Stratified Importance Sampling

Don Knuth, 31 January 2020
23 Feb 1975    $\Rightarrow$    Lehmer = 70
10 Jan 2018    $\Rightarrow$    Knuth = 80
31 Jan 2020    $\Rightarrow$    Diaconis = 75
Mathematics
of
Computation

Volume 29 · Number 129

Special Issue ~ Dedicated to

DERRICK HENRY LEHMER

January 1975

Published by
American Mathematical Society
Providence · Rhode Island
THE EDITORS of Mathematics of Computation are pleased to
dedicate this issue to Derrick Henry Lehmer on the occasion of his
seventieth birthday, February 23, 1975.

Lehmer played a role in the founding of MTAC and was an active
member of the executive committee that produced the first issue in
January, 1943, under the editorship of R. C. Archibald. In the 1946—
1949 volumes, Lehmer joined Archibald as an editor, while in the
1950—1954 volumes, Lehmer was the chairman of an enlarged
group of editors.

In an effort to keep this dedication a surprise and to keep the
issue modest in size, the editors did not inform all of the people who
would have liked to contribute articles to celebrate Lehmer’s birthday.
In fact, the editors have postponed the publication of a paper jointly
authored by the dedicatee. To those who do not appear here, we
apologize.

We must make mention of a special one, who was not informed
about this issue—Emma Lehmer. She has also been associated with
this journal from its beginning and continues to assist us to this day.
We are in her debt too!

The editors are also pleased to acknowledge that this issue was
made possible through the generous support that was received from
friends of the Lehmers.

Eugene Isaacson
for the editors
Estimating the Efficiency of Backtrack Programs*

By Donald E. Knuth

To Derrick H. Lehmer on his 70th birthday, February 23, 1975

Abstract. One of the chief difficulties associated with the so-called backtracking technique for combinatorial problems has been our inability to predict the efficiency of a given algorithm, or to compare the efficiencies of different approaches, without actually writing and running the programs. This paper presents a simple method which produces reasonable estimates for most applications, requiring only a modest amount of hand calculation. The method should prove to be of considerable utility in connection with D. H. Lehmer's branch-and-bound approach to combinatorial optimization.

The majority of all combinatorial computing applications can apparently be handled only by what amounts to an exhaustive search through all possibilities. Such searches can readily be performed by using a well-known "depth-first" procedure which R. J. Walker [21] has aptly called backtracking. (See Lehmer [16], Golomb and Baumert [6], and Wells [22] for general discussions of this technique, together with numerous interesting examples.)

Sometimes a backtrack program will run to completion in less than a second, while other applications seem to go on forever. The author once waited all night for the output from such a program, only to discover that the answers would not be forthcoming for about $10^6$ centuries. A "slight increase" in one of the parameters of a backtrack routine might slow down the total running time by a factor of a thousand; conversely, a "minor improvement" to the algorithm might cause a hundredfold improvement in speed; and a sophisticated "major improvement" might actually make the program ten times slower. These great discrepancies in execution time are characteristic of backtrack programs, yet it is usually not obvious what will happen until the algorithm has been coded and run on a machine.

Faced with these uncertainties, the author worked out a simple estimation procedure in 1962, designed to predict backtrack behavior in any given situation. This procedure was mentioned briefly in a survey article a few years later [8]; and during subsequent years, extensive computer experimentation has confirmed its utility. Several
move solutions of Figure 5, stating that "il est probablement impossible de dénombrer la quantité de ces tours; . . . vraisemblablement, on ne peut effectuer plus de 35 coups." Later [3, p. 20], [4, p. 35] he stated without proof that 35 is maximum.

![Diagram](image)

**Figure 5. Uncrossed knight’s tours**

The backtrack method provides a way to test his assertion; we may begin the tour in any of 10 essentially different squares, then continue by making knight’s moves that do not cross previous ones, until reaching an impasse. But backtrack trees that extend across 30 levels or more can be extremely large; even if we assume an average of only 3 consistent choices at every stage, out of at most 7 possible knight moves to new squares, we are faced with a tree of about $3^{30} = 205,891,132,094,649$ nodes, and we would never finish. Actually $3^{20} = 3,486,784,401$ is nearer the upper limit of feasibility, since it is not at all simple to test whether or not one move crosses another. It is certainly not clear *a priori* that an exhaustive backtrack search is economically feasible.

