AN ELEMENTARY APPROACH TO THE SECRETARY PROBLEM

Lucas Monteiro CHAVES[1]
Devanil Jaques de SOUZA[1]

- *ABSTRACT: Using an interpretation of the factorial, based on counting paths in a diagram, an elementary approach to the traditional secretary problem is developed.*

- *KEYWORDS: Secretary problem; paths; optimum strategy; expected rank.*

1 Introduction

The secretary problem, a classical problem in stochastic optimization, and its variations, have a long history, retracing to Caley (1880). In the literature it assumes some other names as the Problem of the Princess, the dowry problem or the Quandary of the Policeman, in game theory texts. In fact, this subject became a huge research area with an enormous number of publications. The simplicity and various smart solutions made of this problem a regular example in the introduction of theories as, for example, sequential analysis and game theory. There are no new results in this article, except, perhaps, by the expected final rank on section 6. What is new is a surprisingly elementary approach, based on counting paths in an array of points, that allows an intuitive comprehension of the problem. Such an approach is strong enough to permit the construction of all proofs, even the less trivial ones. This is the reason why our bibliography is so short. A much more complete list of references for the problem of secretary may be found in the survey by Freeman. Inspiration for this work came from the excellent text by Landim and from Feller's classical book.

2 The secretary problem

A number n of applicants shows up for a single secretary position. They are interviewed in random order and all $n!$ possible orders are equally likely. Immediately after each interview the applicant is accepted or rejected and, once rejected, she (or he) can't be recalled. According to their abilities, they can be ordered from the best (rank 1) to the worst (rank n) but the interviewer can only ascertain relative ranks, by comparing the applicant to those already interviewed. For example, suppose that four applicants have already been interviewed and ordered by absolute ranks as 4-1-2-3. It means that the first one is the worse, the second the better, and so on. It must be noted that those numbers

[1] Departamento de Ciências Exatas, Universidade Federal de Lavras – UFLA, Caixa Postal Box 37, CEP: 37200-000, Lavras, MG, Brazil. E-mail: *lucas@ufla.br / djs@oi.com.br*

change after each new interview. In the example, if the fifth happens to be the second better, the order becomes 5-1-3-4-2. Equivalently, the corresponding sequence of relative ranks would be 1-1-2-3 and 1-1-2-3-2. The process stops when one is accepted. Case no one has been accepted, the *n-th* is hired necessarily.

The question to be addressed is: what should be the procedure, the strategy, in order to maximize the probability of choosing the best applicant?

Let us consider some possibilities. If we decide always for the first (or the *i-th*, i=1,2…n) applicant, the probability of choosing the best is 1/n. A more elaborated strategy is to reject half (supposing n even) of them and to choose the first better than all previous. In this case, if the second better is in the first half and the best is not, this strategy will lead necessarily to the best. Such a situation occurs with probability $1/2$ (the second better candidate is among the first n/2 interviewed) times $\frac{n/2}{(n-1)}$ (the conditional probability that the best applicant being among the last n/2, given that the second is in the first half) = $\frac{n}{4(n-1)}$. There are other situations that also result in success for this strategy. For example, the third better candidate in the first half and the first is in the second half but before the second one. This justifies the inequality

$$\mathrm{P}\left[success\right] > \frac{n}{4(n-1)} > \frac{1}{4} \quad \forall n \geq 2 \text{ and}$$

$$Lim_{n\to\infty}\mathrm{P}\left[success\right] = \frac{1}{4}$$

Is there a better strategy?

Since, if you are not the best in a subset you can not be the best in the set, the first fact is: the interviewer has to decide to stop or not the process only when the last interviewed has relative rank one (we call them *candidates*). Two questions arise: (1) given a candidate, what is the probability that she (or he) is really the best? (2) does some value of this probability justify interrupting the process?

The solution for these questions will be given in an elementary form based on some interpretation of the factorial of n.

3 Development

Let us consider a triangular array of points, beginning with one point at first column, two points at second, and so on, finishing with n points at *n-th* column, as described by the Figure 1, in the case $n = 3$.

Consider, in this triangular array, oriented paths obtained by joining points from left to right, begining at point 1 and finishing at any point at *n-th* column. As each point of the *k-th* column may be joined to any of the (k+1) points at (k+1)ith column, the total number of paths is $n!$. In Figure 1 this number is $3!$. This construction gives a geometrical interpretation of the factorial. The authors believe that this approach may be useful in the solution of various combinatorial problems with the advantage of being extremelly didatical. Author couldn't find any explicity reference for this construction but these ideas are clearly used by Feller when treating the reflection principle, a classical combinatorial problem (Feller, p. 72).

