the third algorithm is nearly identical to that of the second algorithm. The only difference is that during its operation we now record the rows where the successive candidates for *minmax* reside. This adds $n-1$ comparisons and, on the average, $\ln n$ replacements of the current record value of *minmax*.

The second phase consists of the scan of a single selected row; the scan stops when a matrix element strictly smaller than μ is found. This is the only algorithm of the present trio in which we need make no change to provide an accurate analysis.

We use the methodology developed for the analysis of the second algorithm. Let $r_i(\mu)$ be the probability that the row scan overflows column i, with the convention that $r_0(\mu) = 1$. Here again we start with an evaluation of $r_1(\mu)$, for an equivalent of equation (13). The main difference with the second algorithm is that here, when the scan reaches the column whose maximum is the *minmax*, the matrix element equals μ and the scan is never terminated there, while in the second algorithm this was the case with probability $1/m$ only.

For the computation of $r_1(\mu)$, a convenient way to proceed is to separate the case where C_1 is the *minmax*, in probability $1/n$, and the scan continues; otherwise, with a (conditional) probability of $1/m$, C_1 is in the same row and again the scan continues. If it is not, we need to evaluate an integral, identical to the one that led to equation (13). Writing it all in terms of the transformation we used for equation (16), we find

$$r_1(x) = \frac{1}{n} + \left(1 - \frac{1}{n}\right)\left(\frac{1}{m} + \frac{m-1}{m}\int_x^1 \left(1 - \frac{x}{x_1}\right)\frac{m x_1^{m-1} dx_1}{1-x^m}\right). \tag{22}$$

Hence $r_1(x) = \frac{1}{n} + (1-\frac{1}{n})(\frac{1-x}{1-x^m})$. With $j \geq 1$, similar considerations to those used to produce equation (14) lead to the generalization:

$$\begin{aligned} r_j(x) &= \left(1 - \frac{x-x^m}{1-x^m}(1-\frac{1}{n})^*\right)^j \\ &= \left(\frac{1-x}{1-x^m}\right)^{j-1}\left[\frac{j}{n} + \left(1-\frac{j}{n}\right)\frac{1-x}{1-x^m}\right]. \end{aligned} \tag{23}$$

The expected number of comparisons in phase 2 of the algorithm is, as before, $\int_0^1 (r_0(x) + \cdots + r_n(x)) nmx^{m-1}(1-x^m)^{n-1} dx$.

Expressing $E[r_j]$ in terms of the integrals of type $I_{k,n}$ we see that

$$E[r_j] = jI_{j-1,n-j+1} + (n-j)I_{j,n-j},$$

the same integrals we needed for $E[q_j]$, albeit with different multipliers. A similar calculation produces

$$E[r_j] = \frac{\ln^j n}{m^j}\left[\sum_t \binom{j}{t}\frac{b_t}{\ln^t n}\left(1+\frac{m}{n}\frac{t-1}{\ln n}\right)\right]\left(1+O\left(\frac{\ln n}{m}\right)\right).$$

Since these contributions to the expected number of comparisons decrease geometrically in j we only consider the first ones and find

$$\sum_{j\geq 0} E[r_j] = \left(1+\frac{\ln n+\gamma}{m}-\frac{1}{n}\right)\left(1+O\left(\frac{1}{n}+\frac{1}{m}\right)\right). \tag{24}$$

We use no further terms, since this value is two orders (in m or n) below the cost of the first phase – and unlike algorithm II, it does not get multiplied by m.

5. Conclusion

We have carried out probabilistic analyses of three somewhat similar algorithms, and should not be surprised that their costs are similar as well. Let us write for comparison the costs of all three, inserting Greek letters wherever needed to represent instruction and time multipliers, and using a letter and its primed version to denote similar values. In C_I we exchanged m and n (rotated the matrix) for comparison purposes.

$$\begin{aligned}
C_I &= nT_m + \delta(n+1)\tfrac{m}{m-1} &&\cong n(\alpha+\beta m+\nu\ln m+\delta)+\delta,\\
C_{II} &= nT_m + T_n + \delta'(m+\ln n+\gamma) &&\cong n(\alpha+\beta m+\nu'\ln m+\delta'(m/n+\ln n/n+\gamma/n)),\\
C_{III} &= nT_m + T_n + \kappa\ln n + \delta'(1+\tfrac{\ln n}{m}) &&\cong n(\alpha+\beta m+\nu'\ln m)+\alpha'+\beta n+(\nu+\kappa)\ln n+\delta'.
\end{aligned}$$

Several comments are in order:

(1) In each algorithm cost, the first phase (in II and III, or the determination of row minima in I) dominates. Moreover, the main term is the same in all – βmn – and since β and δ are similar in magnitude, the costs of the first and third algorithm are as close as it seems to matter, with a shade favoring algorithm III, whereas algorithm II is slightly more expensive. When we consider that careful programming (as demonstrated in the implementations of the first two algorithms presented in [3]) can result in nontrivial savings, by combining various components of each algorithm, we realize that much of these differences may be illusory.

