Statistical Auditing and the AOQL-method

Talens, Erik

Chapter 6

EEOQL: The Exact Version of AOQL

6.1 Introduction

This chapter will apply the theory of the previous chapter to develop an algorithm to find the optimal sample size in the EOQL-procedure for fixed values of N, k_0, and P_l. Because the exact underlying distribution (hypergeometric) is used to find the optimal sample size, we have called this method the Exact Expected Outgoing Quality Limit (EEOQL) method. This algorithm will be described in Section 6.2. We will also use it in Section 6.3 to generate tables (triangular arrays) with values of $M^*(n, N)$, and thus $\pi^*(n, N)$, for all possible combinations of n and N.

Section 6.4 compares the sample sizes acquired by our EEOQL-method, the AOQL-method of Dodge and Romig, and the EOQL-method that uses an approximation for the underlying hypergeometric distribution.

Previously, we kept the value of k_0 fixed. Section 6.5 allows this value to vary in order to decide which value of k_0 we have to choose to minimize the expected value of items to be inspected.

The conclusions are discussed in Section 6.6.

6.2 Finding the optimal sample size

This section describes an algorithm that finds the optimal sample size, n^*, in the EEOQL-procedure for fixed values of N, k_0, and P_l. The following definition shows what we mean by the optimal sample size.

Definition 6.2.1. *The optimal sample size n^*, is the smallest value of n such that*

$$\pi^*(n, N) \leq P_l. \tag{6.2.1}$$

The algorithm heavily depends on the theory we developed in the previous chapter. The algorithm we use has two main advantages:

- It uses an efficient way of finding $M^*(n, N)$.
- It allows us to use recursive formulas to calculate the hypergeometric distribution function.

The first advantage is very helpful because to calculate $\pi^*(n, N)$ we need to know $M^*(n, N)$. The second advantage is important because to calculate the expected fraction of errors after inspection we need to calculate the hypergeometric distribution function. Remember that if $M \in \{1, \ldots, N\}$ and $n \in \{0, \ldots, N-1\}$, then

$$\pi(n, M, N) = \frac{M}{N}(1 - \frac{n}{N}) \cdot \Lambda(n, M-1, N-1).$$

The efficient way of finding M^* and thereby π^*, and an outline of the algorithm, are given in Subsection 6.2.1. Subsection 6.2.2 describes how we can use recursive formulas to calculate the hypergeometric distribution.

6.2.1 Description of the algorithm

We will assume that $0 \leq P_l \leq 1$, $n \in \{0, 1, \ldots, N\}$, $N \in \{1, 2, \ldots\}$, and $k_0 \in \{0, 1, \ldots\}$. There indeed exists an optimal sample size, because $\pi^*(\cdot, N)$ is non-increasing, $\pi^*(0, N) = 1$, and $\pi^*(N, N) = 0$. As the examples below show, sometimes a simple expression for π^* can be found. These cases can be used in the algorithm to find the optimal sample size in an efficient way.

Example 6.2.1. *(See Property 5.3.1, part 2). If $n \in \{0, \ldots, k_0\}$, then we find $M^*(n, N) = N$ and $\pi^*(n, N) = 1 - \frac{n}{N}$. From this it immediately follows that*

if $P_l \geq 1 - \frac{k_0}{N}$*, then this coincides with* $n^* \in \{0, \ldots, k_0\}$*, and* n^* *is the smallest value of* n *for which* $1 - \frac{n}{N} \leq P_l$ *holds, i.e.* $n^* = \lceil (1 - P_l) \cdot N \rceil$*. Notice that if* $N \in \{1, \ldots, k_0\}$*, then* $n^* = \lceil (1 - P_l) \cdot N \rceil$*.*

Example 6.2.2. *(See Property 5.3.1, part 3). If* $N \geq k_0 + 1$*, then we know that* $M^*(N-1, N) = k_0 + 1$ *and* $\pi^*(N-1, N) = \frac{k_0+1}{N^2}$*.*

Example 6.2.3. *If* $N = k_0 + 2$*, then*

a) $M^*(n, N) = N$ *and* $\pi^*(n, N) = 1 - \frac{n}{N}$ *for* $0 \leq n \leq k_0$*.*

b) $M^*(k_0 + 1, k_0 + 2) = k_0 + 1$ *and* $\pi^*(k_0 + 1, k_0 + 2) = \frac{k_0+1}{(k_0+2)^2}$*.*

Example 6.2.4. *If* $N = k_0 + 3$*, then*

a) $M^*(n, N) = N$ *and* $\pi^*(n, N) = 1 - \frac{n}{N}$ *for* $0 \leq n \leq k_0$*.*

b) $M^*(k_0 + 1, k_0 + 3) = k_0 + 1$ *and* $\pi^*(k_0 + 1, k_0 + 3) = 2\frac{k_0+1}{(k_0+3)^2}$*.*

c) $M^*(k_0 + 2, k_0 + 3) = k_0 + 1$ *and* $\pi^*(k_0 + 2, k_0 + 3) = \frac{k_0+1}{(k_0+3)^2}$*.*

We can use the examples above to initialize the algorithm. Diagram 1 describes this initialization. It deals with the following simple situations:

- The case $P_l \geq 1 - \frac{k_0}{N}$ gives $n^* = \lceil (1 - P_l) \cdot N \rceil \in \{0, \ldots, k_0\}$. This also covers the cases $N \in \{1, \ldots, k_0\}$.
- The case $P_l \leq \frac{k_0+1}{N^2}$ leads to $n^* = N - 1$ or $n^* = N$.
- Explicit solutions are given for n^* in case $N \in \{k_0 + 1, k_0 + 2, k_0 + 3\}$.

Therefore if $N \leq k_0 + 3$ the optimal sample size will be found by this initialization and if it does not find the optimal sample size, then we know that $N > k_0 + 3$ and

$$\pi^*(N-1, N) < P_l < \pi^*(k_0, N), \tag{6.2.2}$$

since we know $\pi^*(n, N)$ for $n \in \{0, \ldots, k_0, N-1, N\}$ after this initialization step.

Now, we could find the optimal sample size by calculating $\pi^*(k_0 + 1, N)$, $\pi^*(k_0 + 2, N)$, $\ldots$, until this value does not exceed P_l anymore. This means we decrease the right-hand side of (6.2.2) by increasing the sample size by one

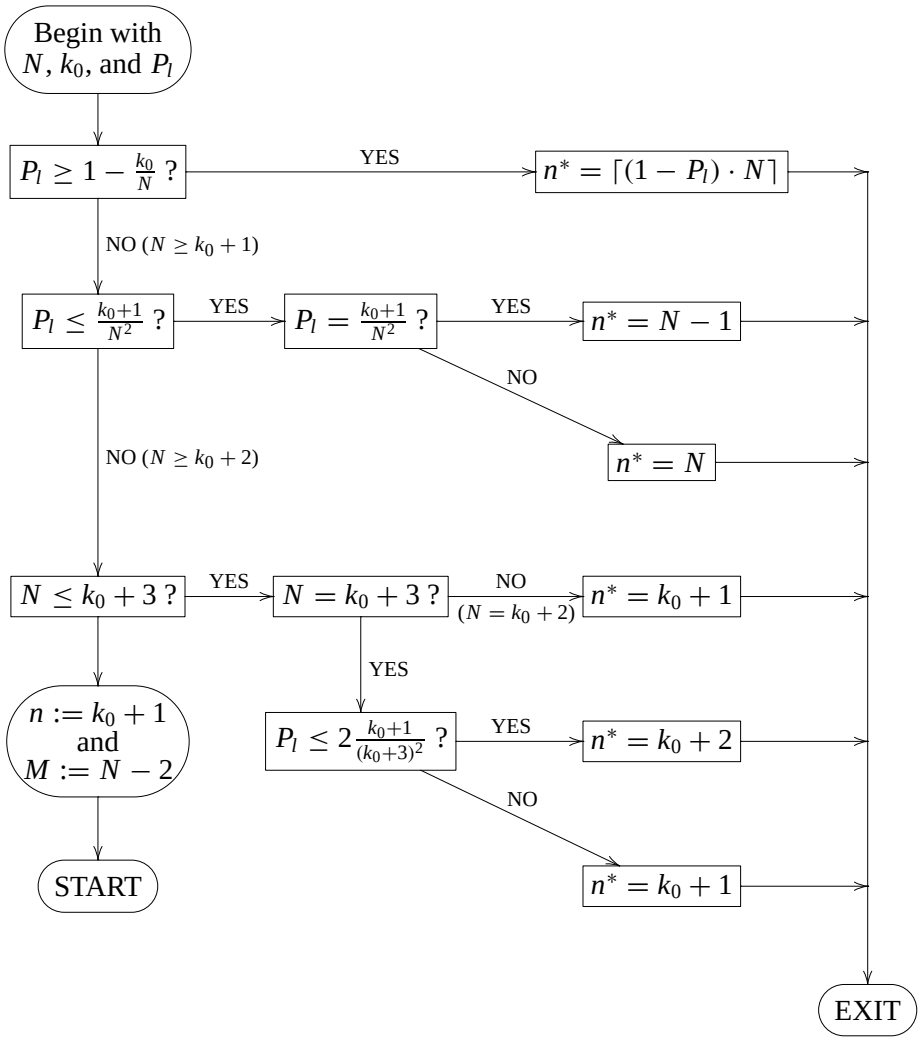

Diagram 1. Start of the algorithm.

unit starting at $k_0 + 1$ until we find that value for which π^* does not exceed P_l anymore. According to Definition 6.2.1 the value of n for which this happens is the optimal sample size. Since for each new value of n we have to find M^*, this may not be the most efficient method. Similarly we could try to increase the left-hand side of (6.2.2) by decreasing the sample size by one unit starting at $N - 2$ until we find that value for which P_l does not exceed π^* anymore. According

to Definition 6.2.1 the value of n for which this happens is the optimal sample size if $\pi^* = P_l$ and else the optimal sample size is this value plus one. Still the algorithm is inefficient. This chapter will construct a kind of mixture of these methods that fully exploits the theory developed in Chapter 5.