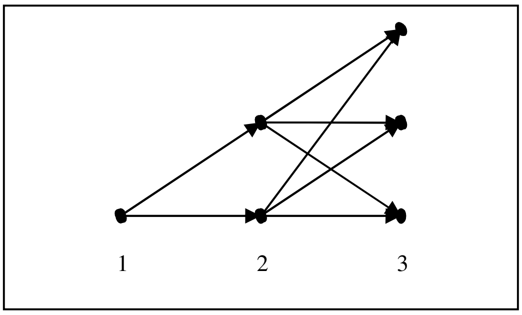

Figure 1 - Triangular array of points for n=3.

The central idea is to represent the sequence of relative ranks by a path in a diagram similar to figure I. We use the matrix notation but lines are enumerated down to up. With this notation, a point (k,r) will belong to the path if, and only if, the *k-th* applicant has relative rank *r* just after his (or her) interview, *i.e.* in the *k-th* interview. Let us review our initial example 4-1-2-3. We have to recover the sequence of relative ranks. The first is always 1. The second is better that the first and therefore we have 1-1. The third is better that the first and worse than the second and so we have 1-1-2 and finally 1-1-2-3. The corresponding path is shown in Figure 2.

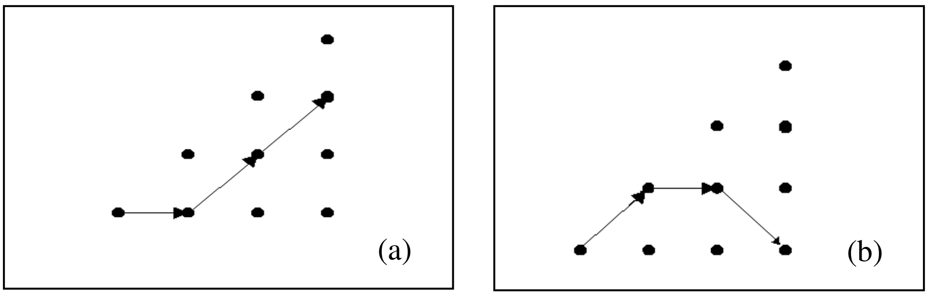

Figure 2 - Path representing. (a) relative ranks 1-1-2-3. (b) relative ranks 1-2-2-1.

Conversely, any path represents a possible sequence of relative ranks and allows recovering the sequence of absolute ranks. The path in Table 1 represents the sequence.

Table 1 - Sequence of relative ranks and allows recovering the sequence of absolute ranks

Step	Relative ranks	Absolute ranks
Step 1	1	1
Step 2	1-2	1-2
Step 3	1-2-2	1-3-2
Step 4	1-2-2-1	2-4-3-1

In other words, there is a one-to-one correspondence between the set of paths and the set of relative ranks. By using (k,l) to represent (*k-th* interview, relative rank of the *k-th* applicant), we have to consider stopping or not the process only at points of the form (k,1), that is, when the *k-th* applicant is a candidate. To derive the probability of she (or he) being the best among all n applicants we need only to count all the paths that touch (k,1) and do not touch any point of the form (j,1), with $j > k$ and to divide by the total number of paths that touch (k,1). Then the problem of getting the probability of success is totally reduced to counting paths.

Number of paths that touch *(k,*1*)*: Black dots in Figure 3-a are all allowed, so that this number is $(k-1)!(k+1)(k+2)...n = \frac{n!}{k}$.

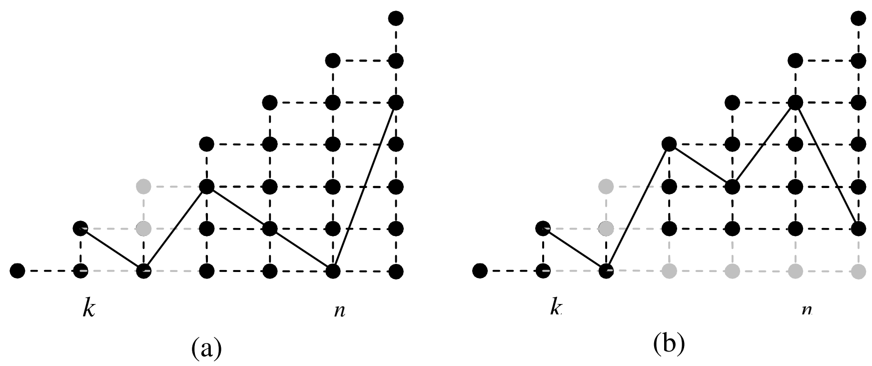

Figure 3 - Number of paths that touch (k,1). (a) all dots allowed. (b) paths do not return to the abscissa axis.

