Available online at www.sciencedirect.com
ScienceDirect
Computational Statistics & Data Analysis 51 (2006) 490–498
COMPUTATIONAL STATISTICS & DATA ANALYSIS
www.elsevier.com/locate/csda

A major improvement to the Network Algorithm for Fisher's Exact Test in 2×c contingency tables

F. Requena[a,*], N. Martín Ciudad[b]

[a] *Department of Statistics and O.R., University of Granada, 18071 Granada, Spain*
[b] *Department of Mathematics, University of Extremadura, 10004 Cáceres, Spain*

Received 3 March 2005; received in revised form 6 September 2005; accepted 6 September 2005
Available online 5 October 2005

Abstract

Based on the Network Algorithm proposed by Mehta and Patel for Fisher's Exact Test on $2 \times c$ contingency tables, the relations between maximum subpath lengths are studied. A recurrence relation between maximum subpath lengths is obtained and an ordering of the maximum path lengths is established. Based on these results, some modifications in the Network Algorithm for $2 \times c$ tables are proposed. These modifications produce a drastic reduction in computation time which in some cases is higher than 99.5% compared to StatXact-5. Moreover, and with purely practical objectives, a grouping in intervals of subpath lengths of the Network Algorithm is proposed which enable us to obtain the p-value with a limited number of exact figures which is more than sufficient in practice, while with a drastic reduction in the amount of memory required and additional reductions in computational time. The proposed modifications are valid for any $2 \times c$ contingency table, and are compatible with other improvements already proposed for the Network Algorithm, and especially with the Hybrid Algorithm of Mehta and Patel.

Keywords: Network Algorithm; Fisher's Exact Test; $2 \times c$ contingency table; Multivariate hypergeometric distribution

1. Introduction

Let X^{o} be an observed $2 \times c$ contingency table, with fixed column sums $(C_1, \ldots, C_c)$ and fixed row sums (R_1, R_2), and $N = \sum R_i = \sum C_j$. Let us denote by F the set of all the possible $2 \times c$ tables with the same marginal totals as X^{o}, by X a $2 \times c$ table of F and by $(x_1, \ldots, x_c)$ the first row of X. It is known that, if we assume the hypothesis of row and column independence, the random vector $(x_1, \ldots, x_c)$ has a Multivariate Hypergeometric distribution $\mathrm{MH}(R_1; C_1, \ldots, C_c)$ and

$$\mathrm{P}(X) = \frac{1}{D} \prod_{j=1}^{c} \binom{C_j}{x_j}, \quad X \in F, \quad D = \binom{N}{R_1}.$$

The best algorithm to carry out Fisher's Exact Test is the Network Algorithm (NA) proposed by Mehta and Patel (1980, 1983, 1986a) (see Hirji and Johnson, 1996). The NA for a $2 \times c$ table can be summed up as follows. A directed

* Corresponding author. Tel.: +34 958 243536; fax: +34 958 246117.
E-mail address: fcoreque@ugr.es (F. Requena).

0167-9473/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.csda.2005.09.004

acyclic network of nodes and arcs is built. The nodes are structured in $c+1$ stages, labelled as $c, c-1, \ldots, 1, 0$. In any stage k there is a set of nodes and each of them is labelled by a pair of integer values $\left(k, R_{(k)}\right)$ with

$$\max\left\{0, R_1 - N + N_{(k)}\right\} \leqslant R_{(k)} \leqslant \min\left\{N_{(k)}, R_1\right\},$$

where $N_{(k)} = C_1 + \cdots + C_k$. In particular, in stage c there is a single node $\left(c, R_{(c)}\right)$ with $R_{(c)} = R_1$ (initial node) and in stage 0 there is also a single node $\left(0, R_{(0)}\right)$ with $R_{(0)} = 0$ (terminal node). The $R_{(k)}$ values of the nodes of any other stage k form a series of positive consecutive integer numbers. The arcs emanate from each node of stage k and are directed towards a node of stage $k-1$. Each node of stage $k-1$ which is joined by an arc with a node $\left(k, R_{(k)}\right)$ of stage k is a daughter node (DN) of $\left(k, R_{(k)}\right)$, which is known as the mother node (MN). Each node of stage $k-1$ is always a DN of at least one MN of stage k. Each arc which reaches a node is connected to each one of the arcs which emanate from that same node. A path across the network is defined as a series of connected arcs which emanate from the initial node and reach the terminal node passing through $c-1$ intermediate nodes. Each intermediate node of a path divides the path into two subpaths.

The network is constructed in such a way that each subpath from the node $\left(k, R_{(k)}\right)$ to the terminal node corresponds to just one $2 \times k$ subtable with row sums $\left(R_{(k)}, N_{(k)} - R_{(k)}\right)$ and column sums $(C_1, \ldots, C_k)$. More specifically, subpath $\left(k, R_{(k)}\right) \to \left(k-1, R_{(k-1)}\right) \to \cdots \to (0,0)$ corresponds to the $2 \times k$ subtable whose first row is $x_j = R_{(j)} - R_{(j-1)}$, $j = 1, \ldots, k$. From now on, we will represent each subpath by its corresponding vector $(x_1, \ldots, x_k)$. For $k = c$ we will have a one-to-one correspondence between all of the paths from the initial node to the terminal node and the set F.