(2) The conclusions in comment (1) depend somewhat on our assumption that m and n are of similar order (see the discussion following equation (17)). If this were not the case, we would need to modify them to some extent, but not drastically, since the leading term in all the cost functions—βmn—is symmetrical in the matrix dimensions.

(3) While we spent much effort on the probabilistic analysis of the second-phase costs, most of the

cost there is deterministic. Indeed, in all three algorithms, the difference between shortest and longest possible computations ("best case" and "worst case"), is quite modest.

(4) The algorithms we describe *assume*—unlike the versions presented in [3]—that all terms are distinct. At the level of our analysis the differences do not play a role.

(5) The performance of saddle point search algorithms when the terms may not be all distinct can be quite different, as some extreme experiments reported in the solution to exercise 1.3.2-10 in [3] show. Clearly it is possible to design specialized algorithms for some scenarios. An obvious case is when the matrix can have two distinct terms only; then we need only search for a row of high value terms, or a column of low values. An algorithm that does just this would require on the average only $2(m+n)$ comparisons. Interestingly, here the worst case may exceed substantially the average one. With more possible values, special algorithms will not bring nearly as much savings, and are probably not cost effective. The next point is relevant here too.

(6) As noted in [1], not much can be done for matrices with repeated values (when known to have more than two levels) to obtain a more efficient algorithm. In particular, there is no escaping the term βmn. However, for matrices with distinct terms, it is possible to design an algorithm with running time bound by $O(n^{\lg 3})$, where $\lg 3 \approx 1.585\ldots$ (this expression assumes the matrix is square, and hides materially higher numerical coefficients than we have seen above).

(7) The assumption of comparable growth rates for m and n seems natural in this context, but does not affect the techniques needed for developing the asymptotic results — only the terms that need to be collected, to get all that belong to a desired order.

Acknowledgement

We would like to thank Donald E. Knuth who pointed out this unsolved exercise in his *Art of Computer Programming,* and gave valuable comments and improvements to the solution we proposed.

Appendix — Depoissonization Asymptotics

This is a different approach to computing an asymptotic expansion of the integral $I_{k,n}$. It is convenient to change here slightly the notation, and look instead at

$$m^k I_{k,n+1} \equiv g_{m,n} = m^k \int_0^1 (1-x)^k m x^{m-1} (1-x^m)^n dx,$$

for a fixed k, and n and m both tending to infinity. We employ the Poisson generating function $G_m(z) = \sum_n \frac{z^n g_{m,n}}{n!} e^{-z}$ and use the depoissonization lemmas presented by Jacquet and Szpankowski in

[2].

Corollary 1 there states that if a sequence of Poisson generating functions $G_m(z)$ are all entire, and satisfy the following two properties, uniformly in m:

1. For all z in a linear cone including the positive real axis[3], the functions $G_m(z)$ are uniformly in $O(z^{\beta}L(z))$, with $L(z)$ a slowly varying function[4].

2. For all z outside the linear cone, $|G_m(z)e^z| = P(z)e^{|z|}$, where $P(z)$ decreases faster than the reciprocal of any polynomial.

Then $g_{m,n} = G_m(n) + O(n^{\beta-1}L(n))$.

Note: The expansion can be further continued, to any order:

$$g_{m,n} = \sum_{i=0}^{i=k} \sum_{j=0}^{i+k-1} b_{i,j} n^i G_m^{(i)}(n) + O(n^{\beta-k-1}L(n)) \ ,$$

where $G_m^{(i)}(n)$ is the ith derivative of $G_m(z)$ at $z = n$. The coefficients $b_{i,j}$ are obtained from the Taylor power series development: $\sum_{i,j} b_{i,j} x^i y^j = \exp(x\ln(1+y) - xy)$. They can computed recursively, starting with $b_{0,0} = 1$ through: $b_{i,j+1} = -(jb_{i,j} + b_{i-1,j-1})/(j+1)$. The recursion implies that $b_{i,j} = 0$, for all $j < 2i$.

We do not use this extension below. $\square$

We show first that the Poisson generation functions of the double sequence $g_{m,n}$ satisfy the above two conditions.