The simple procedure of Section 3 was therefore used to estimate the number of nodes in the tree, using $c(t) = 1$ for all $t$. Here are the estimated tree sizes found in the first ten independent experiments:

| 1571717091 | 209749511 |
| 315291281 | 58736818301 |
| 8231 | 311 |
| 1793651 | 259271 |
| 59761491 | 6071489081 |

The mean value is 6,696,688,822. The next sequence of ten experiments gave the estimates

| 567911 | 238413491 |
| 111 | 6697691 |
| 569585831 | 5848873631 |
| 111 | 161 |
| 411 | 140296511 |

for an average of only 680,443,586, although the four extremely low estimates make
### Table 1. Estimates after 1000 random walks

<table>
<thead>
<tr>
<th>$k$</th>
<th>Estimate, $N'_k$</th>
<th>True value, $N_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>10.0</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>42.8</td>
<td>42</td>
</tr>
<tr>
<td>3</td>
<td>255.0</td>
<td>251</td>
</tr>
<tr>
<td>4</td>
<td>991.4</td>
<td>968</td>
</tr>
<tr>
<td>5</td>
<td>4352.2</td>
<td>4215</td>
</tr>
<tr>
<td>6</td>
<td>16014.4</td>
<td>15646</td>
</tr>
<tr>
<td>7</td>
<td>59948.8</td>
<td>56435</td>
</tr>
<tr>
<td>8</td>
<td>190528.7</td>
<td>182520</td>
</tr>
<tr>
<td>9</td>
<td>580450.8</td>
<td>574555</td>
</tr>
<tr>
<td>10</td>
<td>1652568.7</td>
<td>1606422</td>
</tr>
<tr>
<td>11</td>
<td>4424403.9</td>
<td>4376153</td>
</tr>
<tr>
<td>12</td>
<td>9897781.4</td>
<td>10396490</td>
</tr>
<tr>
<td>13</td>
<td>22047261.5</td>
<td>23978392</td>
</tr>
<tr>
<td>14</td>
<td>44392865.5</td>
<td>47667686</td>
</tr>
<tr>
<td>15</td>
<td>92464977.5</td>
<td>91377173</td>
</tr>
<tr>
<td>16</td>
<td>145815116.2</td>
<td>150084206</td>
</tr>
<tr>
<td>17</td>
<td>238608697.6</td>
<td>235901901</td>
</tr>
<tr>
<td>18</td>
<td>253061952.9</td>
<td>315123658</td>
</tr>
<tr>
<td>19</td>
<td>355460520.9</td>
<td>399772215</td>
</tr>
<tr>
<td>20</td>
<td>348542887.6</td>
<td>427209856</td>
</tr>
<tr>
<td>21</td>
<td>328849873.9</td>
<td>429189112</td>
</tr>
<tr>
<td>22</td>
<td>340682204.1</td>
<td>358868304</td>
</tr>
<tr>
<td>23</td>
<td>429508177.9</td>
<td>278831518</td>
</tr>
<tr>
<td>24</td>
<td>318416025.6</td>
<td>177916192</td>
</tr>
<tr>
<td>25</td>
<td>38610432.0</td>
<td>103894319</td>
</tr>
<tr>
<td>26</td>
<td>75769344.0</td>
<td>49302574</td>
</tr>
<tr>
<td>27</td>
<td>74317824.0</td>
<td>21049968</td>
</tr>
<tr>
<td>28</td>
<td>0.0</td>
<td>7153880</td>
</tr>
<tr>
<td>29</td>
<td>0.0</td>
<td>2129212</td>
</tr>
<tr>
<td>30</td>
<td>0.0</td>
<td>522186</td>
</tr>
<tr>
<td>31</td>
<td>0.0</td>
<td>109254</td>
</tr>
<tr>
<td>32</td>
<td>0.0</td>
<td>18862</td>
</tr>
<tr>
<td>33</td>
<td>0.0</td>
<td>2710</td>
</tr>
<tr>
<td>34</td>
<td>0.0</td>
<td>346</td>
</tr>
<tr>
<td>35</td>
<td>0.0</td>
<td>50</td>
</tr>
<tr>
<td>36</td>
<td>0.0</td>
<td>8</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>3123375511.1</strong></td>
<td><strong>3137317290</strong></td>
</tr>
</tbody>
</table>
Sequential importance sampling for estimating the number of perfect matchings in bipartite graphs:

An ongoing conversation with Laci

Persi Diaconis*
Departments of Mathematics and Statistics
Stanford University

Abstract
Sequential importance sampling offers an alternative way to approximately evaluate the permanent. It is a stochastic algorithm which seems to work in practice but has eluded analysis. This paper offers examples where the analysis can be carried out and the first general bounds for the sample size required. This uses a novel importance sampling proof of Brégman's inequality due to Lovász.

1 Introduction

Let $G = (V, W, E)$ be a bipartite graph with $|V| = |W| = n$ and $E$ a set of edges from $V$ to $W$. Let $\mathcal{M}$ be the set of perfect matchings. Assume throughout that $\mathcal{M}$ is non-empty. There is a large literature on computing and approximating $M = |\mathcal{M}|$. See [4] for background and applications in statistics. The magisterial [12] covers every aspect of matching theory. This paper studies an importance sampling algorithm for Monte Carlo approximation of $M$.
The number of perfect matchings is the Fibonacci number $F_{n+1}$. Indeed, 1 can only be matched to 1' or 2'. If it is matched to 1' the deleted graph is $A_{n-1}$. If it is matched to 2', then 2 must be matched to 1' and the deleted graph is $A_{n-2}$. These matchings are illustrated in Figure 1. Thus there are five perfect matchings consistent with $A_4$; these are shown in Table 1 along with their associated probabilities if the vertices are tried in order 1, 2, 3, 4 for $P_1(\pi)$ and 2, 3, 4, 1 for $P_2(\pi)$. For larger $n$ the possible matching probabilities can be quite different. In what follows the vertex order $1, 2, \ldots, n$ is studied.

Table 1: Perfect matchings

<table>
<thead>
<tr>
<th>$\pi$</th>
<th>1234</th>
<th>2134</th>
<th>1324</th>
<th>1243</th>
<th>2143</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1(\pi)$</td>
<td>1/8</td>
<td>1/4</td>
<td>1/4</td>
<td>1/8</td>
<td>1/4</td>
</tr>
<tr>
<td>$P_2(\pi)$</td>
<td>1/6</td>
<td>1/6</td>
<td>1/3</td>
<td>1/6</td>
<td>1/6</td>
</tr>
<tr>
<td>$P_3(\pi)$</td>
<td>1/6</td>
<td>1/4</td>
<td>1/6</td>
<td>1/6</td>
<td>1/4</td>
</tr>
</tbody>
</table>

Figure 2 shows a histogram of 1000 $T_i$ when $n = 12$; then $F_{n+1} = 89$. The mean of 86.72 is reasonable but the minimum of 24, maximum of 288, and standard deviation of 45.3 give an indication of large variability.
Left to right:
Left to right:

Easy to show:

\[2F_{l+1} + F_l - 1 \text{ nodes on level } l \geq 0;\]
\[2F_{n+3} - n - 1 \text{ nodes total.}\]
“Random” order:
"Random" order:

Theorem (Ira Gessel, Philippe Jacquet, January 2020):

Average tree size, over all $n!$ orders,
is $\sim c\sqrt{n} \cdot \alpha^n$,
where $\alpha \approx 1.71995$ is the real root of $11x^3 - 18x^2 - x - 1 = 0$. 
HEURISTIC SAMPLING: A METHOD FOR PREDICTING THE PERFORMANCE OF TREE SEARCHING PROGRAMS

PANG C. CHEN†

Abstract. Determining the feasibility of a particular search program is important in practical situations, especially when the computation involved can easily require days, or even years. To help make such predictions, a simple procedure based on a stratified sampling approach is presented. This new method, which is called heuristic sampling, is a generalization of Knuth’s original algorithm for estimating the efficiency of backtrack programs. With the aid of simple heuristics, this method can produce significantly more accurate cost estimates for commonly used tree search algorithms such as depth-first, breadth-first, best-first, and iterative-deepening.