We will show that while we try to decrease the right-hand side of (6.2.2) by increasing the sample size by one unit, we can simultaneously increase the left-hand side. Theorem 5.3.8 is the key theorem in constructing this algorithm.

The first step in the kernel of the algorithm is to decrease the right-hand side of (6.2.2) by finding $\pi^*(k_0 + 1, N)$. Theorem 5.3.8, parts 1 and 3 tell us that $M^*(k_0 + 1, N) < M^*(k_0, N) = N$. To find $\pi^*(k_0 + 1, N)$, we can compute $\pi(k_0+1, M, N)$ starting at $M = M^*(k_0, N)-1 = N-1$ and decrease M by one unit until we have found $M^*(k_0 + 1, N)$. At the same time Theorem 5.3.8, part 2 shows that as long as $N - 1 \geq M \geq M^*(k_0 + 1, N)$ we know that $M^*(M, N) = k_0 + 1$. Since $N - 1 \geq M^*(k_0 + 1, N)$ we know that $M^*(N - 1, N) = k_0 + 1$ and $\pi^*(N-1, N) = \pi(N-1, k_0+1, N)$. However, we already processed this during the initialization of the algorithm. Now, suppose that $M = N - 2 \geq M^*(k_0 + 1, N)$, which is equivalent to $\pi(k_0 + 1, N - 2, N) \geq \pi(k_0 + 1, N - 1, N)$, then we know that $M^*(N-2, N) = k_0+1$ and $\pi^*(N-2, N) = \pi(N-2, k_0+1, N)$. Now there are two possibilities:

- If $\pi^*(N - 2, N) \geq P_l$, then according to the definition of n^* we obtain n^* as follows: if equality holds, then $n^* = N - 2$ and else $n^* = N - 1$.

- If $\pi^*(N - 2, N) < P_l$, we will continue our search for $M^*(k_0 + 1, N)$ by checking if $N - 3 \geq M^*(k_0 + 1, N)$, which is equivalent to $\pi(k_0 + 1, N - 3, N) \geq \pi(k_0 + 1, N - 2, N)$.

Thus while we are in the process of decreasing the right-hand side of (6.2.2) by increasing k_0 to k_0+1 we succeeded in increasing the left-hand side of (6.2.2) by decreasing $N - 1$ to $N - 2$. Suppose we have finally found $M^*(k_0 + 1, N)$, then we have also found $M^*(M, N)$ and thus $\pi^*(M, N)$ for $M = M^*(k_0 + 1, N), \ldots, N-1$. Note that since we have found $M^*(k_0+1, N)$ the algorithm has not yet stopped and thus $P_l > \pi^*(M^*(k_0 + 1, N), N)$. Therefore, by decreasing the right-hand side of (6.2.2) by increasing the sample size from k_0 to $k_0 + 1$, we have increased the left-hand side of (6.2.2) by decreasing the sample size

from $N-1$ to $M^*(k_0+1, N)$. If $\pi^*(k_0+1, N) \leq P_l$ then the algorithm stops and $n^* = k_0+1$, else we try to find $\pi^*(k_0+2, N)$. Again by Theorem 5.3.8, parts 1 and 3 we know that $M^*(k_0+1, N) - 1 \geq M^*(k_0+2, N)$, and by Theorem 5.3.8, part 2 we then know that $M^*(M^*(k_0+1, N)-1, N) = k_0+2$. If $\pi^*(M^*(k_0+1, N)-1, N)$ exceeds or equals P_l, then the algorithm stops, else we continue our search for $\pi^*(k_0+2, N)$ by checking if $M^*(k_0+1, N)-2 \geq M^*(k_0+2)$ and repeat the procedure above.

More generally we can say that we start our algorithm at $n = k_0+1$ and $M = N-2$ and the algorithm stops if the following equation does no longer hold:

$$\pi^*(M, N) < P_l < \pi^*(n, N). \tag{6.2.3}$$

While we try to decrease the right-hand side of (6.2.3) by increasing n to $n+1$ we are also able to increase the right-hand side by decreasing M from $M^*(n, N)-1$ to $M^*(n+1, N)$. Notice that by this algorithm the P_l gets 'ambushed' from two sides at once.

Theorem 5.3.8 tells us that we are only allowed to use the algorithm above as long as $n \leq n_l$. Remember that $n = n_l$ if and only if $M^*(n, N) = n$ or $n+1$. However, according to Remark 5.3.2 we then have found $\pi^*(n, N)$ for all possible sample sizes. If the algorithm has not yet finished, this means that $\pi^*(n_l+1, N) < P_l < \pi^*(n_l, N)$. From this it is immediately clear that $n^* = n_l+1$. A schematic overview of the kernel of the algorithm can be found in Diagram 2. The part of the diagram in which the question '$M = n$?' is posed prevents double calculations in case of $\pi^*(n_l-1, N) < P_l < \pi^*(n_l, N)$ and $M^*(n_l, N) = n_l$.

The algorithm poses the question '$M \geq M^*(n, N)$?'. We can check whether this inequality holds by comparing the value of $\pi(n, M, N)$ with the value of $\pi(n, M+1, N)$. As long as this difference is non-negative, then we know that $M \geq M^*(n, N)$ and if $\pi^*(M, N) < P_l$, and the algorithm continues with computing $\pi(n, M-1, N)$. If this difference equals zero, then we know $M = M^*(n, N)$, and if $\pi^*(M, N) < P_l$, we reduce M by one and know that this value of M is smaller than $M^*(n, N)$, therefore if $\pi^*(n, N) = \pi(n, M+1, N) > P_l$ and $M > n$ we continue with increasing n by one and calculating $\pi(n, M, N) = \pi(n, M^*(n-1, N)-1, N)$. If this difference is smaller than zero, then we

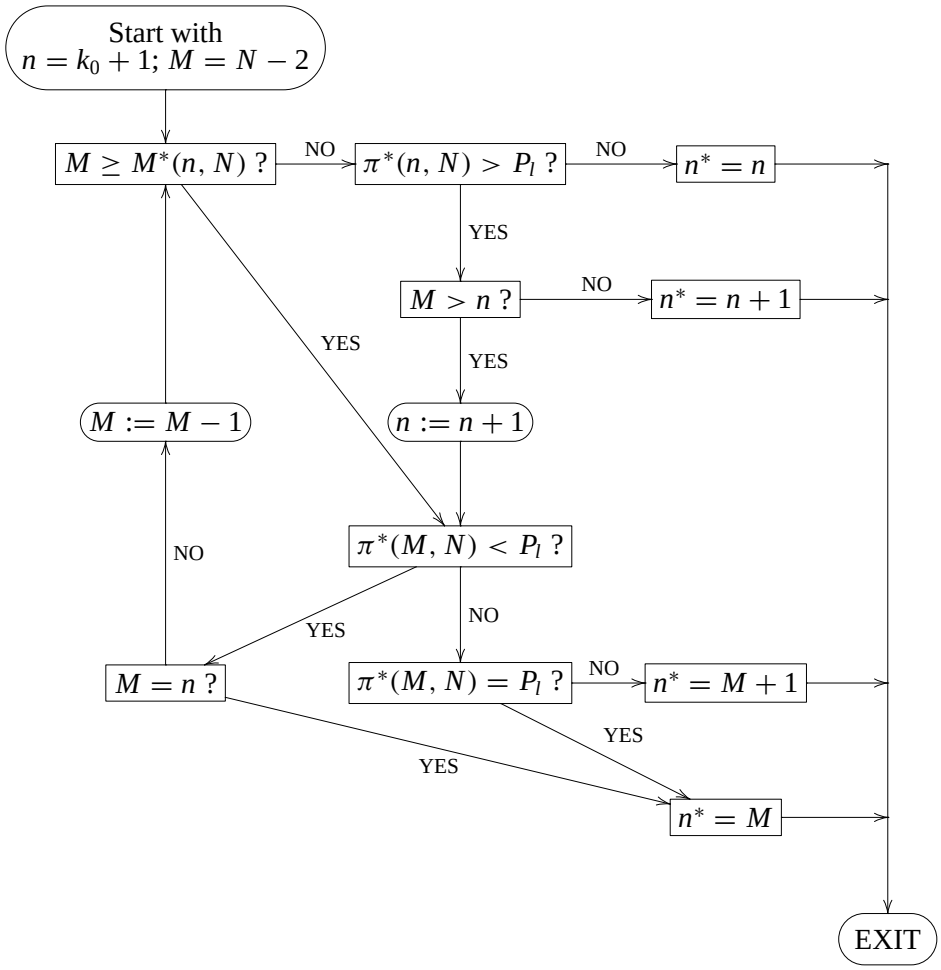

Diagram 2. Schematic description of the kernel of the algorithm.

know that $M = M^*(n, N) - 1$ and if $\pi^*(n, N) = \pi(n, M + 1, N) > P_l$ and $M > n$ we continue with increasing n by one and and calculating $\pi(n, M, N) = \pi(n, M^*(n - 1, N) - 1, N)$.

If we have to check whether $\pi(n, M, N) = \pi(n, M + 1, N)$, then, due to computational errors, equality will not always hold exactly when using a computer. Research did show that in cases where equality should hold the computer program gave an absolute error less than 10^{-14}, and in cases where equality should not hold the difference was far less than 10^{-10}. To be save we assumed

in the computer program that equality holds when the computer program gave an absolute difference less than 10^{-12}. We used the same accuracy in checking whether $\pi^*(n, N) = P_l$.

