N3078 : Levin

1) Janos Korner   At Nov I.T. conf.
Peter Gacs

2) Gábor Balovàr.

3) Geal. activities ≷ ~~USSR~~ $(1-p)^{n-1}$
Reactions to U.S. *

4) Genl. work : Main Interest in A.I. : Applicn. of CBI to ~~xxx~~ Practical. problems.

a) Makon: obtaining order of a linear process.

b) Tag. seqs.

c) Genl. prob. of "best" prob. value for given cc : $\approx P'_{M_T}$.

The idea of "volatility" or "amount of ~~xxx~~ stability" of P'_{M_α} or P'_M
$P'_{M_{1000}}$ v.s. $P'_{M_{2000}}$ v.s. $P'_{M_{3000}}$ any c.B.   any machine.

Possl. analysis of human ~~xxx~~ concept discovery as approxn. or suggested soln. to genl. problem.

d) Reactor safety ; Recomb DNA : meaning of prob. of 10^{-10}.

e) **IPC :** Use of computing cost (≈ dollars) as proper measure of things to be optimized: not speed, not ≷ many. Extreme waste of computing power in modern machines : $\gtrsim 10^4$? — This amounts to $\gtrsim$ 25 yrs of computer hardware development. x2 in 2 yrs is x 1000 in 20 yrs ; x 8000 in 26 yrs.

D178 (Fri)

R_x is ↑ f. of x so is useful in ~~Betting~~. Betting

L. feels that Origin of Life is unlikely in R's Universe!! Mite be probable in many universes. But we don't know a self reproducing

Is interested in obtaining a ~~xx~~ relatively non-ad-hoc

Univl. long. $\max\limits_{x} (K_A(x) - K_B(x)) = C_{AB}$.

Russian. result, int. senses that if t. non-norm R is used,)

$\frac{R(x(n))}{P(x(n))}$ will be a non↓ funct ! ? But in accord w. $R(x)$.

$\frac{P'_M(x)}{P(x)} > \frac{R(x)}{P(x)}$ but will occasionally ↓. (But what about c.B.?)

L. pointed out that there is no Universal Normed measure ≷ True : P'_M is simply t. normed P'_M ; does not dominate all Normed Univl. measures.

* He said he ~~xx~~ had shown 1 way dominance for

$K_A(x)$ & $R(x)$ ≺ or ~~x~~ ≻
perhaps, !

But was unable to show ≷ it t. other way. → That P. Gac that it was not true t. other way But he hadn't got all bugs out of proof yet. I think K_A & R are

defind in t. Z, L. paper : K_A : see p88 ; R : see p103

N3078!

Also bring Eng. xltn of K. Levin – Zvonkin paper.

Discuss w. Levin:

Gabor Belovàri

1) Janos Konner: Budapest. suggested I write to Peter Gak (Göch)

2) Kolmogorov: what is he doing?

3) What has been Levin's reaction to U.S.?

4) What has L. been doing?

5) My own recent work: α) Give him:

a) Old I.i.C.
b) Ing. seqns.
c) IRE Review. — soln. of problem of Winston.
d) Tbilisi. (I never got there, hvr.).
e) CBIS {Explain about Theorem 2 — How it is ~~...~~ to his Theorem
f) UPM (with explan. of (OPP) — But my own proof was derived from Willis.

how Most of first Thm can be ~~...~~ proved much more briefly.

β) Results on P_M' & Randomness: Give both results: Perhaps copy of letter to Schubert.

γ) My Recent work on applied probability: Objection to his idea that P_M' does not constitute a "complete" theory of probability, because P_M' is incomputable (use L's notation for P_M').

That for each value of T P'_{M_T} is a consistent theory of probty or, more generally P'_{M_α} or by CB. or P'_{M_a}, any (not nec. univl.) machine.

The idea of "Volatility" ξ — the opposite of "stability" in a probty estimate. Say $P'_{M_{100}}(x(n))$ ~~wasn't more~~ was .001 and $P'_{M_{10^4}}(x(n))$ was .001 ± .0001 then I feel that P_M' will not change much in the next ΔT = 10⁴. This is a guess — of uncertain reliability.

δ) Other view of ~~...~~ ad-hockness.