Key words. heuristics, stratified sampling, search tree, feasibility testing, cost estimation, Monte Carlo method, analysis of algorithms

AMS(MOS) subject classifications. 68Q25, 65C05

1. Introduction. Tree searching [8] is a general, easily implemented problem-solving technique. Unfortunately, the efficiency of tree searching programs is usually difficult to analyze, even at a rudimentary level. Without analytic cost information, the typical course is to let the computer run until it either finishes the job or exhausts our patience. Switching to a more computational sampling approach provides a less haphazard alternative: We gain accuracy in understanding particular search programs and thus design better ones.

The sampling method that we will discuss generalizes an algorithm of Knuth [5] for estimating properties of a backtrack tree. Starting at the root, Knuth’s algorithm extends a partial path by expanding the end node and picking a child according to a uniform distribution. It then forms an unbiased estimate of the tree property using the branching degrees along the randomly selected path. According to Knuth, this simple estimation procedure worked consistently well in his experiments. But as a refinement, Knuth also suggested the technique of importance sampling [2] in which the child is selected according to a weighted distribution, with the weight of each child being an estimate of the property of the corresponding subtree.
### Table 1

**Estimated tree profiles.**

<table>
<thead>
<tr>
<th>Level</th>
<th>$h_0$</th>
<th>$h_1$</th>
<th>$h_2$</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>43</td>
<td>42</td>
<td>34</td>
<td>42</td>
</tr>
<tr>
<td>3</td>
<td>257</td>
<td>196</td>
<td>204</td>
<td>251</td>
</tr>
<tr>
<td>4</td>
<td>1005</td>
<td>918</td>
<td>864</td>
<td>968</td>
</tr>
<tr>
<td>5</td>
<td>4267</td>
<td>3734</td>
<td>3604</td>
<td>4215</td>
</tr>
<tr>
<td>6</td>
<td>15684</td>
<td>14026</td>
<td>13724</td>
<td>15646</td>
</tr>
<tr>
<td>7</td>
<td>57443</td>
<td>46873</td>
<td>51326</td>
<td>56435</td>
</tr>
<tr>
<td>8</td>
<td>184017</td>
<td>126007</td>
<td>174680</td>
<td>182520</td>
</tr>
<tr>
<td>9</td>
<td>583197</td>
<td>445188</td>
<td>611950</td>
<td>574555</td>
</tr>
<tr>
<td>10</td>
<td>1608183</td>
<td>1613621</td>
<td>1832792</td>
<td>1606422</td>
</tr>
<tr>
<td>11</td>
<td>4371678</td>
<td>4679457</td>
<td>5106228</td>
<td>4376153</td>
</tr>
<tr>
<td>12</td>
<td>9920481</td>
<td>16793389</td>
<td>11767976</td>
<td>10396422</td>
</tr>
<tr>
<td>13</td>
<td>22507886</td>
<td>37750158</td>
<td>25507044</td>
<td>23978392</td>
</tr>
<tr>
<td>14</td>
<td>47054083</td>
<td>31161416</td>
<td>47574616</td>
<td>47667686</td>
</tr>
<tr>
<td>15</td>
<td>95846027</td>
<td>64365367</td>
<td>83428348</td>
<td>91377173</td>
</tr>
<tr>
<td>16</td>
<td>156085563</td>
<td>106268997</td>
<td>130208416</td>
<td>150084206</td>
</tr>
<tr>
<td>17</td>
<td>256650972</td>
<td>205655280</td>
<td>192462996</td>
<td>235901901</td>
</tr>
<tr>
<td>18</td>
<td>344261889</td>
<td>306859328</td>
<td>252526926</td>
<td>315123658</td>
</tr>
<tr>
<td>19</td>
<td>451703678</td>
<td>382647057</td>
<td>280613374</td>
<td>399772215</td>
</tr>
<tr>
<td>20</td>
<td>324806815</td>
<td>372852597</td>
<td>277536920</td>
<td>427209856</td>
</tr>
<tr>
<td>21</td>
<td>253992102</td>
<td>426672714</td>
<td>272541294</td>
<td>429189112</td>
</tr>
<tr>
<td>22</td>
<td>237855139</td>
<td>239446386</td>
<td>288235830</td>
<td>358863034</td>
</tr>
<tr>
<td>23</td>
<td>106973568</td>
<td>352833124</td>
<td>226878632</td>
<td>278831518</td>
</tr>
<tr>
<td>24</td>
<td>80018150</td>
<td>195532658</td>
<td>138579650</td>
<td>177916192</td>
</tr>
<tr>
<td>25</td>
<td>28304640</td>
<td>57001796</td>
<td>87525874</td>
<td>103894319</td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td>72444273</td>
<td>40045814</td>
<td>49302574</td>
</tr>
<tr>
<td>27</td>
<td>0</td>
<td>19732265</td>
<td>16732876</td>
<td>21049968</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
<td>151248453</td>
<td>9984256</td>
<td>7153880</td>
</tr>
<tr>
<td>29</td>
<td>0</td>
<td>0</td>
<td>2985372</td>
<td>2129212</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>0</td>
<td>787150</td>
<td>522186</td>
</tr>
<tr>
<td>31</td>
<td>0</td>
<td>0</td>
<td>138532</td>
<td>109254</td>
</tr>
<tr>
<td>32</td>
<td>0</td>
<td>0</td>
<td>13210</td>
<td>18862</td>
</tr>
<tr>
<td>33</td>
<td>0</td>
<td>0</td>
<td>588</td>
<td>2710</td>
</tr>
<tr>
<td>34</td>
<td>0</td>
<td>0</td>
<td>588</td>
<td>346</td>
</tr>
<tr>
<td>35</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>36</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