Number of paths that touch (k,1) and do not return to the abscissa axis: After column k (Figure 3-b) the number of possibilities is reduced by one at each column, so that the number of paths is $(k-1)!k(k+1)...(n-1) = (n-1)!$.

Based on the above elementary reasoning we may state the following combinatorial result:

Proposition: Given n objects, that may be ordered somehow from the best to the worse without ties, if an object is the best among k randomly chosen ones, the probability of it $\frac{(n-1)!}{n!/k} = \frac{k}{n}$ being the best among all of them is.

4 Looking for the optimal strategy

It is known (Landim – p. 41) that the optimal strategy is in the set of those that interview k applicants and then choose the next with relative rank 1. Then the point is to determine the value of k that maximizes the probability of this choice (of the next candidate) being the choice of the best applicant. Counting paths is again helpful (Figure 4).

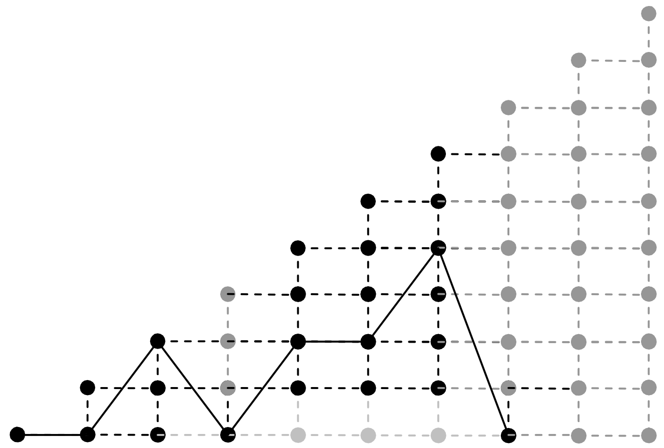

Figure 4 - Path touching abscissas axis at (k,1) and returning at (k+i,1).

A path that touches (k,1) can return to the abscissas axis at point $(k+i,1)$ with $0 < i \leq n-k$. The probability of a path that touched (k,1) to return for the first time to the abscissas axis in $(k+i,1)$ will be denoted by $p_{k,\,i}$.

Number of paths that touch (k,1) and return for the first time to the abscissas axis in (k+i, 1): According to figure VI the possibilities are reduced by one because the path can't touch the axis in any of the steps k+1, k+2, ..., k+i-1. The number of such paths is given by $(k-1)!k(k+1)\ldots(k+i-2)(k+i+1)\ldots n$, so that, for $0 < i < n-k$

$$p_{k,\,i} = \frac{(k-1)!k(k+1)\ldots(k+i-2)(k+i+1)\ldots n}{n!/k} = \frac{k}{(k+i-1)(k+i)}$$

and for $i = n-k$

$$p_{k,\,n-k} = \frac{(k-1)!k(k+1)\ldots(n-2)n}{n!/k} = \frac{k}{(n-1)}$$

Now, if we face a candidate at *k-th* interview, we may:

Decide to stop – in this case the probability of success is the number of paths that pass by $(k,1)$ and never touches again the abscissas axis devided by the total number os paths by $(k,1)$, that is

$$\frac{(k-1)!k(k+1)\dots(n-1)}{n!/k} = \frac{k}{n}.$$

Decide to continue - in this case we must know the probability of success hereafter. Since the probability of the path come to $(k+j,1)$ for first time after *k* is $p_{k,k+j}$, it follows that the probability of $k+j$ being a candidate and being the best is $\frac{k+j}{n}p_{k,k+j}$. Since $j=1,\dots,n-k$ are all available, the probability we need is the weighted mean given by

$$\sum_{i=1}^{n-k} p_{k,k+i}\frac{k+i}{n}.$$

And the decision may be taken by comparing these two numbers in the following way:

$$\max\left\{\frac{k}{n},\sum_{i=1}^{n-k} p_{k,k+i}\frac{k+i}{n}\right\} = \begin{cases}\frac{k}{n} \Rightarrow stop \\ \sum_{i=1}^{n-k} p_{k,k+i}\frac{k+i}{n} \Rightarrow continue\end{cases}$$

where max means the maximum. Since

$$\max\left\{\frac{k}{n},\sum_{i=1}^{n-k} p_{k,k+i}\frac{k+i}{n}\right\} = \frac{1}{n}\max\left\{k,\sum_{i=1}^{n-k} p_{k,k+i}(k+i)\right\} =$$

$$= \frac{1}{n}\max\left\{k,\sum_{i=1}^{n-k}\frac{k}{(k+i-1)(k+i)}(k+i)\right\} =$$

$$= \frac{k}{n}\max\left\{1,\sum_{i=1}^{n-k}\frac{1}{(k+i-1)}\right\},$$

the process should continue whenever $\sum_{i=1}^{n-k}\frac{1}{(k+i-1)} > 1$.