ε) Perhaps bring outline of Proposal.

f) Work on Maxim: A Baysian approach to determining the order of a process.

g) Nuc-reactor Safety.; Recomb DNA Safety

D 378 : Levin   self

concept of small self reproducing automata in space! Using sun for power;
Gravity force is $\propto$ solar lite force! both are $\propto \frac{1}{r^2}$,
Use of very many solar energy converters in space, so they are not
.05 vulnerable to enemy destruction. → 6.18

Re: Origin of life! I sort of expand about clay: Hvr. he still felt
that it was unlikely that I could get a < 300 bit dscrn. of ① self-reproducing
.10 organisms that ② could mutate & ③ carry on mutations to offspring. → 3.15

L. Said something about betting w.r.t. a binary seq., in which if
t. seq. was non-random, then he would have fortune sum of $\frac{R_T(x(n))}{2^n}$.

Here, R_T is t. unnormed P'_{M_T}. I vaguely got t. impressn. that he was
somehow taking advantage of t fact that R_T was monotone f in T —
— but it's not clear.

There is some loss of info in $P'_{M_T} = R_T \cdot$ (normzn. const.)
— we loose t. monotonicity of R_T w. T — but I don't see how this can be taken advantage of
in betting.

One Q is: why I decided that t. normed form of R was more impt:
I think it is because a) Many (if not most) seqns of interest do not end,
so P'_M is more correct. — we should try to use all info we happend to have around.
b) In comparing 2 Pams, if they are not normed, then $R_1(x(n)) > R_2(x(n))$
does not imply R_1 is any better for condl. proby calcns. than R_2.

.25 Hvr, it may be that P'_M & R differ/ only very slightly for large xs:

$\frac{P'_M(x)}{P_0(x)} > 2^{-b}$; $\frac{R(x)}{P_0(x)} > 2^{-b}$. T. the ration betw them is something like

$$\prod_{i=1}^{n} \frac{R(x(i))}{R(x(i)0) + R(x(i)1)}$$

Each factor is ≥ 0, but t. product must be bounded above.

— so $\sum_{i=1}^{\infty} \frac{R(x(i)) - R(x(i)0) - R(x(i)1)}{R(x(i))}$ must converge. : Note this is a seq. of positive terms.

each factor here must → 1 as i → ∞.
← Which means (I guess) that t. relative proby of stopping / at t. next bit → 0 as n → ∞

— I need to fill out t. argt. of .25 ff in a bit more detail! Well: if t. normzn. factor
was unbounded, then $\frac{P'_M(x(n))}{P_0(x(n))}$ would be arbly $> 2^{-b}$ / for all — which seems to be impossl.
Well, not exactly: T. normzn. factors differs for each x seq — so perhaps t. normzn ~~factor~~
.39 product can → ∞ for some x & not for others: But it can → ∞ for only a set of x of
zero measure (whose measure of zero? P_0's or R's or P'_M's?

D378 : Levin :

It ~~may~~ be possi. to show that t. probty of "stopping at t. next bit" ↓ more rapidly than any computable function — using an argt. ~ to that which I used in t. proof about T. Cover's b^*. → 6.01

• Perhaps an argt. like 2.25 ff can be usd to estimate probty of change of Probty estimate for gn. Δcc.

SN: Marvin says book by Jardean & Sibson on Mathematical Taxonomy: Uses "Information Theory" approach: He probly has copy, will try to get it for me. — It could very well be close to CBI approach. — Possibly in need of refinement w/o Ganzn. They use some kind of Axiomatizn. that tells them how to do t. coding

2.10: Origin of Life: T. Q of whether one can derb/ or create a self reprod ... etc. device w., say < 200 bits ~~[illegible]~~ in < 1 sec. or t. Q. of ~~a device~~ t. existence of a device w. a $\frac{pc}{cc} > 2^{-200}$ or whatever is a "solvable" problem, but not nec'ly "practically solvable". Only positive solns. can be found, but no negative statement is provable.

It may be that if one has usd a certain total cc in searching & has found nothing, that one can assert that t. probty of existence of a soln. is < a certain amt. — but a) I don't know how to do this b) T. meaning of "probty of existence of a soln" is unclear.

(Re b) : I think it has meaning ~ to that of "t. 10^{1000} digit of π has a proby of 1 of being 3": T. meaning is: w. t. info available on t. ~~1000~~ 10^{1000} digit of π — we have this ≃ uniform distribn.

Anyway: Levin puts problem on ~~[illegible]~~ not bad theo. footing: assuming any reasonable set of instructions, can we ~~[illegible]~~ devise a pgm. to create t. self-reprod. device w. $\frac{pc}{cc} > 2^{-200}$?