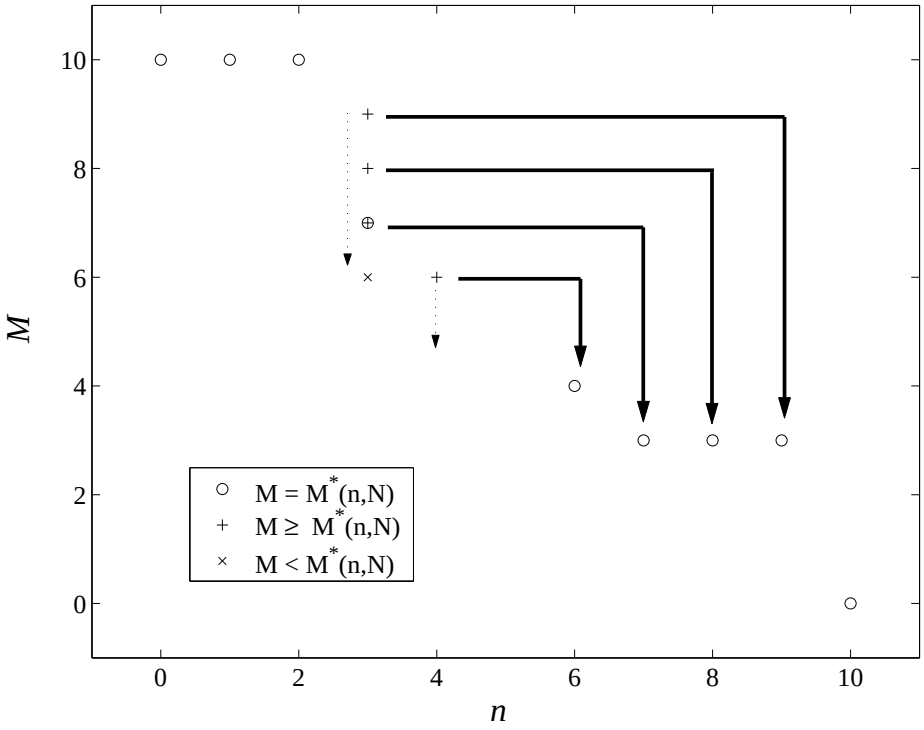

Figure 6.1. Graphical illustration of the algorithm for $N = 10$, $k_0 = 2$, and $\pi^*(6, 10) < P_l < \pi^*(3, 10)$.

A graphical illustration is given in Figure 6.1 for $N = 10$, $k_0 = 2$ under the assumption that $\pi^*(6, 10) < P_l < \pi^*(3, 10)$. The dotted arrows symbolise the process of trying to decrease the right-hand side of (6.2.3) and the bold arrows the 'bonus' of increasing the left-hand side while doing so. According to Example 6.2.1 we know $M^*(0, 10) = M^*(1, 10) = M^*(2, 10) = 10$. By Example 6.2.2 we know $M^*(9, 10) = 3$ and we also know that $M^*(10, 10) = 0$. Since we know that $\pi^*(6, 10) < P_l < \pi^*(3, 10)$, the algorithm will not finish during the initialization stage of the algorithm (see Diagram 1). Since we know that $\pi(3, 9, 10) > \pi(3, 10, 10) = 0$, we know that $9 \geq M^*(3, 10)$, but this implies that $M^*(9, 10) = 3$. The points $(3, 9)$ and $(9, 3)$ are a dual pair.

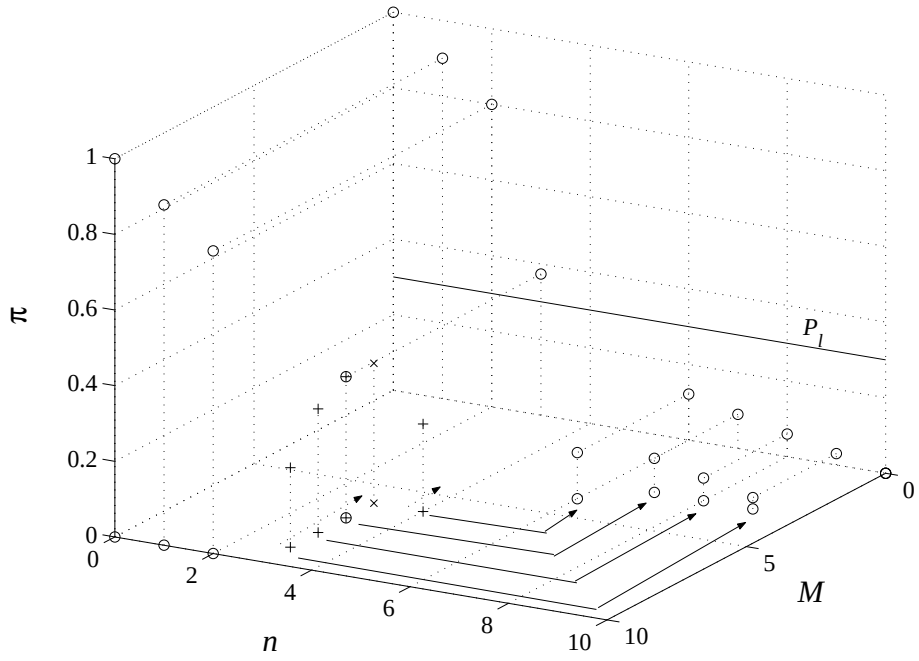

Figure 6.2. Three dimensional graphical illustration of the algorithm for $N = 10$, $k_0 = 2$, and $\pi^*(6, 10) < P_l < \pi^*(3, 10)$.

We already knew that $M^*(9, 10) = 3$ by Example 6.2.2. This information is processed in Diagram 1. Therefore the algorithm effectively starts in Diagram 2 with $n = k_0 + 1 = 3$ and $M = N - 2 = 8$ (instead of $M = 9$). Since $\pi(3, 8, 10) > \pi(3, 9, 10)$ we know that $8 \geq M^*(3, 10)$. This implies $M^*(8, 10) = 3$. The pairs $(3, 8)$ and $(8, 3)$ are dual. Now we have obtained a new M^* and can calculate $\pi^*(8, 10) = \pi(8, 3, 10)$. This is smaller than P_l. We note that $8 = M \neq n = 3$ and the algorithm continues by decreasing M by one unit and it becomes 7. We find that $7 \geq M^*(3, 10)$, because $\pi(3, 7, 10) > \pi(3, 8, 10)$. This gives $M^*(7, 10) = 3$, the pairs $(3, 7)$ and $(7, 3)$ are dual, and since $\pi^*(7, 10) < P_l$ and $7 = M \neq n = 3$ the algorithm continues with $M = 6$. Because $\pi(3, 6, 10) < \pi(3, 7, 10)$ we know that $6 < M^*(3, 10)$ and therefore $M^*(3, 10) = 7$. Hence we can calculate $\pi^*(3, 10) = \pi(3, 7, 10)$ and obtain that it is larger than P_l. We still have $6 = M > n = 3$. We increase n by one and it becomes 4. We know that $M^*(3, 10) > M^*(4, 10)$. This implies

$6 \geq M^*(4, 10)$ and therefore $M^*(6, 10) = 4$. The pairs $(4, 6)$ and $(6, 4)$ are dual. We have again obtained a new M^* and calculate $\pi^*(6, 10) = \pi(6, 4, 10)$ which is smaller than P_l. Since $6 = M \neq n = 4$ we can continue our algorithm by decreasing M with one unit and it becomes 5.

Figure 6.2 gives a representation of the case we described above in three dimensions, i.e. n, M, and π. Notice that the projection on the plane $\pi = 0$ gives the same result as Figure 6.1. We also projected $\pi^*(n, N) = \pi(n, M^*(n, N), N)$ on the plane $M = 0$. From this projection we can see that (6.2.3) is not violated yet, therefore the search for n^* continues.

The algorithm can stop because the right-hand side inequality of (6.2.3) is violated, or because the left-hand side inequality of (6.2.3) is violated, or finally because P_l lies between $\pi^*(n_l, N)$ and $\pi^*(n_l + 1, N)$ and then $n^* = n_l + 1$. Figure 6.3 illustrates these three examples. This figure treats the same case as

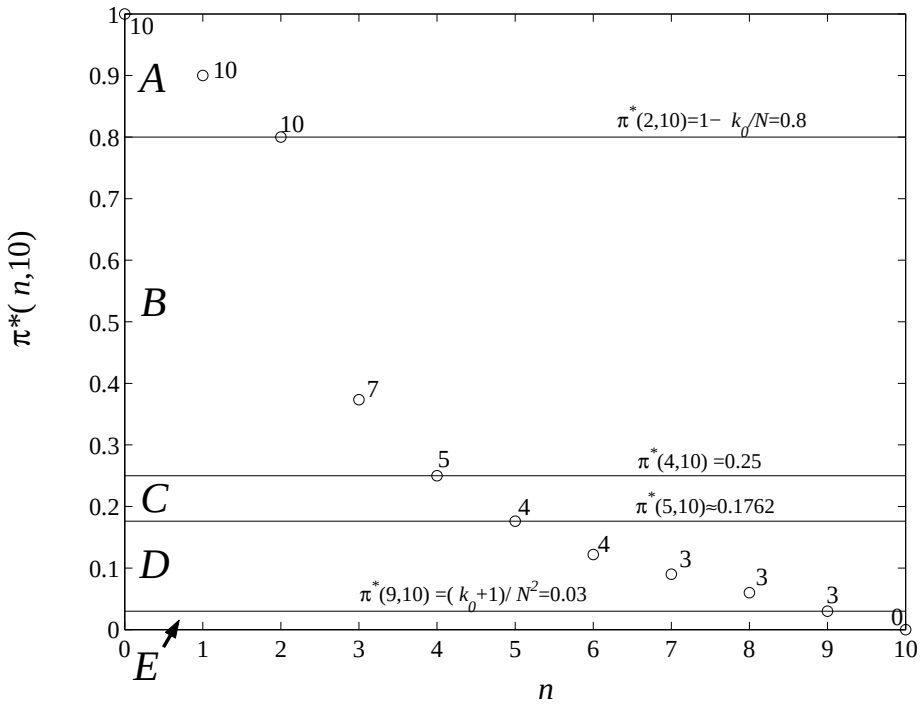

Figure 6.3. The values of $\pi^*(n, 10)$ for all possible values of n for $N = 10$, $k_0 = 2$, with the different stop-areas A, B, C, D, and E. The values of $M^*(n, 10)$ are also given.

before with $k_0 = 2$ and $N = 10$. This figure plots the values of $\pi^*(n, 10)$ with the corresponding values of $M^*(n, 10)$ for all possible values of n. If P_l lies in area A or E with inclusion of the boundaries, then the algorithm will stop during the initialization. If P_l lies in area B with inclusion of the lower boundary only, then the algorithm will stop because of a violation of the right-hand side inequality of (6.2.3). The algorithm will stop due to a a violation of the left-hand side inequality of (6.2.3), if P_l lies in area D with inclusion of the upper boundary only. Finally, if P_l is located within area C without inclusion of the boundaries, then the algorithm will stop because we have calculated $\pi^*(n, N)$ for all possible values of n by then and therefore $n^* = 5$, because $n_l = 4$.