The length of the arc which joins an MN $\left(k, R_{(k)}\right)$ with its DN $\left(k-1, R_{(k-1)}\right)$ is defined by

$$\ell\left(k, R_{(k)}, R_{(k-1)}\right) = \binom{C_k}{R_{(k)} - R_{(k-1)}}, \tag{1}$$

and the length of a path (or subpath) is defined as the product of the length of their arcs. Therefore, the length of the path which corresponds to $X \in F$ will be $D\mathrm{P}(X)$.

The p-value associated to the observed X^{o} table is given by

$$p\text{-value}\left(X^{\mathrm{o}}\right) = \sum_{X \in F^{\mathrm{o}}} \mathrm{P}(X),$$

where $F^{\mathrm{o}} = \{X \mid X \in F, \mathrm{P}(X) \leqslant \mathrm{P}\left(X^{\mathrm{o}}\right)\}$. The NA calculates this p-value by identifying and adding up the lengths of all of the paths in the network which are not longer than $D\mathrm{P}(X^{\mathrm{o}})$, but with no need to explicitly enumerate each path. The decision as to whether or not the paths of the network contribute to the p-value takes place based on sets of paths and this decision is made in the nodes of the network. For each DN $\left(k-1, R_{(k-1)}\right)$ of the MN $\left(k, R_{(k)}\right)$ we must check if one of the following conditions holds:

$$\mathrm{PAST}\ \ell\left(k, R_{(k)}, R_{(k-1)}\right) \mathrm{LP}\left(k-1, R_{(k-1)}\right) \leqslant D\mathrm{P}\left(X^{\mathrm{o}}\right) \tag{2}$$

and

$$\mathrm{PAST}\ \ell\left(k, R_{(k)}, R_{(k-1)}\right) \mathrm{SP}\left(k-1, R_{(k-1)}\right) > D\mathrm{P}\left(X^{\mathrm{o}}\right), \tag{3}$$

where $\mathrm{LP}\left(k-1, R_{(k-1)}\right)$ and $\mathrm{SP}\left(k-1, R_{(k-1)}\right)$ are the lengths of the longest and shortest subpath, respectively, from the DN $\left(k-1, R_{(k-1)}\right)$ to the terminal node, and PAST is the length of a subpath from the initial node to the MN $\left(k, R_{(k)}\right)$ (see Mehta and Patel, 1983). If Q is the set of all of the paths which pass through the DN $\left(k-1, R_{(k-1)}\right)$ and which have a common subpath (of length PAST) to the MN $\left(k, R_{(k)}\right)$, then no path of Q will contribute to the p-value if (3) holds and all of the paths of Q will contribute to the p-value if (2) holds, in which case the overall contribution is

$$\binom{N_{(k-1)}}{R_{(k-1)}} \ell\left(k, R_{(k)}, R_{(k-1)}\right) \mathrm{PAST}.$$

In both cases, the paths of Q are not considered again in the NA.

Since Mehta and Patel (1980, 1983, 1986a,b), various improvements have been proposed for this NA in the general case of $r \times c$ tables (Joe, 1988; Clarkson et al., 1993; Aoki, 2002). The statistical package StatXact-5 (2001) incorporates this NA for the exact analysis of the unordered $r \times c$ tables. Furthermore, in the case of $2 \times c$ tables, Shao (1997) has proposed a modification of the NA which is more efficient when the column sums are equal, but is generally less efficient when the column sums are different.

In this paper, we present some modifications to the NA which are valid for any $2 \times c$ table and which always make the NA much more efficient (see Section 4). On the one hand, we propose a general recursive method to calculate all of the exact LP(., .) quantities which are necessary in the processing of NA, based on the recurrence relation obtained in Section 2. On the other hand, in each stage of the NA, conditions (2) and (3) are checked for all of the DNs and all of the PAST values corresponding to each MN, but in the overwhelming majority of cases it is not necessary to check these conditions, especially in the case of condition (2). In Section 2, we obtain a relation between the maximum lengths of groups of subpaths which emanate from an MN, which, along with the consideration of the ordered PAST values, makes the number of times that condition (2) must be checked practically insignificant. At the same time, this makes it possible for the sets of paths which contribute to the p-value to be handled in big blocks. These modifications (which are described in Section 3) will produce a drastic reduction in the total computation time of the NA applied to a $2 \times c$ table.