Then, according to this strategy, we must find the number k_n such that

$$\frac{1}{k_n+1}+\frac{1}{k_n+2}+\dots+\frac{1}{n-1} < 1 < \frac{1}{k_n}+\frac{1}{k_n+1}+\dots+\frac{1}{n-1}$$

and choose the next candidate.

The strategy consists then in rejecting the first k_n applicants and to accept the first better than all the previous. Example: For $n = 8$ we have $k_8 = 3$. Indeed,

$$\frac{1}{4}+\frac{1}{5}+\frac{1}{6}+\frac{1}{7}=\frac{319}{420}<1<\frac{153}{140}=\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\frac{1}{6}+\frac{1}{7}$$

5 Probability of success of the strategy

The probability of selecting the best applicant by using this strategy can also be obtained by counting paths. A path will represent a success for the strategy if, after k_n, it returns only once to the abscissas axis (Figure 5).

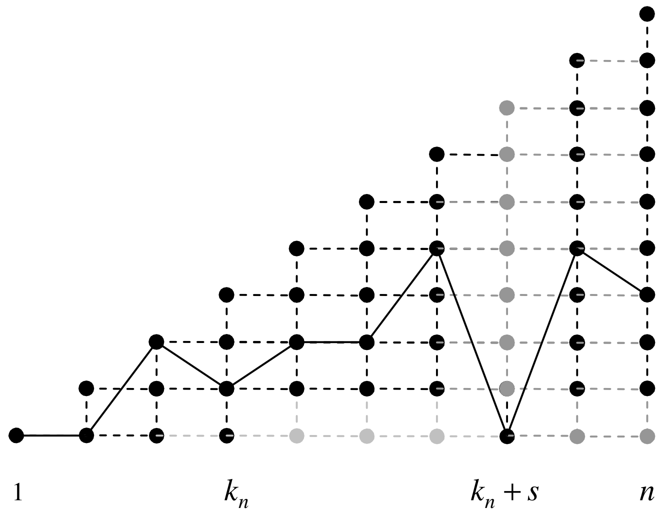

Figure 5 - A success path: only one return to the abscissas axis after k n.

For $1 \le s \le n-k_n$, the number of paths that touch the abscissa axis only once, at some point $(k_n+s,1)$, is given by

$$\sum_{s=1}^{n-k_n} k_n!\, k_n.(k_n+1)...(k_n+s-2).(k_n+s)...(n-1)=$$

$$=k_n \sum_{s=1}^{n-k_n} k_n!\,(k_n+1)...(k_n+s-2).(k_n+s-1).(k_n+s)...(n-1)\frac{1}{(k_n+s-1)}$$

$$=k_n \sum_{s=1}^{n-k_n} \frac{(n-1)!}{k_n+s-1} = k_n(n-1)! \sum_{s=1}^{n-k_n} \frac{1}{k_n+s-1}$$

and the probability of success is

$$\mathrm{P}[\text{success}] = \frac{k_n (n-1)! \sum_{s=1}^{n-k_n} \frac{1}{k_n + s - 1}}{n!} = \frac{k_n}{n}\left(\frac{1}{k_n} + \frac{1}{k_n+1} + \ldots + \frac{1}{n-1}\right) = \frac{k_n}{n}\sum_{k=k_n}^{n-1}\frac{1}{k}.$$

For $n = 8$ the probability of success is

$$\mathrm{P}[\text{success}] = \frac{k_n}{n}\sum_{k=k_n}^{n-1}\frac{1}{k} = \frac{3}{8}\sum_{k=3}^{7}\frac{1}{k} = \frac{3}{8}\left(\frac{1}{3} + \ldots + \frac{1}{7}\right) = 40{,}98\ \%$$

When n is large enough this probability is approximately $1/e$. This can be seen as follows. Observing the Figure 6 we write