6.2.2 Calculating the hypergeometric distribution

Chapter 4 deduced some recursive properties to calculate the hypergeometric distribution. Now, we will show how we use these formulas in the algorithm and show that the algorithm is indeed very efficient and accurate in calculating the hypergeometric distribution.

We know that in order to calculate $\pi(n, M, N)$ we have to calculate the value of $\Lambda(n, M-1, N-1)$, because if $M \in \{1, \ldots, N\}$ and $n \in \{0, \ldots, N-1\}$, then

$$\pi(n, M, N) = \frac{M}{N}(1 - \frac{n}{N}) \cdot \Lambda(n, M-1, N-1).$$

The values of π^* in the cases we treated in the initialization of the algorithm follow immediately from the Examples 6.2.1-6.2.4 and need no further explanation.

The kernel of the algorithm consists of three parts where a more detailed discussion about the calculation of Λ is needed. These three parts are:

A: The part that enables us to answer the question $M \geq M^*(n, N)$?

B: The part that enables us to continue our algorithm by increasing the sample size by one unit after the question $\pi^*(n, N) > P_l$? is answered affirmatively and M still exceeds n.

C: The part that enables us to answer the question $\pi^*(M, N) < P_l$?

Part *A*

The kernel of the algorithm starts at $n = k_0 + 1$ and $M = N - 2$. We have to check whether $M = N - 2 \geq M^*(k_0 + 1, N)$. To answer this question we need to compute and compare the values of $\pi(k_0 + 1, N - 1, N)$ and $\pi(k_0 + 1, N - 2, N)$. We start with computing $\pi(k_0 + 1, N - 1, N)$ and therefore we need to calculate $\Lambda(k_0 + 1, N - 2, N - 1)$. Note that $\pi(k_0 + 1, N, N) = 0$, because $\Lambda(k_0+1, N-1, N-1) = 0$. From Property 4.3.1, part 1 it is immediately clear that

$$\Lambda(k_0 + 1, N - 2, N - 1) = 0 + \frac{(k_0 + 1)\,!(N - 2)\,!}{k_0\,!(N - 1)\,!} = \frac{k_0 + 1}{N - 1}.$$

We continue with calculating $\pi(k_0 + 1, N - 2, N - 1)$ and therefore we have to calculate $\Lambda(k_0 + 1, N - 3, N - 1)$. Now, we are allowed to use Property 4.3.1, part 1 to calculate this quantity. If we apply this property to $\Lambda(k_0+1, N-3, N-1)$ we get

$$\begin{aligned}
&\Lambda(k_0 + 1, N - 3, N - 1) \\
&\qquad = \Lambda(k_0 + 1, N - 2, N - 1) + \frac{N - 3 - k_0 + 1}{N - 3 + 1} \times \\
&\qquad\quad \times \frac{N - 1 - (N - 3) - 1}{N - 1 - (N - 3) - (k_0 + 1) + k_0} \times \\
&\qquad\quad \times \big(\Lambda(k_0 + 1, N - 2, N - 1) - \Lambda(k_0 + 1, N - 1, N - 1)\big) \\
&\qquad = \frac{k_0 + 1}{N - 1} + \frac{N - k_0 - 2}{N - 2} \times \frac{k_0 + 1}{N - 1}.
\end{aligned}$$

More generally, suppose in our algorithm $M < M^*(n - 1, N) - 1$, then in order to calculate $\pi(n, M, N)$ for the purpose of checking whether $M \geq M^*(n, N)$, we need to calculate $\Lambda(n, M - 1, N - 1)$. If $k_0 + 1 \leq n \leq N - 2$ and $k_0 + 1 \leq M \leq N - n + k_0 - 1$, then this quantity can be calculated in a simple way by using the recursive property of Property 4.3.1, part 1. This gives

$$\begin{aligned}
\Lambda(n, M - 1, N - 1) = &\Lambda(n, M, N - 1) + \frac{M - k_0}{M} \times \frac{N - M - 1}{N - M - n + k_0} \times \\
&\times \big(\Lambda(n, M, N - 1) - \Lambda(n, M + 1, N - 1)\big).
\end{aligned}$$

From this property it is clear that we can calculate $\Lambda(n, M - 1, N - 1)$ with the help of $\Lambda(n, M, N - 1)$ and $\Lambda(n, M + 1, N - 1)$. Since $M < M^*(n, N) - 1$, we have already calculated these values of Λ while we calculated $\pi(n, M +$

1, N) and $\pi(n, M+2, N)$. Our algorithm is constructed in such a way that the conditions on n and M are always satisfied. For n this is immediately clear. The conditions on M are also satisfied. The value of M always exceeds k_0+1 in our algorithm because $M^*(n, N) \geq k_0+1$ and the algorithm stops if $M = n$. The value of M does not exceed $N-n+k_0-1$ because $M^*(n, N) \leq N-n+k_0$ and we will only apply this property if $M < M^*(n-1, N)-1$. We are always able to apply this property if $M < M^*(n-1, N)-1$, which will become clear further on. A graphical display of the above can be found in Figure 6.4, graph *A*.

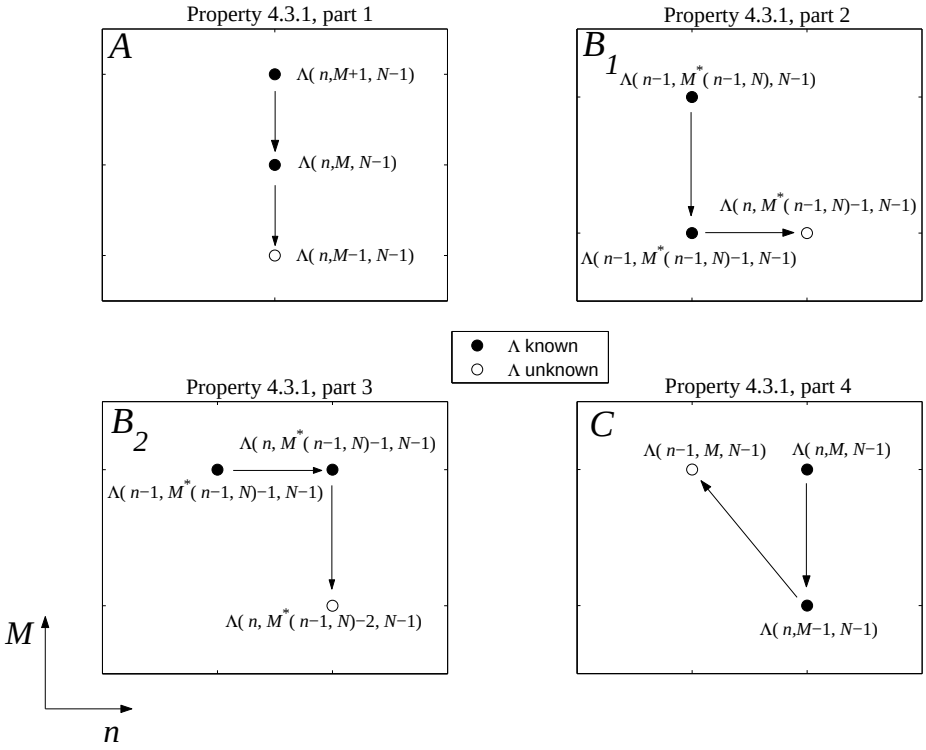

Figure 6.4. Schematic overview of the different recursive properties we use to calculate Λ in our algorithm.

Part B

Suppose we have found $M^*(n-1, N)$ and $\pi^*(n-1, N) > P_l$, then we increase the sample size by one and we have to calculate $\pi(n, M^*(n-1, N)-1, N)$ and therefore $\Lambda(n, M^*(n-1, N)-2, N-1)$ as the first step to find $M^*(n, N)$. We will do this in two steps:

B_1: We will compute $\Lambda(n, M^*(n-1, N)-1, N-1)$ from $\Lambda(n-1, M^*(n-1, N)-1, N-1)$ and $\Lambda(n-1, M^*(n-1, N), N-1)$. These values of Λ are known to us because we computed these values to find $\pi(n-1, M^*(n-1, N), N)$ and $\pi(n-1, M^*(n-1, N)+1, N)$.

B_2: We will compute $\Lambda(n, M^*(n-1)-2, N-1)$ from $\Lambda(n-1, M^*(n-1, N), N-1)$, which is already known to us, and from $\Lambda(n, M^*(n-1, N)-1, N-1)$, which we computed in the previous step.