2. Relation between maximum subpath lengths in the NA

According to the construction of the NA, for each node $(k, R_{(k)})$ $k > 1$ there is a one-to-one correspondence between the set of subpaths from the node $(k, R_{(k)})$ to the terminal node and the set of points of Multivariate Hypergeometric distribution $\mathrm{MH}(R_{(k)}; C_1, \ldots, C_k)$, and there is also a one-to-one correspondence between the modes of this distribution and the longest subpaths from $(k, R_{(k)})$ to the terminal node. Specifically, if $(x_1^*, \ldots, x_k^*)$ is a mode of the previous distribution, then the longest subpath which corresponds to it is determined by nodes $(i, R_{(i)})$ $i = k, k-1, \ldots, 0$ such that $x_j^* = R_{(j)} - R_{(j-1)}$ $j = 1, \ldots, k$. Therefore, we can say that the distribution $\mathrm{MH}(R_{(k)}; C_1, \ldots, C_k)$ is associated with node $(k, R_{(k)})$. Logically, for any node $(k, R_{(k)})$ $k > 1$, $R_{(k)} \leqslant N_{(k)}$ must hold.

From the generation process of the DNs, given an MN $(k, R_{(k)})$, the values $R_{(k-1)}$ of its s DNs in stage $k-1$ constitutes a succession of integer values, which we will denote by $R_{(k-1)i}$ $i = 1, \ldots, s$, such that $R_{(k-1)1} = \max\{0, R_{(k)} - C_k\}$ (for the first DN), $R_{(k-1)s} = \min\{R_{(k)}, N_{(k-1)}\}$ (for the last DN) and $R_{(k-1)i+1} = R_{(k-1)i} + 1$ $i = 1, \ldots, s-1$. Given what we have previously established, each node of stage $k-1$ $(k > 2)$ will have an associated Multivariate Hypergeometric distribution, and particularly, for $i = 1, \ldots, s$, $\mathrm{MH}(R_{(k-1)i}; C_1, \ldots, C_{k-1})$ will be the distribution associated with DN $(k-1, R_{(k-1)i})$. These s distributions constitute a family which we will denote by $\mathscr{J}_k(R_{(k)})$ and which we will associate with the MN $(k, R_{(k)})$. Moreover, according to the construction of the NA, if the distribution of $(x_1, \ldots, x_k)$ is $\mathrm{MH}(R_{(k)}; C_1, \ldots, C_k)$, then the family $\mathscr{J}_k(R_{(k)})$ will coincide with the family of conditional distributions of $(x_1, \ldots, x_{k-1} \mid x_k = R_{(k)} - R_{(k-1)i})$ $i = 1, \ldots, s$.

During stage k of the NA we need to obtain the exact lengths $\mathrm{LP}(k-1, R_{(k-1)})$ corresponding to the nodes of stage $k-1$. The following result establishes a recurrence relation between these lengths, which gives us a recursive method to obtain the LPs of all of the nodes of stage $k-1$ starting from the LP corresponding to one of the nodes of that stage.

Theorem 1. *If* $(k-1, R_{(k-1)}-1)$, $(k-1, R_{(k-1)})$ *and* $(k-1, R_{(k-1)}+1)$ *are consecutive arbitrary nodes of stage* $k-1$ $(k>2)$ *and* $(x_1^*, \ldots, x_{k-1}^*)$ *is a mode of distribution* $\mathrm{MH}(R_{(k-1)}; C_1, \ldots, C_{k-1})$ *associated with the node* $(k-1, R_{(k-1)})$, *then* $(x_1^*, \ldots, x_v^* - 1, \ldots, x_{k-1}^*)$ *and* $(x_1^*, \ldots, x_u^* + 1, \ldots, x_{k-1}^*)$ *are modes of the MHs associated with nodes* $(k-1, R_{(k-1)}-1)$ and $(k-1, R_{(k-1)}+1)$, *respectively*, *and the recurrence relations*

$$\mathrm{LP}(k-1, R_{(k-1)}+1) = \mathrm{LP}(k-1, R_{(k-1)}) \frac{C_u - x_u^*}{x_u^* + 1} \tag{4}$$

and

$$\mathrm{LP}(k-1, R_{(k-1)}-1) = \mathrm{LP}(k-1, R_{(k-1)}) \frac{x_v^*}{C_v - x_v^* + 1} \tag{5}$$

hold, where v *is a value of* j $(1 \leqslant j \leqslant k-1)$ *which maximizes the expression* $x_j^* / \left(C_j + 1\right)$, *and* u *is a value of* j $(1 \leqslant j \leqslant k-1)$ *which minimizes the expression* $(x_j^* + 1)/(C_j + 1)$.

The proof is given in the Appendix.

Let us now denote by $\mathrm{LP}'(k, R_{(k)}, R_{(k-1)})$ the product $\ell\left(k, R_{(k)}, R_{(k-1)}\right) \mathrm{LP}\left(k-1, R_{(k-1)}\right)$ which appears on the left of expression (2), which represents the longest subpath length that starts from an MN $\left(k, R_{(k)}\right)$ and passes through its DN $\left(k-1, R_{(k-1)}\right)$. Given an MN $\left(k, R_{(k)}\right)$, we will have an $\mathrm{LP}'\left(k, R_{(k)}, R_{(k-1)i}\right)$ for each one of its DNs $\left(k-1, R_{(k-1)i}\right)$, $i = 1, \ldots, s$.