One Q: Certain Chemical reactions may just make self-reprodn. very easy. ~~Note~~ V. Neuman's / self-reprod. autom. looks very "complex": Note " reliable neuron nets were much more complex than Shannon's reliable relays: suggesting that small changes in technology can make lots of difference. → 6.11

It may be useful to write a paper on this Q. for Morisson's Journal.

It may be that we could solve this problem in real time by using nuclear or sub-nuclear computers. Levin suggs some $\Delta E \cdot \Delta T \gtrsim \hbar$ constraints — but I'm not sure its relevant. He suggs temp of operation < 3°K is unfeasible. Perhaps I should tell him about Bennett's "reversable computer" that doesn't use entropy. He feels that $\frac{\Delta E}{T} = \Delta H$ is what is needed to do one bit of computation. (Perhaps this is t. unit of information?)

P478 Levin (OOL)!

consider all codes for which $\frac{pc}{cc} \geq 2^{-200}$,

Say codes of length n; $pc = 2^{-n}$; there are 2^n such codes!

.03 $\frac{2^{-n}}{cc} \geq 2^{-200}$ $\therefore$ $cc < 2^{200-n}$

so 2^n codes of this type can use total cc of 2^{200};

,09 for n difrnt lengths considered $\sum cc = n \cdot 2^{200}$ units of cc.

This $\frac{pc}{cc}$ seems to lead to a simple picture (.09) in this case.

~~Consider also~~ consider t. fraction of codes of length r that do not converge (i.e. inf. comp loop at nth ~~bit~~ input bit), but that did converge for t. $(r-1)^{th}$ input bit: $(\equiv f(r))$! $f(r)$ must be ~~a~~ non-↓ funct of n & $\lim_{r\to\infty} f(r) = 1$ — ~~[illegible]~~ If $f(r)$ were bounded above, then t. measure of inputs that converged for long outputs, would be zero — & I think this is false! So a hyer fraction of short codes diverge.

Consider codes of length 10; say 10% diverge: then we will spend / at least 10% of 2^{200} on them — or $\frac{1}{10 \cdot n}$ of all available cc on them. So it looks like we wouldn't save much ~~[illegible]~~ via input codes that quickly converge to something $\neq$ t. corpus. — An impt. part of our cc will be spent on non-convergent codes. We still could save a factor of — $\approx$ "10.n" on this — but that will have to be looked into.

T. cutoff criterion $\frac{2^{-n}}{cc} > 2^{-200}$ (say) of .03 gives these not bad results for "random search"! & Q is, then, how to organize RS so that we don't get into t. "problem of initial pc of t. dcon of a low pc param — like Marm". I think I did have ideas about this — one idea was to not code a sequential corpus — i.e. use an $\approx$ unordered corpus — so back tracking is easier.

T. idea of .03 — .09 hvr, suggests that we can afford to ~~[illegible]~~ expend a rather large cc in order to try ~~[illegible]~~ out any possl. "short code"!

Just why $\frac{pc}{cc}$ ~~<~~ constant should be t. criterion for ~~a~~ search abandonment is unclear — but I do remember ~~&~~ $\frac{pc}{cc}$ ~~param~~ or $\left(\frac{dpc}{dcc}\right)$ maxzn, as

.32 being a desideratum in searching — so we should first try all codes for which $\frac{pc}{cc} > \epsilon_1$; then all ~~for~~ for which $\frac{pc}{cc} > \epsilon_2$, etc. — This method would then give us greatest rate of growth of pc w. cc. !

So t. search method of .32 ff may be $\approx$ optimum!

Well, maybe not so clearly! ? Say we have a trial input of length r (so far). and ~~when~~ it fails to converge after $\frac{2^{-r}}{cc} < \epsilon$, we then know that any continuation of this string will ↑ t. cc & ~~[illegible]~~ ↑ (or remain t. same) for t. numerator —

D578 **Levin.** 5

so $\frac{pc}{cc} < \epsilon$ would be true a fortiriori — which superficially proves "optimality".

Hvr. not so! Say we have expanded cc so $\frac{2^{-r}}{cc} < \epsilon$: The I. Q. of whether to continue trying is: can we expect a low $\frac{\Delta pc}{\Delta cc}$ from continuing?