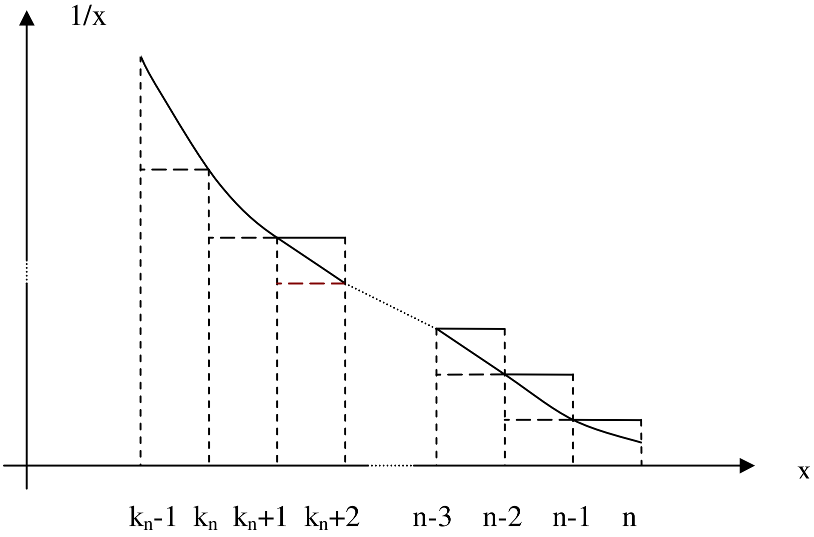

Figure 6 - Approximation of $\sum 1/k$.

$$\int_{k_n-1}^{n-1} \frac{1}{x}dx > \frac{1}{k_n} + \frac{1}{k_n+1} + \ldots + \frac{1}{n-1} \qquad \textit{(horizontal broken lines)}$$

$$\int_{k_n+1}^{n} \frac{1}{x}dx < \frac{1}{k_n+1} + \frac{1}{k_n+2} + \ldots + \frac{1}{n-1} \qquad \textit{(horizontal continuous lines)}$$

Recalling that k_n is such that

$$\frac{1}{k_n+1} + \frac{1}{k_n+2} + \ldots + \frac{1}{n-1} < 1 < \frac{1}{k_n} + \frac{1}{k_n+1} + \ldots + \frac{1}{n-1}$$

it follows that

$$\int_{k_n+1}^{n}\frac{1}{x}dx < \frac{1}{k_n+1}+\frac{1}{k_n+2}+...+\frac{1}{n-1} < 1 < \frac{1}{k_n}+\frac{1}{k_n+1}+...+\frac{1}{n-1} < \int_{k_n-1}^{n-1}\frac{1}{x}dx$$

$$Ln\left(\frac{n}{k_n+1}\right) < 1 < Ln\left(\frac{n-1}{k_n-1}\right)$$

$$\frac{n}{k_n+1} < e < \frac{n-1}{k_n-1}$$

$$\frac{k_n+1}{n} > \frac{1}{e} > \frac{k_n-1}{n-1} > \frac{k_n-1}{n}$$

$$\frac{k_n}{n}+\frac{1}{n} > \frac{1}{e} > \frac{k_n}{n}-\frac{1}{n}.$$

Then we get $\frac{k_n}{n} \cong \frac{1}{e}$ and

$$\text{P[success]} = \frac{k_n}{n}\sum_{k=k_n}^{n-1}\frac{1}{k} \cong \frac{k_n}{n}\int_{k_n}^{n}\frac{1}{x}dx = \frac{k_n}{n}\text{Ln}\left(\frac{n}{k_n}\right) \cong \frac{1}{e} \cong 36,78\ \%$$

To show how robust is the optimal strategy a simulation was done, using R. For $n = 40$ the probability of success is 38.42%. In 10.000 repetitions, the best applicant has been chosen 3.776 times. Figure 7 shows the frequency each rank has been chosen:

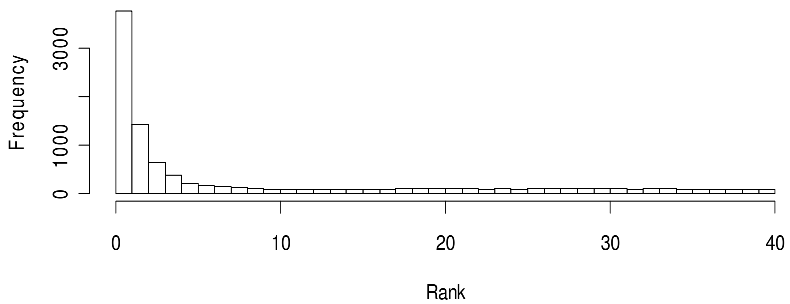

Figure 7 - Frequency for each possible rank when n=40.