Definition 1. Given an MN $\left(k, R_{(k)}\right)$, we can say that a DN $\left(k-1, R_{(k-1)p}\right)$, $1 \leqslant p \leqslant s$, is a principal daughter node (PDN) of $\left(k, R_{(k)}\right)$ when $R_{(k-1)p} = R_{(k)} - x_k^*$ for some mode $\left(x_1^*, \ldots, x_k^*\right)$ of the $\mathrm{MH}\left(R_{(k)}; C_1, \ldots, C_k\right)$ associated with $\left(k, R_{(k)}\right)$.

As the mode is not necessarily unique, the PDN is not necessarily unique either but, according to Lemma 2.2 and Theorem 2.1 described in Requena and Martín (2000), the maximum number of PDNs of an MN is two, in which case both should be consecutive. This allows us to divide the succession of DNs of an MN in two subsuccessions separated by the PDN (or by the two PDNs). More specifically, let $S_L = \left\{\left(k-1, R_{(k-1)1}\right), \ldots, \left(k-1, R_{(k-1)p-1}\right)\right\}$ and $S_U = \left\{\left(k-1, R_{(k-1)q}\right), \ldots, \left(k-1, R_{(k-1)s}\right)\right\}$ be the two subsuccessions, where $q = p+1$ if $\left(k-1, R_{(k-1)p}\right)$ is the only PDN and $q = p+2$ if there are two PDNs, $\left(k-1, R_{(k-1)p}\right)$ and $\left(k-1, R_{(k-1)p+1}\right)$, and s is the total number of DNs. S_L, S_U, or both might not exist, i.e. not contain any DN.

The next result establishes an ordering of the $\mathrm{LP}'\left(k, R_{(k)}, R_{(k-1)i}\right)$ quantities which correspond to the DNs of an MN, which makes the number of times that condition (2) must be checked practically insignificant (as we shall see in the next section).

Theorem 2. *Given an* MN $\left(k, R_{(k)}\right)$, *its* DNs $\left(k-1, R_{(k-1)i}\right)$, $i = 1, \ldots, s$, *and* p *and* q *which determine the subsuccessions* S_L *and* S_U, *then*

$$\mathrm{LP}'\left(k, R_{(k)}, R_{(k-1)i+1}\right) < \mathrm{LP}'\left(k, R_{(k)}, R_{(k-1)i}\right) \tag{6}$$

holds, for $\left(k-1, R_{(k-1)i+1}\right)$ *belonging to* S_U, $i = q-1, q, \ldots, s-1$, *and*

$$\mathrm{LP}'\left(k, R_{(k)}, R_{(k-1)i-1}\right) < \mathrm{LP}'\left(k, R_{(k)}, R_{(k-1)i}\right) \tag{7}$$

holds, for $\left(k-1, R_{(k-1)i-1}\right)$ *belonging to* S_L, $i = p, p-1, \ldots, 2$. *Moreover, if there are two* PDNs, $\left(k-1, R_{(k-1)p}\right)$ *and* $\left(k-1, R_{(k-1)p+1}\right)$, *then*

$$\mathrm{LP}\left(k, R_{(k)}\right) = \mathrm{LP}'\left(k, R_{(k)}, R_{(k-1)p}\right) = \mathrm{LP}'\left(k, R_{(k)}, R_{(k-1)p+1}\right), \tag{8}$$

satisfying only the first equality of (8) *if* $\left(k-1, R_{(k-1)p}\right)$ *is the only* PDN.

The proof is given in the Appendix.

3. Modifications to the NA for 2 × c tables

We can sum up the modifications to the NA in the following points:

1. As Clarkson et al. (1993) have shown, in each stage k the exact $\mathrm{LP}\left(k-1, R_{(k-1)}\right)$ quantities must be calculated and stored, in order to be recovered when necessary. Using the recurrence relations of Theorem 1, we propose a recursive method to calculate those LPs for all of the nodes of stage $k-1$ starting from the LP which corresponds to one of the nodes of that stage (for example, the first node). This initial LP is calculated through a method developed

by Requena and Martín (2003). This procedure also simultaneously gives us a mode of the MHs associated with each one of nodes of stage $k-1$ (which will be used to determine the PDNs).