We concentrate on the first step, B_1, first. Because the construction of the algorithm ensures that $k_0+2 \le n \le N-1$, because we have just increased n by one and we started the kernel of the algorithm at $n = k_0+1$. It also ensures that $k_0+2 \le M^*(n-1, N) \le N-n+k_0+1$, because $M^*(n-1, N) > M^*(n, N) \ge k_0+1$ and $M^*(n, N) \le N-n+k_0$. Therefore, we are allowed to use Property 4.3.1, part 2 to calculate $\Lambda(n, M^*(n-1, N)-1, N-1)$ from the values of $\Lambda(n-1, M^*(n-1, N)-1, N-1)$ and $\Lambda(n-1, M^*(n-1, N), N-1)$. This gives

$$\begin{aligned}
&\Lambda(n, M^*(n-1, N)-1, N-1) \\
&\quad = \Lambda(n-1, M^*(n-1, N)-1, N-1) - \frac{N-M^*(n-1, N)}{N-n} \times \\
&\qquad \times \frac{M^*(n-1, N)-k_0-1}{n-k_0-1} \times \big(\Lambda(n-1, M^*(n-1, N)-1, N-1) + \\
&\qquad - \Lambda(n-1, M^*(n-1, N), N-1)\big).
\end{aligned}$$

A graphical display can be found in Figure 6.4, graph B_1.

At the second step, B_2, we know the values of $\Lambda(n-1, M^*(n-1, N)-1, N-1)$ and $\Lambda(n, M^*(n-1, N)-1, N-1)$. The construction of the algorithm ensures that $k_0+2 \le M^*(n-1, N) \le N-n+k_0+1$, which we already proved at part B1, and $k_0+1 \le n \le N-1$, which is obvious. Hence, we are allowed to use Property 4.3.1, part 3 to calculate $\Lambda(n, M^*(n-1, N)-2, N-1)$ from

$\Lambda(n-1, M^*(n-1, N)-1, N-1)$ and $\Lambda(n, M^*(n-1, N)-1, N-1)$. This gives

$$\begin{aligned}\Lambda(n, M^*(n-1,N)-2, N-1)&\\ = \Lambda(n, M^*(n-1, N)-1, N-1) + \frac{n}{M^*(n-1, N)-1}\times&\\ \times\big(\Lambda(n-1, M^*(n-1, N)-1, N-1)+&\\ -\Lambda(n, M^*(n-1, N)-1, N-1)\big).&\end{aligned}$$

A graphical display can be found in Figure 6.4, graph B_2. Note that, if necessary, we are now allowed to use Property 4.3.1, part 1 again to calculate $\Lambda(n, M^*(n-1, N)-3, N-1)$ to find $\pi(n, M^*(n-1, N)-2, N)$ in our search for $M^*(n, N)$.

Part *C*

In our algorithm, if $M \geq M^*(n, N)$, then we know that $M^*(M, N) = n$ and we have to check if $\pi^*(M, N) \geq P_l$. To do so we have to calculate $\pi(M, n, N)$ and thus $\Lambda(M, n-1, N-1)$. If $n = k_0+1$, then $\Lambda(M, n-1, N-1) = \Lambda(M, k_0, N-1) = 1$ for all values of M and we have no problems calculating π. If $n > k_0+1$, then we use Theorem 4.1.1, part 1, which shows that $\Lambda(M, n-1, N-1)$ equals $\Lambda(n-1, M, N-1)$. To compute $\Lambda(n-1, M, N-1)$ we can use Property 4.3.1, part 4, since we know the values of $\Lambda(n, M-1, N-1)$ and $\Lambda(n, M, N-1)$. We computed the values of $\Lambda(n, M-1, N-1)$ and $\Lambda(n, M, N-1)$ while we computed $\pi(n, M, N)$ and $\pi(n, M+1, N)$. Because $k_0+1 \leq M \leq N-n+k_0$ and $k_0+1 \leq n \leq N-1$, we are allowed to use Property 4.3.1, part 4 to calculate $\Lambda(M, n-1, N-1)$. This gives

$$\begin{aligned}\Lambda(M, n-1, N-1)&\\ &= \Lambda(n-1, M, N-1)\\ &= \Lambda(n, M-1, N-1) + \frac{M-n}{n}\times\big(\Lambda(n, M-1, N-1)+\\ &\quad -\Lambda(n, M, N-1)\big).\end{aligned}$$

A graphical display can be found in Figure 6.4, graph C. For $M = M^*(n-1, N)-1$, we have to calculate $\pi^*(M^*(n-1, N)-1, N) = \pi(M^*(n-1, N)-1, n, N)$, and therefore $\Lambda(M^*(n-1, N)-1, n-1, N-1)$, which corresponds with $\Lambda(n-1, M^*(n-1, N)-1, N-1)$. But this value is already known

to us, because we computed this value when we computed $\pi^*(n-1, N) = \pi(n-1, M^*(n-1, N), N)$. Notice that if $M = M^*(n-1, N) - 2 \geq M^*(n, N)$, then we have to calculate $\pi^*(M^*(n-1, N) - 2, N)$. We already calculated the corresponding value of Λ when we calculated $\pi(n-1, M^*(n-1, N) - 1, N)$, unless $\pi(n-1, M^*(n-1, N), N)$ equals $\pi(n-1, M^*(n-1, N)+1, N)$, because then we did not compute $\pi(n-1, M^*(n-1, N) - 1, N)$.

We have shown that our algorithm calculates the hypergeometric distribution function in an efficient way. But we do not know yet how accurate our computation of the hypergeometric distribution function is. To check this we compared the values of the hypergeometric distribution function we calculated with the values that are computed according to Wu (1993). Wu showed that his method using prime number factorization to rewrite the factorials, and then using cancellation of common factors to simplify the computation, compares favourably to the method described in the IMSL-libraries (*IMSL libraries*, 1984). In the least favourable cases the differences between our computations and those according to Wu were smaller than 10^{-13} and often much smaller than this. Therefore, we can state that our approach is not only efficient but also very accurate.

6.3 Generating tables

This section describes how we will use the results from the previous chapter to generate tables (triangular arrays) for M^* and π^* simultaneously for all pairs (N, n), with $N = 0, \ldots, N_l$ and $n = 0, \ldots, N_l$, for a certain value of k_0. These tables can be generated up to arbitrary $N = N_l$. Notice that these tables are the same triangular arrays we mentioned in Chapter 5. We will denote these tables by $\{M_{N,n}\}$ and $\{\pi_{N,n}\}$. Their elements are denoted by $M_{N,n}$ and $\pi_{N,n}$. Of course the elements of these tables are of no interest to us if $n > N$. For $n = N$, the elements of both tables are filled with zeros because of Property 5.3.1, part 1. The elements of the first $k_0 + 1$ columns of $\{M_{N,n}\}$ and $\{\pi_{N,n}\}$ for which $n \leq k_0$ can be filled in the following way. According to Property 5.3.1, part 2, we know that $M_{N,n} = N$ and $\pi_{N,n} = 1 - \frac{n}{N}$ for $n \leq k_0$ and $N > n$. It is sensible to fill $\{M_{N,n}\}$ and $\{\pi_{N,n}\}$ column-wise. We start filling each column at $(n+1, n)$. We increase N by one until we reach N_l, then we go to the next column until

Table 6.1. The values of $M_{N,n}$ at $N, n \in \{0, \ldots, 7\}$ and $k_0 = 2$.

$N \setminus n$	0	1	2	3	4	5	6	7
0	0	-	-	-	-	-	-	-
1	1	0	-	-	-	-	-	-
2	2	2	0	-	-	-	-	-
3	3	3	3	0	-	-	-	-
4	4	4	4	3	0	-	-	-
5	5	5	5	3	3	0	-	-
6	6	6	6	4	3	3	0	-
7	7	7	7	5	4	3	3	0

$n = N_l$. According to Theorem 5.4.2, we have

$$\pi_{N,n} = \max\big(\pi(n, M_{N-1,n}, N), \pi(n, M_{N-1,n} + 1, N)\big). \tag{6.3.1}$$

If we know the value of $M_{N-1,n}$, then either this value or this value plus one are the only two possible values that $M_{N,n}$ can take. We fill $M_{N,n}$ with the value for which $\pi_{N,n}$ takes on the larger value. It is possible that the two feasible values of $M_{N,n}$ give the same value of $\pi_{N,n}$ (see Example 5.2.1), in this case $M_{N,n} = M_{N-1,n}$ as defined in Theorem 5.4.2.

Chapter 4 introduces $P(n, M, N)$ as a $(k_0 + 1)$-vector, with elements

$$P_j(n, M, N) = \mathrm{P}\{K = j | n, M, N\} = \frac{\binom{M}{j}\binom{N-M}{n-j}}{\binom{N}{n}},$$

for $j = 0, \ldots, k_0$, and also $\iota' = (1, \ldots, 1)$, a $(k_0 + 1)$-vector. Combining this with (6.3.1), we get

$$\pi_{N,n} = \max\Big(\frac{M_{N-1,n}}{N}(1 - \frac{n}{N}) \cdot \iota' \cdot P(n, M_{N-1,n} - 1, N - 1),$$
$$\frac{M_{N-1,n} + 1}{N}(1 - \frac{n}{N}) \cdot \iota' \cdot P(n, M_{N-1,n}, N - 1)\Big)$$

To calculate $\pi_{N,n}$, we have to calculate the probability vectors $P(n, M_{N-1,n} - 1, N - 1)$ and $P(n, M_{N-1,n}, N - 1)$. We can reduce the time needed to compute these values of the hypergeometric distribution function by using Properties 4.3.2, and 4.3.3. Since we have already computed $P(n, M_{N-1,n} - 1, N - 2)$

Table 6.2. The values of $\pi_{N,n}$ at $N, n \in \{0, \ldots, 7\}$ and $k_0 = 2$.