6 Expected rank

As the applicants are presented in random order, relative and final ranks are random variables. Let E[ ] denotes the expected value of a random variable. Using the optimal strategy, some applicant, from the $(k_n+1)-th$ to the $n-th$, will be chosen. As the strategy does not guarantee that the best will be chosen, it is natural to ask for the expected final (absolute) rank of the selected one. It is known (Ferguson - Optimal stopping and applications) that, if an applicant has rank s in the first j applicants then her (or his) expected rank in n applicants is given by $\frac{n+1}{j+1}s$.

The expected final rank of a chosen candidate is:

$$\text{E[final rank]} = \sum_{i=1}^{n-k_n} \text{P}[(k_n+i)\text{-th applicant be a candidate}].\text{E[rank of this candidate]}\cdot$$

For $1 \le i \le n-k_n-1$, the probability of chosen the (k_n+i)-th candidate is

$$p_{k_n,\,i} = \frac{k_n}{(k_n+i-1)(k_n+i)}$$

For $i = n-k_n$, we have to count the number of paths that, after the step k_n, do not return to the abscissas axis in the steps $k_n+1,\ldots,n-1$ to calculate the probability

$$\text{P[choose the } n\text{-th candidate]}=\frac{1}{n!}k_n!k_n(k_n+1)\ldots(n-2)n = \frac{k_n}{n-1}.$$

For $n=40$, $k_n=15$ and this probability is 38.46 %. To ilustrate, the same 10.000 simulations showed that the last applicant was chosen 3898 times. Figure 8 shows how many times the *i-th* applicant $(i=1,\ldots,40)$ was chosen in 10.000 simulations:

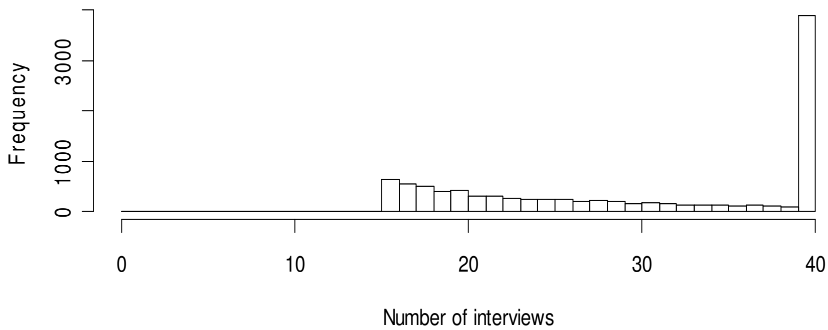

Figure 8 - Frequency of the number of interviews in 10,000 simulations.

Case the last applicant is chosen, what is her (or his) expected rank? This seems a not easy question, because it depends on the strategy. Nevertheless, the approach using paths brings a trivial solution. Each path that do not touch the X-axis from k_n+1 to

$n-1$ corresponds to a sequence that the strategy chooses the last applicant. The symmetry of the problem shows clearly that the number of paths finishing in any final rank is the same. More specifically, given a sequence of absolute ranks which results in the choice of the last applicant with rank s_1, it is possible to construct another sequence of absolute ranks, which results in the choice of the last applicant with a different rank s_2 in such a way that both sequences lead to the same sequence of relative ranks, except, of course, for the last position. That is, both sequences correspond to the same path in the graph. To illustrate, consider de absolute ranks sequence $[7 \quad 1 \quad 5 \quad 3 \quad 6 \quad 2 \quad 4]$ which corresponds to relative ranks $[1 \quad 1 \quad 2 \quad 2 \quad 4 \quad 2 \quad 4]$. To obtain another sequence of absolute ranks with, say, 6 at the final position one have to:

1. subtract one in all position with rank greater or equal to four and clean the last position: $[6 \quad 1 \quad 4 \quad 3 \quad 5 \quad 2 \quad ...]$
2. put 6 in the last position and add one in all position with rank greater or equal to six: $[7 \quad 1 \quad 4 \quad 3 \quad 5 \quad 2 \quad 6]$

Reader may check that this last sequence has the same relative ranks $[1 \quad 1 \quad 2 \quad 2 \quad 4 \quad 2 \quad 4]$. This procedure works for any two different final ranks, which means that there is a bijection between these sequences (a routine to do this in general is in the appendix). So, the expected rank, in this case, is $\frac{n}{2}$. Figure 9 shows the uniformity of the final rank frequency, when the strategy chooses the last candidate, with 10.000 repetitions:

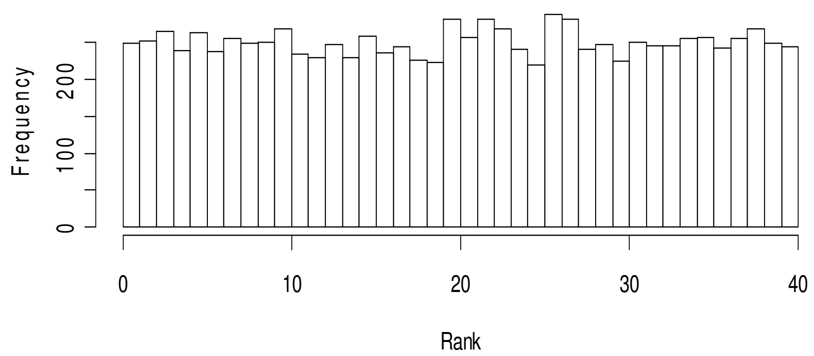

Figure 9 - Final rank frequency when last candidate is chosen.

Putting the above results together,

$$\text{E[final rank]} = \sum_{i=1}^{n-k_n-1} \frac{k_n}{(k_n+i-1)(k_n+i)}\left(\frac{n+1}{k_n+i+1}\right) + \frac{k_n}{n-1}\left(\frac{n}{2}\right)$$

$$= (n+1)k_n \sum_{j=k_n+1}^{n-1} \frac{1}{(j-1)j(j+1)} + \frac{1}{2}\frac{n}{n-1}k_n .$$

For n relatively large, using a construction similar to Figure 6,

$$\sum_{j=k_n+1}^{n-1} \frac{1}{(j-1)j(j+1)} \cong \sum_{j=k_n+1}^{n-1} \frac{1}{j^3} \cong \int_{k_n+1}^{n} \frac{1}{x^3}\, dx = \frac{1}{2}\left(\frac{1}{(k_n+1)^2} - \frac{1}{n^2}\right)$$

$$\text{E[final rank]} = (n+1)k_n \sum_{j=k_n+1}^{n-1} \frac{1}{(j-1)j^2} + \frac{1}{2}\frac{n}{n-1}k_n$$

$$\cong (n+1)k_n \frac{1}{2}\left(\frac{1}{(k_n+1)^2} - \frac{1}{n^2}\right) + \frac{1}{2}k_n$$

$$\cong \frac{(n+1)k_n}{2(k_n+1)^2} - \frac{(n+1)k_n}{2n^2} + \frac{1}{2}k_n$$

$$\cong \frac{1}{2}\frac{n}{k_n} - \frac{1}{2}\frac{k_n}{n} + \frac{1}{2}k_n \cong \frac{1}{2}\left(e - \frac{1}{e} + k_n\right) \cong \frac{1}{2}k_n \cong \frac{n}{2\,e}$$

For $n = 40$ this number is $\frac{40}{2\,e} = 7.35$. By simulation the number obtained was 8.99.

Authors could not find any reference to this result in the vast literature on the subject.

It is also possible, using the same argument of counting, to obtain the expected time for running the optimal strategy. Considering each interview as a unit of time, the expected time of execution of the strategy, T, is given by:

$$\mathrm{E}(T) = \sum_{i=1}^{n-k_n-1} (k_n+i)\ \text{P[choose the } (k_n+i)\text{-th applicant]+}$$

$$+n\ \text{P[choose the } n\text{-th applicant]}$$

$$\mathrm{E}(T) = \sum_{i=1}^{n-k_n-1} (k_n+i)\ \frac{k_n}{(k_n+i-1)(k_n+i)} + n\frac{k_n}{n-1}$$

$$= k_n \sum_{i=1}^{n-k_n-1} \frac{1}{(k_n+i-1)} + \frac{n}{n-1}k_n\ .$$

For large n, E[T] is approximately:

$$\mathrm{E}[T] \cong \frac{n}{e}\left(\int_{k_n+1}^{n-1} \frac{1}{x}dx + 1\right) = \frac{n}{e}\left(\mathrm{Ln}\frac{n-1}{k_n - 1} + 1\right) \cong$$

$$\cong \frac{n}{e}\left(\mathrm{Ln}\frac{n}{k_n} + 1\right) \cong$$

$$\cong \frac{n}{e}\left(\mathrm{Ln}(e) + 1\right) = 2\ \frac{n}{e}$$

This result was obtained by Quine (1996). For $n = 40$ this number is $\frac{80}{e} = 29.43$ and the simulation showed 30.25.

CHAVES, L. M.; SOUZA, D. J. de. Uma abordagem elementar para o problema da secretária. *Rev. Bras. Biom.*, São Paulo, v.27, n.1, p.7-21, 2009.