2. In order to analyse the MN $\left(k, R_{(k)}\right)$, we recover its PASTs ($\mathrm{PAST}_1, \ldots, \mathrm{PAST}_n$) which we consider in ascending order, we determine its DNs (which we represent in a simplified way with $1, 2, \ldots, s$) and its PDN, and we define subsuccessions S_L and S_U. In order to simplify the algorithm, we can ignore the fact that there may be two PDNs; what is necessary is that a PDN separates both subsuccessions (if there is another PDN and it is within S_L or S_U, it is not important in practice). From Theorem 2, if condition (2) is fulfilled for a PAST and a DN, it will also be fulfilled for all of the most extreme DNs of the subsuccession (the subsequent DNs if it is S_U, or the previous DNs if it is S_L). Moreover, if (2) is fulfilled for a PAST_j, this implies that it will also be fulfilled for all of the $\mathrm{PAST}_{j'}$ with $j' < j$. Let us denote by r_i the subindex of maximum PAST such that (2) is fulfilled for DN i, $i = 1, \ldots, s$, considering that $r_i = 0$ if (2) is not fulfilled for PAST_1. It can be shown that $r_i = 0$ always holds for the PDN. Furthermore, we always find $0 \leqslant r_i \leqslant r_{i+1} \leqslant n$ within S_U and $0 \leqslant r_i \leqslant r_{i-1} \leqslant n$ within S_L. According to these previous considerations, we can construct an algorithm to determine all of the r_i, checking condition (2) a small number of times. We will call a' and b' the minimum i in S_L and the maximum i in S_U, respectively, such that $r_i < n$ (taking $a' = 1$ if S_L does not exist, and $b' = s$ if S_U does not exist). In the same way, and as long as there is some $r_i > 0$, a and b will represent the maximum i in S_L and the minimum i in S_U, respectively, such that $r_i > 0$.
3. Having finished the analysis of condition (2) for the present MN, we can begin the analysis of condition (3). We determine for each DN i, $a' \leqslant i \leqslant b'$ (this will necessarily include the PDN) the subindex (t_i) of the minimum PAST such that (3) holds, considering that $t_i = n + 1$ if (3) is not fulfilled for PAST_n. As $r_i < t_i \leqslant n + 1$ for every i and it usually holds that $t_{i+1} \leqslant t_i$ within S_L and $t_{i-1} \leqslant t_i$ within S_U, and as (3) holds for a PAST_j this implies that it will also hold for all of the $\mathrm{PAST}_{j'}$ with $j' > j$, we can construct an algorithm to determine all of the t_i checking (3) a relatively small number of times.
4. The next step in the analysis of the present MN $\left(k, R_{(k)}\right)$ is to store the updated PASTs. For DN i, $a' \leqslant i \leqslant b'$, the updated PASTs will be $\mathrm{PAST}_j \ell(k, R_{(k)}, R_{(k-1)i})$ $j = r_i + 1, \ldots, t_i - 1$, which will obviously be ordered just as the PAST_j's are ordered.
5. The last step is to calculate the contribution of the present MN to the p-value. We obtain

$$T(r_i) = \sum_{j=1}^{r_i} \mathrm{PAST}_j f_j$$

for each $r_i > 0$, where f_j is the frequency of PAST_j. These $T(r_i)$ are calculated by recursion, i.e. if m and m' are two consecutive values of r_i $(m < m')$,

$$T(m') = T(m) + \sum_{j=m+1}^{m'} \mathrm{PAST}_j f_j.$$

The total contribution of DN i (if $r_i > 0$) to the p-value is H_i / D, where

$$H_i = \binom{N_{(k-1)}}{R_{(k-1)i}} \ell(k, R_{(k)}, R_{(k-1)i}) T(r_i),$$

and the total contribution of the present MN to the p-value will be

$$\frac{1}{D}\left(\sum_{i=1}^{a} H_i + \sum_{i=b}^{s} H_i\right).$$

6. With purely practical aims, it is enough to know the p-value of Fisher's Exact Test with three or even two exact figures. In this sense, a simple modification of the NA allows us to consider the PASTs grouped in intervals. In a simple way, we consider that each previously stored PAST belongs to a different interval. When storing a new updated PAST A, if there is any previously stored PAST which is within $A \pm A10^{-g}$, then both PASTs will belong

to the same interval and their frequencies will be added, or, otherwise, A will generate a new interval. Therefore, the length of the intervals depends on the value of g and the updated PAST values which are stored. Taking a suitable value of g, we can obtain the number of exact figures wanted for the p-value. In this way, we drastically reduce both the total computation time and the amount of memory required.

7. Finally, let us see some aspects which we have verified empirically. When we consider column sums C_j in descending order, there is frequently a saving in computation time and, in some cases, this saving is significant. Moreover, if in condition (3) we use the exact values of $\mathrm{SP}\big(k-1, R_{(k-1)}\big)$, for a small or moderate c, the computation time of these values is generally small in relation to the total computation time of the NA, however, for a large c and C_j's which are not too small, that computation time can be considerable in the first stages of the NA. In these cases, there is a certain time saving if condition (3) is not applied in the first stages of the NA (for example, in stages c, $c-1$ and $c-2$).