We really can't tell: it may very well be that w. very little Δ cc, we will obtain a code of length ~~r~~ r or just a little longer (say r' longer); giving a rather large $\frac{\Delta pc}{\Delta cc} = \frac{\approx 2^{-(r+r')}}{\Delta cc}$. So t. method of ~~~~ 2.32 ff is not a "sure thing" — like t. $\alpha\beta$ heuristic.

.10 Hvr., an impt Q is: If we have spent cc = a on a gn. input string w.o. output,

.11 what is t. proby distribn. for spending Δ cc. = x on that string w.o. output?

We could obtain some empirical relation for this that would enable us to make fairly realistic cut-off criteria for optimum search. This sort of argt. looks very much like my work on S.P. ("stochastic Pert").

A very impt. condition in t. Progr. is that we are not allowed to "look inside" t. machine to see how things are going. This may be a very impt. constraint on t. method, & may be a very big advantage that

.20 Human search has — so if certain subgoals are attained we have a modifn. of t. curve for .10 — .11. (again like in S.P.).

Anyway using t. curves of .10 - .11 we can perhaps do an optimum search.

A method that may utilize t. sub-goals of .20, is t. old one of coding sequencially & retaining t. 1000 (say) best codes thus far. These "1k best codes" prevent Backtracking only to t. extent that t. "road not taken" is still in t. top "1k".

It would be better if we could in some way retain t. best 1k "characteristic" codes — or 1k "important factors" of t. corpus dern. — so we would be effectively retaining >> 1k codes. T. "factors" idea seems not far from non-sequencial coding. (i.e. corpus ≡ set of unorderd objects)

T. "unorderd objects" of which t. corpus consists could be: [redacted]

a) A sequ. of events having to do w. Physics

b) " " " " " " " Chemistry

c) " " " " " " " Biology ... etc.

t. times of these events intertwine, but they are in ll & somewhat independent.

D578 Larn:

a 50% op on first floor.

.01: 3.03 spec: This discn. shows that $\frac{R(x(i)\,0)}{R(x(i))}$ is very close to

$\frac{P_M'(x(i)\,0)}{P_M'(x(i))}$ for large i — for P_M' and R / this is condl. probty:

Paper Phone.

: Since the Σ err² theorem is for condl. probys. of P_M', it may also be true for condl. probys of R — tho Σ err² may be greater for R's than for (P_M') condl. probys ⟶ (See 51.01 – .27 for perhaps

demo. that t. normzn const. must be bnd for at least one x.

ABCD EFGH

.09 Aβ

.10 3.35: **OOL**: I think t. problem in OOL may not be that any of t. trials take very long. That t. "Halting prob.", a fortiori, is not relevant. We may have to simply create a code, then try t. whole code — Somewhat in ll — & have a relatively short time/to need test each code. T. big problem arrises because there are so many codes to try (say $\approx 2^{200}$).

Control data 2

Digital 12

30: 30

14

1000

54

.18: Due to grav & lite pressure being proportional, there is a certain ρ = mass/cm² ⟹ any thin film w. < that ρ will be repelld by t. sun. This is ~ 1 μ thickness for fair density. So little sails of that thickness can control their flight in space by reflecting lite, & opening or closing sectors of t. sails or making them absorbing, etc.

L. mentiond that "smallest" UMC had (symbol × states) = 30; so apparently he was unaware of Marvin's/ result. $4 \times 7 = 28$ Perhaps L created this Machine?

Ask L. about P99 (Eng) (of Z i L.) on Prim Proc. process

T. problem is not "incomputable"; since C.B. are given, t. prob. is "solvable" negatively — but I suspect that t. only way to solve it involves computations that are about t. cc available during t. 10^{10}, say, yrs. of t. Earth (or "Universe"). To show a positive soln. could require an arbly short compn. — all one has to do is show a way. But to prove no way exists within certain CB's, can require enormous cc.

Gabor Belovàri ask Peter if he knows him.

~1·6·79 Levin

0) Is L or G familiar w. ~~Jaynes~~? There were refs to his work on t. Black board

1) Best progbl. calc to get [TI $57 or $58 — ~$50 ~$90

2) Why P_M' is normzd ; what are its properties? $|\psi(t)| \cdot e^{i\theta(t)}$

3) What is best machine to use?

4) What is best set of laws for a country?

5) L feels that Entropy is impt. thing controlling "cost" of computing — I don't really understand Entropy well enuf to comment. Why did he feel t. compus. were irreversable?