$N \setminus n$	0	1	2	3	4	5	6	7
0	0	-	-	-	-	-	-	-
1	1	0	-	-	-	-	-	-
2	1	0.5000	0	-	-	-	-	-
3	1	0.6667	0.3333	0	-	-	-	-
4	1	0.7500	0.5000	0.1875	0	-	-	-
5	1	0.8000	0.6000	0.2400	0.1200	0	-	-
6	1	0.8333	0.6667	0.3000	0.1667	0.0833	0	-
7	1	0.8571	0.7143	0.3265	0.1959	0.1224	0.0612	0

when we computed $\pi_{N-1,n}$, we can use Property 4.3.2 to compute $P(n, M_{N-1,n}-1, N-1)$. To compute $P(n, M_{N-1,n}, N-1)$, we can use Property 4.3.3 to compute this value from $P(n, M_{N-1,n}-1, N-2)$ again.
We start computing a column at $(n+1, n)$ and then we know that $M_{n+1,n} = k_0+1$ and we also know that $P_j(n, k_0, n) = 0$ for $j = 0, \ldots, k_0-1$ and $P_{k_0}(n, k_0, n) = 1$. We use $P(n, k_0, n)$ to calculate $\pi(n, k_0+1, n+1)$. From this it is clear that the conditions needed to use Properties 4.3.2 and 4.3.3 are always satisfied when calculating the values of P for subsequent elements of this column of $\{\pi_{N,n}\}$.

We fill the array column-wise, each time we have filled a column we increase n by one. This enables us to use Theorem 5.3.8 when we look at the array row-wise. Remember that n_l is the largest value of n for which $n \leq M_{N,n}$. Theorem 5.3.8 tells us that after we have found $M_{N,n}$ and it proves that $n \leq n_l$, then we immediately find the values of $\{M_{N,n}\}$ for other columns in the same row. To be more precise, then we know that $M_{N,j} = n$ for $j = M_{N,n}, \ldots, M_{N,n-1}-1$. Notice that Remark 5.3.2 shows that once we have found $M_{N,n}$ for all values of $n \in \{k_0+1, \ldots, n_l\}$, then we have found the values of $M_{N,n}$ for all the other cells of this row, i.e. for $n \in \{n_l+1, \ldots, N-1\}$. This property we can use in our array-filling algorithm, because we know that when we arrive at a particular cell in our algorithm and the value of $M_{N,n}$ is not yet known to us, then we know that $n \leq n_l$ and we are allowed to apply Theorem 5.3.8. Conversely, if this value is known to us when we arrive at the cell, then we know that $n > n_l$ and there is no need to apply Theorem 5.3.8, because all values of this row are already known

to us.

A more schematic overview of the procedure we use to generate tables is given below.

1. Fill the elements of $\{M_{N,n}\}$ and $\{\pi_{N,n}\}$ with -1 if $n > N$ and the other elements with zeros.

2. For $n \leq \min(k_0, N_l)$ and $N > n$ calculate the elements of $\{M_{N,n}\}$ and $\{\pi_{N,n}\}$ by $M_{N,n} = N$ and $\pi_{N,n} = 1 - \frac{n}{N}$.

3. If $N_l \leq k_0 + 1$, then stop, else set $n = k_0$.

4. Set $n = n + 1$ and $N = n + 1$. Fill $M_{N,n}$ with $k_0 + 1$ and $\pi_{N,n}$ with $\frac{k_0+1}{N^2}$. Set $P(n, k_0, n) = (0, \ldots, 0, 1)$.

5. Set $N = N + 1$.

6. Make the distinction between the following two cases.

   (a) If $M_{N,n} = 0$, use Properties 4.3.2 and 4.3.3 to calculate the corresponding values of P needed to fill $\pi_{N,n}$ and $M_{N,n}$ with the help of (6.3.1).
   Use Theorem 5.3.8 to fill the elements $M_{N,j}$ with the value n for $j = M_{N,n}, \ldots, M_{N,n-1} - 1$.

   (b) If $M_{N,n} > 0$, then we use the following procedure. In case $M_{N,n} = M_{N-1,n}$, we use Property 4.3.2 to compute the value of P needed to compute $\pi_{N,n}$. In case $M_{N,n} = M_{N-1,n} + 1$, we use Property 4.3.3 to compute the value of P needed to compute $\pi_{N,n}$.

7. Return to step 5 until $N = N_l$.

8. Return to step 4 until $n = N_l - 1$.

This procedure could be extended by using Theorem 5.3.1, which tells us that if $k_0 = 0$, then $M_{N,n} = \lceil \frac{N-n}{n+1} \rceil$.

We have shown that our algorithm calculates the hypergeometric distribution function in an efficient way. But we do not know yet how accurate our computation of the hypergeometric distribution function is. To check this we compared

Table 6.3. Pieces of the tables $\{M_{N,n}\}$ and $\{\pi_{N,n}\}$ at $N \in \{22, 23, 24\}$, $n \in \{20, \ldots, 24\}$ and $k_0 = 2$.

$N \setminus n$	20	21	22	23	24
$M_{N,n}$					
22	3	3	0	-	-
23	3	3	3	0	-
24	3	3	3	3	0
$\pi_{N,n}$					
22	0.0124	0.0062	0	-	-
23	0.0170	0.0113	0.0057	0	-
24	0.0208	0.0156	0.0104	0.0052	0

the values of the hypergeometric distribution function we calculated with the values that are computed according to Wu (1993) as we did in Section 6.2. In the least favourable cases the difference was smaller than 10^{-12} and often much smaller than this. Therefore, we can state that our approach is not only efficient but also very accurate.

Our algorithm also has to check whether

$$\pi(n, M_{N-1,n}, N) = \pi(n, M_{N-1,n} + 1, N)$$

Of course due to computational errors this will not always hold exactly when using a computer. Research shows that we may assume that equality holds if the computer gives a difference less than 10^{-12}.

If we would use these tables to find the optimal sample size in an EOQL-sampling plan for a certain predefined value of P_l with fixed k_0 and N, then we have to find the smallest value of n such that

$$\pi_{N,n} \leq P_l$$

Tables 6.1, 6.2, and 6.3 give some examples of pieces of tables generated for $k_0 = 2$. Notice that in Table 6.2 the optimal solution for $P_l = 0.01$ for all tabulated population sizes is to check the entire population. In fact, it turns out that $N = 18$ is the first value of N for which we can find an optimal solution ($n^* = 17$, with $M^*(17, 18) = 3$ and $\pi^*(17, 18) = 0.0093$) without checking the

entire population. Suppose we want to find the optimal n at $N = 24$, $k_0 = 2$ and $P_L = 0.01$. Using Table 6.3, we would find $n = 23$ as the optimal solution, because $n = 23$ is the smallest value of n for which the function $\pi^*(n, N) \leq 0.01$. But $\pi(23, 3, 24) = 0.0052$, which is considerably smaller then P_L. In this case it could well be that $n = 22$ would be preferred as optimal solution, because $\pi(22, 3, 24) = 0.0104$ lies considerably closer to P_L. This shows a clear advantage of using a table; it provides more insight.

Before using the theory we have developed in this chapter, it took us more than a day to generate $\{M_{N,n}\}$ and $\{\pi_{N,n}\}$ for $N_l = 2000$ and $k_0 = 2$ on a Pentium 4 computer. Using the algorithm we presented in this section it takes under two minutes to accomplish this task. This shows how efficient the algorithm actually is.

6.4 Comparison between the various methods

The previous sections described an algorithm how we can find the optimal sample size of the EOQL-method, which is an improvement of the AOQL-method, using the true underlying hypergeometric distribution instead of an approximation by the Poisson distribution. This section will compare the AOQL-method used by Dodge and Romig, the EOQL-method we discussed in Chapter 5 proposed by Simons et al. (1989), and the method developed by us. The first two methods use the approximation by the Poisson distribution and our method uses the true underlying hypergeometric distribution. We will call our method the EEOQL-method (Exact-EOQL).

Table 6.4, 6.5 and 6.6 give the result of our comparison of the optimal sample sizes under varying conditions of the three different approaches. From the tables we can see that the EOQL-method tends to be too conservative and the sample sizes of the AOQL-method tend to be too tight. Hence, the sampling costs are too high for the EOQL-method and the AOQL-method does not guarantee that the expected fraction of errors after inspection does not exceed P_l. Simons et al. already showed that for large N, the AOQL-method and the EOQL-method give the same optimal sample size. From the tables we can see that for large N, the EEOQL-method gives the same optimal sample size. This is exactly what we would expect, because for large N, the Poisson-approximation becomes better.

We can also see that for larger values of k_0 and smaller values of P_L, the differences between the methods become larger. These differences can be quite substantial.

Table 6.4. Comparison between the optimal sample sizes, using the EEOQL-method, the AOQL-method, and the EOQL-method at $P_L = 0.005$ for $k = 0, 1, 2$.

$P_L = 0.005$			
N	EEOQL	AOQL	EOQL
$k_0 = 0$			
50	38	29	50
75	47	38	63
100	50	43	70
150	58	50	74
250	64	57	74
500	68	64	74
750	70	67	74
1000	71	69	74
10000	73	74	74
$k_0 = 1$			
50	44	39	50
75	61	52	75
100	75	63	100
150	94	80	117
250	117	101	134
500	139	126	149
750	148	138	155
1000	153	144	158
10000	167	166	167
$k_0 = 2$			
50	46	43	50
75	66	59	75
100	84	74	100
150	113	97	142
250	151	131	174
500	198	177	212
750	220	201	229
1000	232	215	239
10000	270	267	271

Table 6.5. Comparison between the optimal sample sizes, using the EEOQL-method, the AOQL-method and the EOQL-method at $P_L = 0.01$ for $k_0 = 0, 1, 2$.