4. Discussion

In this paper, we have presented some modifications to the NA in order to implement Fisher's Exact Test in $2 \times c$ tables, which make it much more efficient than the classic NA of Mehta and Patel (1986a). Henceforth, we shall call the NA with these proposed modifications, except the one referred to previously in Point 6, the modified NA. These modifications are valid for any $2 \times c$ contingency table and are compatible with other improvements which have already been proposed for the NA. It is particularly important that they are compatible with the Hybrid Algorithm of Mehta and Patel (1986b).

With the sole objective of comparing the classic NA with the modified NA and to study the advantages of the latter, we have developed two FORTRAN programmes, one for the classic NA (Mehta and Patel, 1986a) with the LPs and the SPs calculated according to Joe (1988) and another identical one for the modified NA (without considering the modification mentioned in Point 6 above). Both programmes carry out the calculations in FORTRAN double precision. Nonetheless, they can be improved with more efficient programming techniques and other improvements, such as the incorporation of Mehta and Patel's (1986b) hybrid method. This improvement is already incorporated in StatXact-5 (2001). We have solved many $2 \times c$ tables through both FORTRAN programmes and StatXact-5, which were implemented on a Pentium PC. The set of tables resolved includes $2 \times c$ tables with (a) values of c between 8 and 22, (b) values of N up to 300, (c) column sums which are equal, moderately different and very different, and (d) small p-values, large p-values and around 0.05 (the majority). Moreover, in all of these tables it is not advisable to use purely asymptotic methods. We will now comment upon the features and performance of the proposed modifications for the NA.

1. Using the method presented in Joe (1988), the total computation time of the exact values of all of the necessary LPs in the NA is normally small (in relation to the total time of the NA) for a small or moderate c, however, for a large c and relatively large C_j's that time might be considerable. On the contrary, using the recursive method proposed in Point 1 above, this total computation time is always absolutely negligible, even for large values of c and C_j's. This is interesting especially in the tables where the use of Mehta and Patel's Hybrid Algorithm is advisable.
2. In the classic NA, condition (2) is checked a great number of times throughout the algorithm and that number increases as c and the C_j's increase. On the contrary, in the modified NA, (2) is checked an insignificant number of times in relation to the classic NA, although in both cases (classic and modified NA) the exact LP quantities are used. For example, in a particular 2×12 table we have verified that the number of times that (2) is checked is around 10^9 in the classic NA and around 10^4 in the modified NA. This is a consequence of Theorem 2, as is shown in Point 2 of the previous section. Therefore, the total computation time of this aspect of the algorithm is negligible in the modified NA.
3. Storing the set of updated PASTs as was indicated in the previous section implies an important saving in computation time, because, for DN i, both the set of updated PASTs which are going to be stored and the previously stored PASTs are ordered and this represents a great advantage when checking the previously stored PASTs in order to determine the frequency of each PAST. Moreover, always maintaining the PASTs in order does not represent an important additional effort (we do not need to order them in each stage), since in stage $c-1$ each MN has a single PAST, and in the later stages the updated PASTs which are stored are already ordered and are stored in order.

4. In order to make a comparison with StatXact-5, we have calculated the ratio "StatXact-5 computation time/modified NA computation time". This ratio is quite variable, depending on the observed $2 \times c$ table, but it is always much higher than 1. This ratio tends to increase as c and the C_j's increase. We have found ratios between 10 and 70 for moderate values of c (between 8 and 12), and up to approximately 200 for $c > 12$. The usefulness of the modified NA is greater for large values of c, since the real computation times through StatXact-5 can be considerable. For example, for the 2×18 table

8	6	3	8	4	6	5	3	4	3	3	5	4	3	6	3	5	7
5	7	10	4	8	6	5	5	13	14	14	10	14	15	13	15	13	6

(p-value=0.051572) the real computation times were 1 h, 2 min and 4.06 s with StatXact-5 and only 53.07 s with the modified NA (ratio = 70.2). The largest ratio and the more striking difference in real computation time has been found in a particular 2×22 table in which the times were about 7 h with StatXact-5 and less than 2 min with the modified NA (ratio = 218).

The ratio also depends on how the column sums are balanced. In the tables with $C_1 = C_2 = \cdots = C_c$ we have obtained ratios between 10 and 115, whilst the highest ratios were found in tables with unbalanced column sums. With balanced column sums, the classic NA of StatXact-5 uses an easily computable formula to obtain the exact LP values, so the improvement introduced in Point 1 of the previous section does not offer any advantage. The reduction in computation times is due to the other improvements in the modified NA, especially Point 2. With unbalanced column sums, StatXact-5 uses an upper bound for LP, so the improvement introduced in Point 1 also contributes to the reduction in total computation time because it reduces the number of paths to be analysed (see Joe, 1988) and because of what was stated in Point 1 of this section.

Finally, the reduction in computation time depends on the p-value of the table. For tables with a p-value of around 0.05, and more specifically, for tables with a p-value between 0.01 and 0.10 (which is the most interesting range in practice) we have obtained ratios between 10 and 90, and in most cases the ratio was over 40. Similar ratios were found for tables with a large p-value. On the other hand, the highest ratios were found in tables with a small p-value.