---

1) On Normzn. of $R \rightarrow P_M'$:

a) P_M' is to be usd if we know t. seq. will not terminate — which is often true.

b) since $P_M' \geq R$, P_M' is better to use in gambling, since yeald $P_M' 2^n$ or $R\, 2^n$. and $P_M' 2^n \geq R\, 2^n$.

c) P_M' was chosen not because of its uniqueness, but because it satisfies

$$\frac{P_M'(x(n))}{P_0(x(n))} > 2^{-b}$$

, and, because P_M' is expressable as the limit of a seq. of normalized cpm's. T. UPM report is a step forward determining extent to which P_M' is unique.

d) I wanted to chose P_M' so it would be best for predicting.

P_0 is property and normzn (when ... cut know t. seq. continues) seems better than R — unless we use R to get conditional probys. assuming the seq. continues — in which case the **results** are t. same as those using P_M'.

e) Discuss that

$$\epsilon(n) \equiv \left(\frac{R(x(n)\frown 0) + R(x(n)\frown 1)}{R(x(n))} - 1 \right) \rightarrow 0 \text{ as } n \rightarrow \infty$$

also $\sum \epsilon(n)$ converges. $\epsilon(n) \geq 0$;

~~Is nearly monotone~~ I don't know if $\epsilon(n)$ is monotonic in n — if it is monotonic & converges $\epsilon(n) < \frac{1}{n}$.

Anyway: for large n, t. likelyhood of t. seq. stopping (if it has not yet stopped) $\longrightarrow 0$, so condl. proby of P_M' & R are $\simeq$.

$(x+1)^{-1} = x$

$x + 1 = \frac{1}{x}$

$x^2 + x - 1$

$\frac{-1 \pm \sqrt{1+4}}{2}$

$\frac{\sqrt{5}-1}{2}$

.618

.339

2) On t. ideas of a) "Randomness"
b) "The identification problem".

The second problem is "bad": i.e. often is a poor ~~state~~ approximation to what is wanted.

T. "Randomness" problem is; of interest historically; of interest wrt. "t. identifn. problem".

3) Also of interest: That making decisions discards information — So if it is possl., always ~~do~~ make all sub. decisions as late as possl.

4) (Al. Meyer on "B term") On T. "Best" ~~as~~ Universal Proby Measure: I look on this as t. main problem of the Scientific Community: That it characterizes t. particular Universe we live in. That t. "apri" can be divided somewhat arbitrarily into a Machine (or Algorithm or Process) and an anticedant Sequence. T. total Machine plus anticedant seq. is ≡ t. "Apriori".

5) IPC and Memory: a) $\text{cost}/\text{bit} \approx \frac{\text{constant}}{\text{access time}}$

So use fast memy for rapidly needed info, slower (cheaper) memy for less freq. used info. For certain reasonable patterns of use of info, one can have $\approx \infty$ memy for finite cost. This Hierarchy of memys of difont speeds is ordinarily used in big machines — But I don't know if it's properly optimized for cost.

b) Present day computers use memy poorly. 1 bit × 16k is very wasteful: $\text{IPC of memy} \propto \frac{\lg_2(\text{no. of bits})}{\tau}$; since cost/bit is ≈ const for large memys, $\frac{\text{IPC}}{\text{cost}} \propto \frac{\lg_2(n)}{\tau \cdot n}$ which ↓ as n↑.

"Best" memys are 256 bit, 50 ns, \$1. v.s. 16k bits, 300 ns, \$8.

$\frac{20 \times 8}{\cancel{20}} = 160$ (units ns, \$1) v.s $\frac{14}{.3 \times 8} = \frac{14}{2.4} \approx 6$ bits/ns \$.

Discn w. Levin:

Re: Normzn: I think that the main reason L likes R, is that it is neater than P_M'. — i.e. R_T is monotonic in T, R is universal among all semi-computable measures.

On the other hand, P_{M_T}' is not monotonic in T, & ~~universal~~ there is no universal "normzd scpm"

I mentioned P_M' giving a hyer yeild in Betting, but this didn't seem to phase him: he felt that R was optimum to within a constant, so that using P_M' instead of R would only ↑ yeald by a "constant factor" — "No big deal". I also mentioned desirability of using the info. that the seq. would not stop. Again he was not phased by this — had no orig. reply.

He feels that there is some great optimum universal Machine or Algorithm, that is Universe independent.

P. Gacs felt that an impt. Q. was how to best make approxns. to these Universal Measures. One could use R_T of P_{M_T}'; Or use a sequence of Measures of better & better approxn. — This latter mite be equivt, hvr., to making T; here a funct. of the length of the code a/o of the/corpus. I mentioned danger of getting Biased results if anything other than simply ordrd searches were used.