$P_L = 0.01$			
N	EEOQL	AOQL	EOQL
$k_0 = 0$			
50	25	22	35
75	29	25	37
100	31	27	37
150	33	30	37
250	34	32	37
500	36	35	37
750	36	36	37
1000	36	36	37
10000	37	37	37
$k_0 = 1$			
50	38	32	50
75	47	40	59
100	54	46	65
150	62	54	70
250	70	63	75
500	77	72	79
750	79	76	81
1000	80	78	82
10000	84	84	84
$k_0 = 2$			
50	42	37	50
75	57	49	71
100	67	58	79
150	83	72	92
250	99	89	106
500	116	108	120
750	123	116	125
1000	126	121	128
10000	136	136	137

Table 6.6. Comparison between the optimal sample sizes, using the EEOQL-method, the AOQL-method and the EOQL-method at $P_L = 0.05$ for $k_0 = 0, 1, 2$.

$P_L = 0.05$			
N	EEOQL	AOQL	EOQL
$k_0 = 0$			
50	7	7	8
75	7	7	8
100	7	7	8
150	7	8	8
250	7	8	8
500	7	8	8
750	7	8	8
1000	7	8	8
10000	7	8	8
$k_0 = 1$			
50	14	13	15
75	15	14	16
100	16	15	16
150	16	16	17
250	16	16	17
500	17	17	17
750	17	17	17
1000	17	17	17
10000	17	17	17
$k_0 = 2$			
50	20	18	22
75	22	21	23
100	24	22	24
150	25	24	25
250	26	25	26
500	27	26	27
750	27	27	27
1000	27	27	28
10000	28	28	28

The EOQL-method with the Poisson-approximations, uses the following definition for the expected fraction of errors after inspection

$$\pi(n, M, N) = \sum_{k=0}^{\min(k_0, M)} \frac{M-k}{N} \mathrm{P}\{K = k|n, M, N\}\,.$$

This method approximates the hypergeometric probability

$$\mathrm{P}\{K = k|n, M, N\} = \frac{\binom{M}{k}\binom{N-M}{n-k}}{\binom{N}{n}},$$

by the Poisson probability

$$\mathrm{P}\{K = k|n, M, M\} = \frac{e^{-\lambda}\lambda^k}{k!}, \text{ with } \lambda = \tfrac{n \cdot M}{N}.$$

But we can also write π as

$$\pi(n, M, N) = \frac{M}{N}(1 - \frac{n}{N}) \cdot \sum_{k=0}^{k_0} \mathrm{P}\{K = k|n, M-1, N-1\}\,,$$

and approximate the hypergeometric probability

$$\mathrm{P}\{K = k|n, M-1, N-1\} = \frac{\binom{M-1}{k}\binom{N-M}{n-k}}{\binom{N-1}{n}},$$

by the Poisson probability

$$\mathrm{P}\{K = k|n, M-1, N-1\} = \frac{e^{-\lambda}\lambda^k}{k!}, \text{ with } \lambda = \tfrac{n \cdot (M-1)}{N-1}.$$

We will call this approach the modified EOQL-method. Table 6.7 compares this modified EOQL-method with our exact approach for different values of k_0 and P_l. The values of the optimal sample sizes of the modified EOQL-approach lie closer to the values of the exact approach than the values of the AOQL- and EOQL-approach. However, the modified EOQL-method still differs from the exact method. Especially for small values of k_0 and P_l, and small population sizes the modified EOQL approach gives optimal sample sizes which are too small, and for larger population sizes it gives optimal sample sizes which are too large. For very large population sizes the modified EOQL-method and the exact method give the same optimal sample size again.

Table 6.7. Comparison between the optimal sample sizes, using the EEOQL-method (E) and the modified EOQL-method (P) at different values of P_L for $k_0 = 0, 1, 2$.

	$P_L = 0.5\%$		$P_L = 1\%$		$P_L = 5\%$	
$k_0 = 0$						
N	E	P	E	P	E	P
50	38	34	25	26	7	8
75	47	45	29	32	7	8
100	50	53	31	34	7	8
150	58	63	33	36	7	8
250	64	70	34	37	7	8
500	68	73	36	37	7	8
750	70	74	36	37	7	8
1000	71	74	36	37	7	8
10000	73	74	37	37	7	8
$k_0 = 1$						
N	E	P	E	P	E	P
50	44	42	38	36	14	15
75	61	58	47	47	15	15
100	75	72	54	54	16	16
150	94	93	62	63	16	16
250	117	118	70	72	16	17
500	139	143	77	78	17	17
750	148	152	79	80	17	17
1000	153	156	80	82	17	17
10000	167	167	84	84	17	17
$k_0 = 2$						
N	E	P	E	P	E	P
50	46	45	42	40	20	20
75	66	63	57	53	22	22
100	84	79	67	64	24	24
150	113	107	83	80	25	25
250	151	146	99	99	26	26
500	198	197	116	117	27	27
750	220	220	123	123	27	27
1000	232	233	126	127	27	27
10000	270	270	136	137	28	28

6.5 Choice of k_0

Previously, we considered k_0 to be fixed. If we allow k_0 to vary, we can minimize the expected number of items to be inspected. We denote the number of items to be inspected by I, and we denote $W = \mathrm{E}(I)$. If we find less or equal than k_0 defects in the sample we have to inspect n items, and if we find more than k_0 defects we have to inspect the entire population. This gives the following expression for $\mathrm{E}(I)$,

$$\begin{aligned} W(k_0, n, M, N) &= \mathrm{E}(I(k_0, n, M, N)) \\ &= n \cdot \mathrm{P}\{K \leq k_0 | n, M, N\} + N \cdot \mathrm{P}\{K > k_0 | n, M, N\} \\ &= N - (N - n) \cdot \mathrm{P}\{K \leq k_0 | n, M, N\}\,. \end{aligned}$$

Using the EEOQL-procedure, we can find the optimal sample size n^*, which depends on N, k_0 and P_l. If we look at W for various values of k_0 keeping N and P_l fixed, then the values of k_0 are accompanied by the corresponding values of $n^*(N, k_0, P_l)$. The expected number of items to be inspected also depends on M, and this is an unknown quantity to us. To make a sensible choice out of the possible values of k_0, we first have to observe the behaviour of W for different values of k_0 while we vary M. Therefore we study the curves $W_{k_0}(M)$ for varying combinations of N and P_l. These curves are defined by

$$W_{k_0}(M|N, P_l) = N - (N - n^*(N, k_0, P_l)) \cdot \mathrm{P}\left\{K \leq k_0 | n^*(N, k_0, P_l), M, N\right\}.$$

Notice that $W_{k_0}(M)$ is equal to $n^*(N, k_0, P_l)$ if $M \leq k_0$, and $W_{k_0}(M) = N$ if $M > N - n^*(N, k_0, P_l) + k_0$. For other values of M the function $W_{k_0}(M)$ is a monotone increasing function of M. Figure 6.5 shows $W_{k_0}(M)$ for different values of k_0 while we vary P_l keeping N fixed. Figure 6.6 shows $W_{k_0}(M)$ for different values of k_0 while we vary N keeping P_l fixed. Of course, we are interested in $\min_{k_0} W_{k_0}(M)$. We will denote the value of k_0 for which $W_{k_0}(M)$ takes on its minimum by $k_0^*(M)$. The figures show that $\min_{k_0} W_{k_0}(M)$ is very sensitive in M. By sensitive we mean that a small change in M will change the value of k_0 for which $W_{k_0}(M)$ will take on its minimum. In fact, the smaller N and P_l are, the more sensitive $\min_{k_0} W_{k_0}(M)$ becomes. However, the differences between the different values of $W_{k_0}(M)$ become smaller, choosing the wrong value of k_0 has less consequences for the expected amount of work to be done.