| total  | 2422806779 | 3046195329 | 2393871699 | 3137317290 |
| std dev | 10672087188 | 2331712470 | 878149488 | 5 |
| cost    | 11503 | 10875 | 11345 |
| trials  | 1000 | 40 | 5 |
The Pi graph (an infinite graph on the nonnegative integers):

let \( \pi \)'s binary digits be

\[
\pi_0 \pi_1 \pi_2 \pi_3 \pi_4 \pi_5 \pi_6 \pi_7 \pi_8 \pi_9 \pi_{10} \pi_{11} \pi_{12} \pi_{13} \ldots \quad =
\]

11001001000011\ldots; then

\[j \rightarrow k \text{ and } j < k \iff \pi_{j+k(k-1)/2} = 1.
\]

\[
\begin{array}{cccccccc}
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
\vdots & & & & & & & \\
\end{array}
\]
Chen subsets:

A subset $S$ of the search tree $T$
is a \textit{Chen subset} if

i) $S$ is not empty.

ii) If $s \in S$, then $\hat{s} \in S$. ($\hat{s}$ denotes the parent of $s$.)

iii) If $\hat{s} \in S$, then $s' \in S$ for some node $s'$ with $h(s) = h(s')$.

iv) If $s \in S$ and $s' \in S$ and $s \neq s'$, then $h(s) \neq h(s')$. 
a subset $S$ of the search tree $T$

is a Chen subset if

i) $S$ is not empty.

ii) If $s \in S$, then $\hat{s} \in S$. ($\hat{s}$ denotes the parent of $s$.)

iii) If $\hat{s} \in S$, then $s' \in S$ for some node $s'$ with $h(s) = h(s')$.

iv) If $s \in S$ and $s' \in S$ and $s \neq s'$, then $h(s) \neq h(s')$. 