- RESUMO: Usando uma interpretação do fatorial baseada na contagem de caminhos em um diagrama, uma abordagem elementar para o tradicional problema da secretária é desenvolvida.

- PALAVRAS-CHAVE: Problema da secretária; caminhos, estratégia ótima; posto esperado.

References

LANDIM, C. Otimização randômica In: COLÓQUIO BRASILEIRO DE MATEMÁTICA, 18., 1991. Rio de Janeiro.

FELLER, W. *An introduction to probability theory and its applications*, New York: Wiley International Edition, 1968. v.1.

FERGUSON, T. S. Who solved the secretary problem? *Stat. Sci.*, Hayvard, v.4, n.3, p.282-289, 1989.

FERGUSON, T. S. *Optimal stopping and applications*. Disponível em: <http://www.math.ucla.edu/~tom/Stopping/Contents.html>. Acesso em: nov.2008.

FREEMAN, P. R. The secretary problem and its extensions: review, *Int. Stat. Revi.*, Warburg, v.51, n.2, p.189-206, 1983.

R Development Core Team (2008). *R: A language and environment for statistical computing. R Foundation for Statistical Computing*, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

QUINE, M. P., LAW, J. S. Exact results for a secretary problem, *J. Appl. Prob*. Sheffield, v.33, p.630-639, 1996.

Recebido em 08.04.2008.

Aprovado após revisão 20.03.2009.

Appendix

The routines written in R:

```
#       Function to choose the secretary in a random generated sequence of absolute ranks
#           The input is the number of applicants n.
#           The output is a vector (x1,x2,x3,x4) where x1 is the number of interviews up
#                       to decision, x2 the rank of chosen candidate, x3 the number of  dummy
#                       interviews and x4 the generated sequence of absolute ranks
#

secr <- function(n){
            if (n<3) cat("the number of applicants must be greater than 2")
            s_inv_n = 1/c(1:n)
            #           determining kn - the number of dummy interviews

            kn = 0
            flag = 0
            while (flag == 0){
                        kn = kn+1
                        Si = sum(s_inv_n[(kn+1):(n-1)])
                        Ss = sum(s_inv_n[kn:(n-1)])
                        if (Si < 1){
                                    if (Ss >1) flag = 1
                        }

            }
            #           end
            fila = order(runif(n))
            flag = 0
            k = kn+1
            while (flag == 0 & k < n ){
                        ref = min(fila[1:(k-1)])
                        if (fila[k] < ref) flag = 1
                        k = k +1
            }
            k = k-1
            if (flag == 1) sec = fila[k]
            if (flag == 0){
                        k = n
                        sec = fila[n]
            }
            return(c(k,sec,kn,fila))
}

#           Procedure relative rank
```

```
#       The input: ar - a permutation in {1,2,...,k} = the absolute ranks
#       The output: rr - the corresponding sequence of relative ranks

relrank <- function(ar){
        k = length(ar)
        rr = 1:k
        for (i in 2:k){
                ref = ar[i]
                so = sort(ar[1:i])
                p = 0
                for (j in 1:i){
                        if (so[j] == ref) p = j
                }
                rr[i] = p
        }
        return(rr)
}

#       procedure two sequences with same relative rank
#       The inputs:     n - the size of the sequence of absolute ranks to be generated
#                       s1 - the n-th rank at first sequence
#                       s2 - the n-th rank at second sequence
#       The output:     a matrix 4Xn
#                               line 1 - first sequence of absolute ranks
#                               line 2 - relative ranks for line 1
#                               line 3 - second sequence of absolute ranks
#                               line 4 - relative ranks for line 3

samerank <- function(n,s1,s2){
        flag = 0
        ss = matrix(0,4,n) #    the output matrix
        while (flag == 0){ #    start selecting first sequence
                sec = secr(n)
                if (sec[1] == n & sec[2] == s1) flag = 1
        }
        ss1 = sec[4:(n+3)]              #       chosen first sequence
        ss[1,(1:n)] = ss1
        ss[2,1:n] = relrank(ss1)#   relative rank for first sequence
                #       start modifying first sequence to obtain second
        for (i in 1:(n-1)) if (ss1[i] >= s1) ss1[i] = ss1[i]-1
        for (i in 1:(n-1)) if (ss1[i] >= s2) ss1[i] = ss1[i]+1
        ss1[n] = s2
        ss[3,(1:n)] = ss1               #       second sequence
        ss[4,(1:n)] = relrank(ss1)
        return(ss)
}
```