5. We also compared the classic NA (with the LPs calculated according to Joe, 1988) with the modified NA, using for this purpose the two FORTRAN programmes mentioned at the beginning of this section. The ratio "classic NA computation time/modified NA computation time" depends on c, C_j's and the p-value in the same way as previously expressed in Point 4, although this ratio is always clearly higher than that obtained in the comparison made with StatXact-5. For moderate values of c, ratios between 20 and 300 have been found (although most of the time they were higher than 50), and for large values of c, ratios up to approximately 1000 have been obtained. For tables with a p-value between 0.01 and 0.10, ratios between 20 and 400 have been found.
6. Most of the total computation time with the modified NA is consumed in the storing of the updated PASTs of the DNs. Therefore, if the number of PASTs which are accumulated in the last stages of the NA is high, the total computation time increases considerably and also creates problems due to the great amount of memory required. In these cases, we can use the additional modification indicated in Point 6 of the previous section (which until now we have not used). This will considerably reduce the total number of PASTs (and, therefore, the total computation time and the amount of memory required) in exchange for a slight loss in precision in the p-value which, in practical terms, is not important. In this sense, we have slightly changed the FORTRAN programme of the modified NA in order to include the new modification. We have solved many $2 \times c$ tables for various values of g and we have calculated the reductions both in computation time and the maximum number of PASTs in a stage, with respect to those obtained through the modified NA. Taking $g = 4$, in all of the cases we have obtained at least 4 exact figures in the p-value (e.g. if $p = 2.463402 \times 10^{-5}$, at least the figures 2, 4, 6 and 3 are exact, which is more than we usually need in practice). The reductions are variable and rather more important when the total computation time with the modified NA is higher (which is when we most need the aforementioned reduction). For tables with computation times lower than 1 min, we have obtained reductions of between 15% and 60% in computation time, and between 23% and 65% in the maximum number of PASTs. For tables with a computation time near to or higher than 1 min, the reductions range from 50% to 92% for computation time and from 57% to 97% for the maximum number of PASTs. With $g = 3$, rather more important reductions are obtained, with a number of exact figures remaining in

the p-value which is sufficient in practice. Therefore, computation time and the amount of memory required will depend on the number of exact figures wanted for the p-value.

Acknowledgements

The authors would like to thank the co-editor and the two referees for helpful comments which improved this paper. This research was supported by the MEC, Spain, Grant number MTM-2004-00989 (with cofinancing of the FEDER) and by the C. I. C. E., Junta de Andalucía, Spain, Grant number FQM-0235.

Appendix A

Proof of Theorem 1. The first part of the theorem is deduced from Theorem 2.1 given in Requena and Martín (2003). On the other hand, if $(k-1,\, R_{(k-1)})$ and $(k-1,\, R_{(k-1)}+1)$ are DNs of the same MN $(k,\, R_{(k)})$, then the demonstration of expression (4) is simple, since the distributions associated with these DNs would belong to the family $\mathscr{J}_k\left(R_{(k)}\right)$ associated with MN $\left(k,\, R_{(k)}\right)$, and from Theorem 3.2 given in Requena and Martín (2000) (where a family like $\mathscr{J}_k\left(R_{(k)}\right)$ is studied) expression (4) would be easily deduced. Therefore, it would be enough to demonstrate that given two nodes $(k-1,\, R_{(k-1)})$ and $(k-1,\, R_{(k-1)}+1)$ there is always a node $\left(k,\, R_{(k)}\right)$ which is an MN of both. This is true since, if it were false, from the process of generation of DNs we would find that $(k-1,\, R_{(k-1)})$ would be the last DN of an MN, e.g. $\left(k,\, R_{(k)}\right)$, and $(k-1,\, R_{(k-1)}+1)$ would be the first DN of the following MN $\left(k,\, R_{(k)}+1\right)$, i.e.

$$\min\left\{R_{(k)},\, N_{(k-1)}\right\} < \max\left\{0,\, R_{(k)} - C_k + 1\right\},$$

but this never holds because $R_{(k)} \geqslant 0$, $N_{(k-1)} > 0$, $C_k \geqslant 1$ and $R_{(k)} + 1 \leqslant N_{(k)}$. Expression (5) is obtained in a similar way. □