The summation to obtain the functions is immediate: $G_m(z) = m^k \int_0^1 (1-x)^k m x^{m-1} e^{-x^m z} dx$, and they are clearly entire. For this function, $|G_m(z)| \leq G_m(\Re(z))$. Hence, when we prove condition 1 for a z which is real and positive, we have shown that it holds for an entire linear cone S_θ (in the right half complex plane – i.e., $0 < \theta < \pi/2$).

Since $|z| \geq |\Re(z)|$, condition 2 is satisfied trivially for this function.

Let z be real and positive. We prove that $G_m(z) \leq \frac{(\ln z)^k}{z}$ by transforming it. The change of variable $x^m = e^{-y}$ leads to: $G_m(z) = m^k \int_0^\infty (1 - e^{-y/m})^k e^{-y} \exp[-ze^{-y}] dy$.

Since $\frac{1-e^{-x}}{x} \leq 1$ for all positive x, we find

$$G_m(z) \leq \int_0^\infty y^k e^{-y} \exp[-ze^{-y}] dy. \tag{25}$$

We only add $O(e^{-z})$ to the integral when we extend the limit of integration to $-\infty$ (and this is shown below to be negligible). Hence, $G_m(z) \leq \int_{-\infty}^\infty y^k e^{-y} \exp[-ze^{-y}] dy + O(e^{-z})$.

[3]Such a cone is a region in the complex plane defined by $S_\theta = \{z : \ |\arg z| \leq \theta, \quad 0 < \theta < \pi/2\}$.

[4]$L(z)$ is a slowly varying function (at infinity) in the cone S_θ if $\lim_{u\to\infty} L(aue^{i\rho})/L(u) = 1$, for all $a > 0$ and $|\rho| < \theta$.

By obvious change of variable we get $G_m(z) \leq \frac{1}{z}\int_{-\infty}^{\infty}(x+\ln z)^k e^{-x}\exp[-e^{-x}]dx$.

The integral $\int_{-\infty}^{\infty}(-x)^j e^{-x}\exp[-e^{-x}]dx$ equals the jth derivative of the gamma function at 1 (using the same representation that leads to (18)), which is bounded by $j!$, hence as z increases the leading term in the above bound is $(\ln z)^k/z$. Therefore condition 1 is proved with $\beta = -1$ and $L(z) = (\ln z)^k$. □

To obtain our expansion we evaluate $G_m(z)$ as above, up to the expression leading to equation (27), but avoid simplifying. Instead we consider

$$G_m(z) = \frac{1}{z}\int_{-\infty}^{\infty} m^k(1-e^{-(y+\ln z)/m})^k e^{-y}\exp[-e^{-y}]dy + O(e^{-z}) \ ,$$

and expand the exponential in $m^k(1-e^{-(y+\ln z)/m})^k$ by powers of y:

$$G_m(z) = \frac{1}{z}\int_{-\infty}^{\infty}(y+\ln z)^k\left[\sum_{i\geq 0}\frac{(-1)^{i+1}}{(i+1)!}\left(\frac{y+\ln z}{m}\right)^i\right]^k e^{-y}\exp[-e^{-y}]dy + O(e^{-z}) \ .$$

Keeping the first term ($i=0$) only leads to

$$\begin{aligned} G_m(z) &= \frac{1}{z}\int_{-\infty}^{\infty}(y+\ln z)^k e^{-y}\exp[-e^{-y}]dy + O\left(\frac{\ln z}{mz}\right) \\ &= \frac{1}{z}\sum_{j\geq 0}\binom{k}{j}\ln^{k-j}(z)(-1)^j\Gamma^{(j)}(1) + O\left(\frac{\ln z}{mz}\right) \ . \end{aligned}$$

Note that we have "lost," as promised, the contribution of $O(e^{-z})$. The depoissonization lemma then gives

$$g_{m,n} = \frac{1}{n}\sum_{j\geq 0}\binom{k}{j} b_j \ln^{k-j} n + O\left(\frac{\ln n}{mn}\right) \ , \tag{26}$$

where $b_j = (-1)^j\Gamma^{(j)}(1)$. This is the exact analog of the result obtained by direct calculation. We expect that for higher-order terms this approach is likely to prove more manageable.

References

[1] D. Bienstock, F. Chung, M. Fredman, A. Schäffer, P. Shor, and S. Suri, "A note on finding a strict saddle-point," *American Math. Monthly* **98**, 418–419 (1991).

[2] P. Jacquet, W. Szpankowski, "Analytical depoissonization and its applications," submitted for *Theor. Comp. Science*, as a Fundamental Study, 1997.

[3] D. Knuth, *The Art of Computer Programming*, Addison-Wesley, volume 1: Fundamental Algorithms, 1968. 3rd. Ed. 1997.

[4] F. W. J. Olver, *Asymptotics and Special Functions,* Academic Press, New York, 1974.