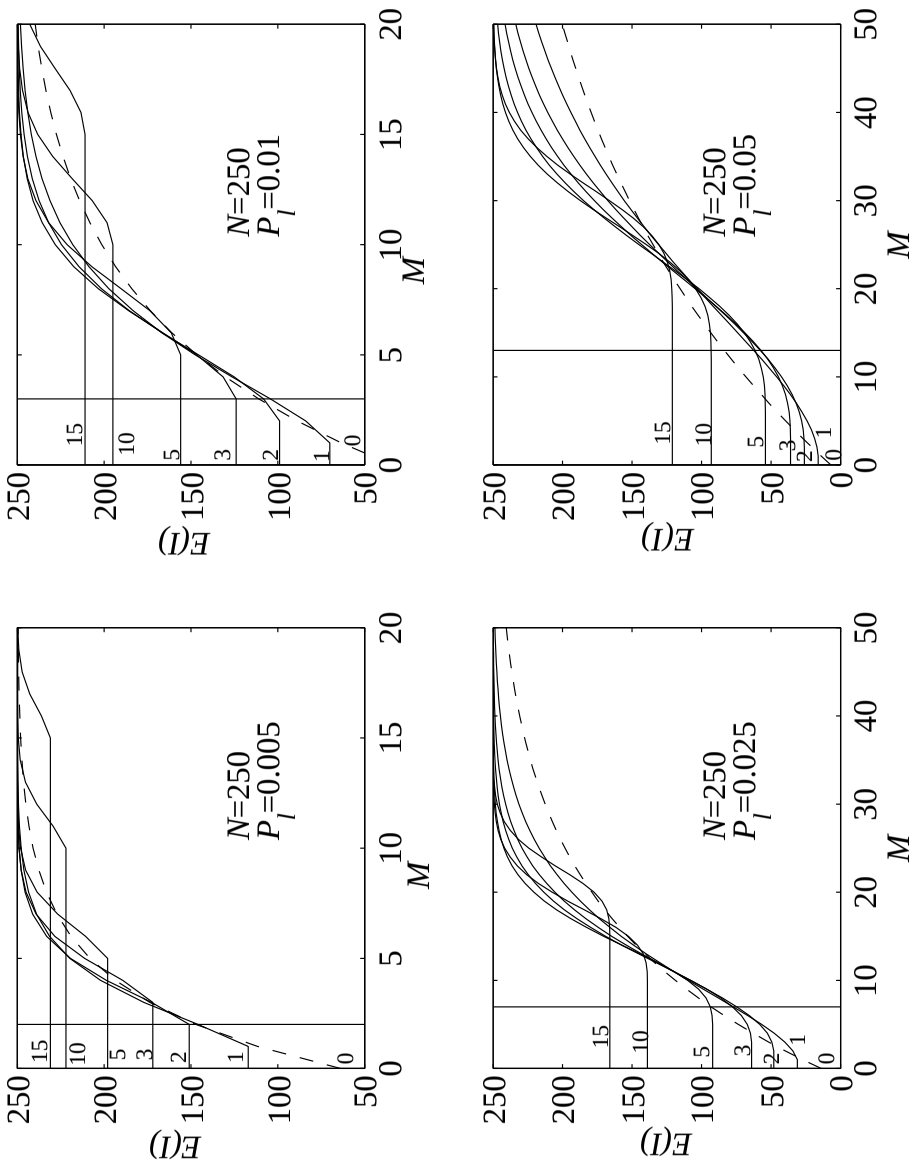

Figure 6.5. The function $W_{k_0}(M)$ for values of $k_0 \in \{0, 1, 2, 3, 5, 10, 15\}$ ($k_0 = 0$ is striped), at $N = 250$ and for values of P_l equal to 0.5%, 1%, 2.5%, and 5%. The vertical line equals $\lceil N \cdot P_l \rceil$.

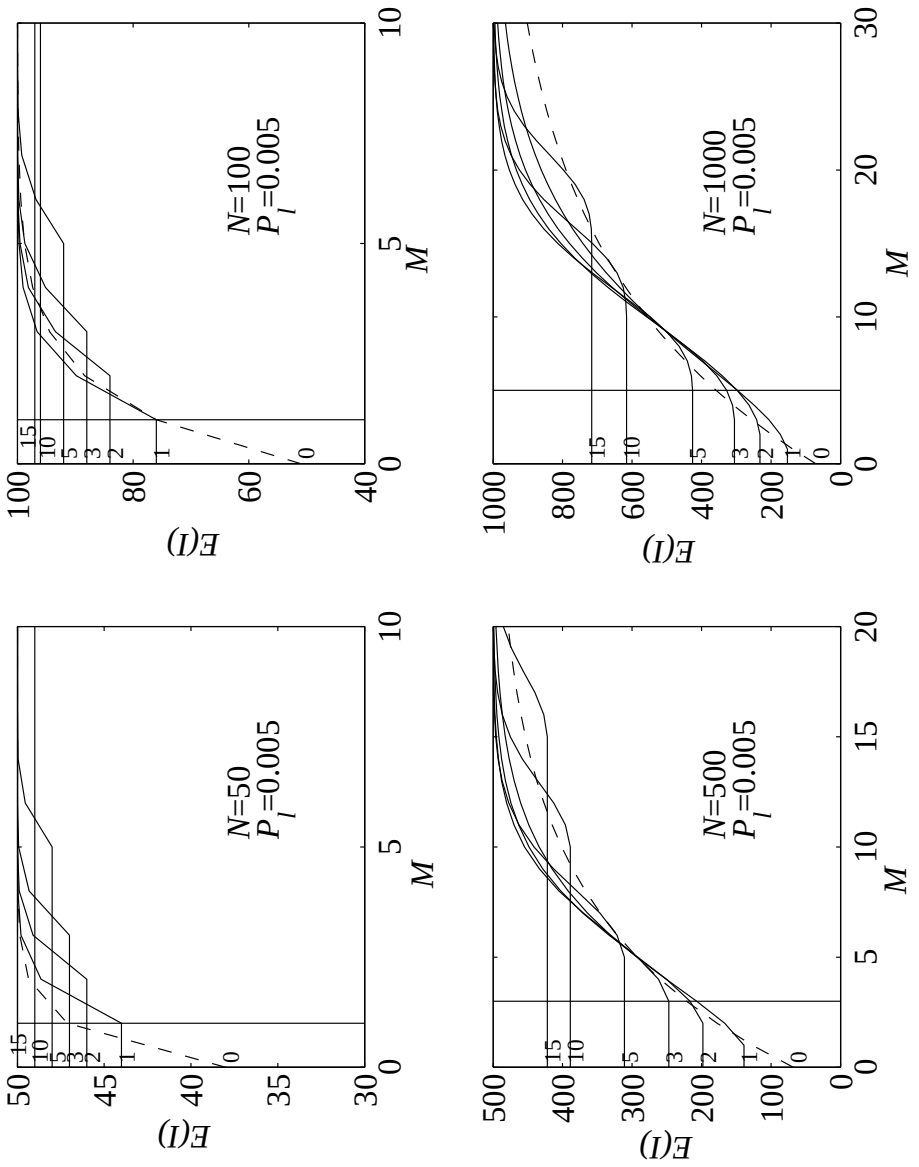

Figure 6.6. The function $W_{k_0}(M)$ for values of $k_0 \in \{0, 1, 2, 3, 5, 10, 15\}$ ($k_0 = 0$ is striped), at $P_l = 0.5\%$ and for values of N equal to 50, 100, 500, and 1000. The vertical line equals $\lceil N \cdot P_l \rceil$.

Another observation we make from observing these curves is that $k_0^*(M)$ seems to increase with M.

Since we do not have any knowledge of the true value of M, it is impossible to choose a value of k_0 that minimizes the expected amount of items to be inspected. Hence, finding a minimizing value of k_0 is an illusion. If we have some information (confidence interval) or make some assumptions on what the value of M might be, finding an minimizing k_0 will still be very difficult, if not impossible, due to the sensitivity of k_0^* in M.

Looking at Figure 6.5 and 6.6 a strategy could be constructed which could minimize the amount of work in the long run. If we would apply the EEOQL-procedure several times in a row, it can be expected that M lies in the neighbourhood (perhaps a bit less) of the number of errors allowed in the population, i.e. $N \cdot P_l$. Because $N \cdot P_l$ can take a non-integer value, in contrast to M, we will use $\lceil N \cdot P_l \rceil$ instead of $N \cdot P_l$. If our assumption holds, it is not difficult to see that choosing k_0 equal to zero (striped line in the figures) would not be very cost effective in most cases, although this is often done in practice. Choosing $k_0 = k_0^*(\lceil N \cdot P_l \rceil)$ would be minimizing if the true value of M equals $\lceil N \cdot P_l \rceil$. For values of M not much larger than this value this choice of k_0 might not be minimizing anymore, probably it will be minimizing for a value of k_0 larger than the one we chose, but if we look at Figure 6.5 and 6.6 the difference with the minimizing solution seems to be relatively small. For values of M not much smaller than $\lceil N \cdot P_l \rceil$ this choice of k_0 might not be minimizing any more too, most likely it will be minimizing for some value of k_0 smaller than the one we chose, but looking at Figure 6.5 and 6.6 the difference with the minimizing solution can be quite substantial. By choosing $k_0 = k_0^*(\lceil N \cdot P_l \rceil) - 1$, it seems we solve this problem largely and still the difference with the minimizing solution for values of M not much larger than $\lceil N \cdot P_l \rceil$ will not become too large. For very small values of M and for relatively large values of M the difference with the minimizing solution still can be very substantial. But for very large values of M one would expect an inspection of the entire population, and probably this will give insight in what causes the high number of defects. Measures will be taken to solve this problem and probably the value of M will decrease rapidly. Therefore, the method will not stay inefficient for a long time.

We will also study the behaviour of $W_M(k_0)$. This means we vary k_0 while we

fix M. We are especially interested in values of M equal to $M^*(n^*(N, k_0, P_l), N)$. It is possible that for different values of k_0 we find the same value of M^*. Now, we denote by $K(M)$ the set of all possible values of k_0 for which $M = M^*(n^*(N, k_0, P_l), N)$. This could also be an empty set.

Conjecture 6.5.1. *Let $K(M)$ be a non-empty set. If $M > 0$, then $\min_{k_0} W_{M-1}(k_0)$ takes on its minimum for a value of $k \in K(M)$.*

Note that if $K(M)$ has only one element, we exactly know for which value of k_0 the function $\min_{k_0} W_{M-1}(k_0)$ takes on its minimum. The difference between the minimal solution and the solution for the other elements of $K(M)$ is not very large. This conjecture shows that if we find a certain optimal sample size $n^*(N, k_0, P_l)$ with corresponding $M^*(n^*(N, k_0, P_l), N)$, then in the neighbourhood of this M^* the amount of work to be done when we use this value of k_0 will not differ too much from the minimal solution.

Further research should be done to determine the way in which k_0 could be chosen. Here, we only gave some heuristic solutions.

6.6 Conclusions

The Exact Expected Outgoing Quality Limit (EEOQL) method is a very useful control instrument, which is easy to implement with the help of general software. It does not make the mistake Dodge and Romig made in their AOQL-method, which leads to sample sizes that are too low. Hence, the AOQL-method gives a false feeling of reassurance, because this method does not guarantee that the expected fraction of errors after inspection does not exceed the predefined level set by management. The EEOQL-method uses the underlying hypergeometric distribution instead of the Poisson approximation. This Poisson approximation leads to sample sizes that are too large and therefore it leads to sampling costs that are too high. The EEOQL-method is both efficient as well as accurate. Especially for relatively small values of N and P_l, our EEOQL-method compares favourably to the AOQL-method of Dodge and Romig and the EOQL-method of Simons et al. (1989).