Proof of Theorem 2. Let us consider the set of all the subpaths from $\left(k,\, R_{(k)}\right)$ to $(0,\, 0)$, and let us denote by L any of these subpaths which has a maximum length $\mathrm{LP}\left(k,\, R_{(k)}\right)$. Bearing in mind what we said at the beginning of Section 2, and from Definition 1, it is clear that through each PDN of the MN $\left(k,\, R_{(k)}\right)$ at least one subpath L passes and every subpath L must pass through a PDN. This demonstrates the last part of the theorem and expressions (6) and (7) for $i = q-1$ and $i = p$, respectively. Now we can demonstrate expression (6) by induction, since it is true for the first value of i (i.e. $i = q-1$). Therefore, we have to demonstrate that (6) is true for i supposing that it is true for $i-1$ $(q \leqslant i \leqslant s-1)$. From expressions (1) and (4), in order to demonstrate that (6) is true for i we will have to demonstrate that the following holds:

$$\left(C_u - x_u^*\right) / \left(C_k - R_{(k)} + R_{(k-1)i} + 1\right) < \left(x_u^* + 1\right) / \left(R_{(k)} - R_{(k-1)i}\right), \tag{9}$$

where $\left(x_1^*, \ldots, x_{k-1}^*\right)$ is a mode of the $\mathrm{MH}\left(R_{(k-1)i};\, C_1, \ldots, C_{k-1}\right)$ associated to the node $\left(k-1,\, R_{(k-1)i}\right)$ and u is defined in Theorem 1. On the one hand, if inequality (6) is true for $i-1$, from expressions (1) and (5) we will have

$$\left(R_{(k)} - R_{(k-1)i} + 1\right) / x_v^* < \left(C_k - R_{(k)} + R_{(k-1)i}\right) / \left(C_v - x_v^* + 1\right),$$

with v defined in Theorem 1, and from this inequality we deduce

$$\left(R_{(k)} - R_{(k-1)i}\right) / \left(C_k + 1\right) < x_v^* / \left(C_v + 1\right). \tag{10}$$

On the other hand, as $\left(x_1^*, \ldots, x_{k-1}^*\right)$ is a mode of the $\mathrm{MH}\left(R_{(k-1)i};\, C_1, \ldots, C_{k-1}\right)$, from Theorem 2.1 given in Requena and Martín (2000) the inequalities

$$x_i^* / \left(C_i + 1\right) \leqslant \left(x_i^* + x_j^* + 1\right) / \left(C_i + C_j + 2\right) \leqslant \left(x_j^* + 1\right) / \left(C_j + 1\right)$$

hold for $i,\, j = 1, \ldots, k-1,\ i \neq j$, from where we can easily deduce

$$x_v^* / \left(C_v + 1\right) \leqslant \left(x_u^* + 1\right) / \left(C_u + 1\right), \tag{11}$$

with u and v defined as in Theorem 1. Finally, from inequalities (10) and (11) we find

$$(C_u + 1) / (C_k + 1) < \left(x_u^* + 1\right) / \left(R_{(k)} - R_{(k-1)i}\right),$$

from where we directly deduce inequality (9). Inequality (7) is demonstrated in a similar way. □

References

Aoki, S., 2002. Improving path trimming in a network algorithm for Fisher's Exact Test in two-way contingency tables. J. Stat. Comput. Simul. 72 (3), 205–216.

Clarkson, D.B., Fan, Y.A., Joe, H., 1993. A remark on algorithm 643: FEXACT: an algorithm for performing Fisher's Exact Test in $r \times c$ contingency tables. ACM Trans. Math. Software 19 (4), 484–488.

Hirji, K.F., Johnson, T.D., 1996. A comparison of algorithms for exact analysis of unordered $2 \times k$ contingency tables. Comput. Statist. Data Anal. 21, 419–429.

Joe, H., 1988. Extreme probabilities for contingency tables under row and column independence with application to Fisher's Exact Test. Comm. Statist. Theory Methods 17 (11), 3677–3685.

Mehta, C.R., Patel, N.R., 1980. A network algorithm for the exact treatment of the $2 \times k$ contingency table. Comm. Statist. Simulation Comput. 9 (6), 649–664.

Mehta, C.R., Patel, N.R., 1983. A network algorithm for performing Fisher's Exact Test in $r \times c$ contingency tables. J. Amer. Statist. Assoc. 78, 427–434.

Mehta, C.R., Patel, N.R., 1986a. Algorithm 643. FEXACT: A FORTRAN subroutine for Fisher's Exact Test on unordered $r \times c$ contingency tables. ACM Trans. Math. Software 12, 154–161.

Mehta, C.R., Patel, N.R., 1986b. A hybrid algorithm for Fisher's Exact Test in unordered $r \times c$ contingency tables. Comm. Statist. Theory Methods 15 (2), 387–403.

Requena, F., Martín, N., 2000. Characterization of maximum probability points in the Multivariate Hypergeometric Distribution. Statist. Probab. Lett. 50, 39–47.

Requena, F., Martín, N., 2003. The maximum probability $2 \times c$ contingency tables and the maximum probability points of the multivariate hypergeometric distribution. Comm. Statist. Theory Methods 32 (9), 1737–1752.

Shao, X.M., 1997. An efficient algorithm for the exact test on unordered $2 \times J$ contingency tables with equal column sums. Comput. Statist. Data Anal. 25, 273–285.

StatXact-5, 2001. A Statistical Package for Exact Nonparametric Inference. Cytel Software Corporation, Cambridge, MA.