The problem of "Today": Both ~~Gacs~~ Gacs & Levin felt that "Today" was a particular, unusual pt. in time & so one couldn't do extrapolns. from "Today". That "Today" had a special name, & special treatment. I told them that this was not so, that one had to code things properly, but I don't think I got this idea to them.

1) Concious Mind. — Levin didn't understand this.
2) Conciousness (RTM) } Both did understand
3) discn. of very intelligent Machines, w. 1 country or 2 countries
4) T. Winograds "Blox wrld" as a dead end.
5) Dendral is — just what the Mol. construction srtn did. Apparently, the problem of determining if 2 graphs are identical "has not been solvd" It may be NP complete.

Another thing. L. said that if I knew every ~~other~~ even bit of a long seq., I couldn't predict the ~~even~~ odd bits at all. I demmured a guess. Now, I suspect that a good guess mite be that the bits are all doubld, since this could have a simple code; if the even bits are all random. Otherwise, if there is a

1.7.79 Levin

Suitable preamble, + ~~[illegible]~~ even bits could say (in ü English) "all odd bits are 1".

I didn't ask about VH Thrm
I didn't explain how CBI deals w. A.I. problem of "representation of Data".

L. was very enthusiastic about his > 1 m thick flakes + computers moving around in space — as war weapon that could focus heat on any part of Earth.

Mentioned: while R_M gave same values for various M within constant factor — it was not clear as to how m.d.f. of M + condl. probys were. Perhaps I should have emphasized this in my disc'n. of ~~[illegible]~~ + utility of Normalizing R.

I mentioned $\epsilon(n) = \frac{R(x(n)\,0) + R(x(n)\,1)}{R(x(n))}$.

+ $\sum_n \epsilon(n)$ converged. L. said that it was thot (or was shown) that for some seqs., $\epsilon(n)$ converged very slowly. That I could discuss this in more detail.

I vaguely gather that ~~the~~ while this CBI stuff was much worked on in Russia, there was little if any attempt to use it for anything not purely Mathematical.

In genl., Gacs seemd much more aware of stuff in t. International (usually U.S.) literature than was L.

~ 1-9-79 | Tues | Levin:

$\boxed{\text{Phone Conversation}}$

Say we have a funct. of 2 strings $A(x, y)$ ∋ $A = 0$, or 1.

~~Given~~ It is easy to find A giv. x, y (takes time, say ~~∝~~ $l(x) + l(y)$)

Giv. x to find a y ∋ $A(x, y) = 1$. This prob. ~~may~~ be NP complete.

If so, let us hunt for strings, y in t. foll. order: (= $2^H \cdot T$)

$\frac{2^{-H}}{T}$ $= 2^{(\ldots)}$

⟨Chaitin complexity of y (since t. desc must tell when t. machine is to stop — since y is a finite object)⟩ + ln(time to ~~apply~~ implement $M(s) = y$, s of min'l complexity of y)

T. problem of ordering t. y's in this way takes only time $\propto 2^n$, where n is t. complexity of t. soln, y. Since $n \ll l(y)$ usually, this is a great savings over trying y's in order of length — which would take a time $\propto 2^{l(y)}$.

$\approx \frac{pc}{cc}$!!

L. says he has a ~~practical~~ method of ordering t. y's in this order [which he ~~[illegible]~~ (& Danofsky) notes this ordering is a solvable problem] which ~~he can~~ takes $\propto 2^n$ time & is optimal to within a const. factor.

He says t. main problem is to devise an optimum universal algorithm. He feels that this is some truly universal thing — that it can be found on t. basis of intuitive criteria of "beauty".

I think that things found on such criteria are optimal, but not for t. reasons L. feels.

Anyway: Diffs & remarks:

.30 1) "T. cost of computing a certain y value of a certain complexity range" may not be a good concept always. I suspect that often it is poss. to compute a *set* of y values of a gu. complexity range & assign a computing cost to this entire *set* of y values.

2) ~~Occasionally~~ Often, t. values of A & X ~~will ca~~ can be used to narrow t. search down considerably. This info has to be used somehow — ~~i.e.~~ one way: we want t. complexity order of y wrt A & X. Probly Levin *was* considering this.

3) [In addition to 2), t. complexity ~~wrt~~ of y would have to be wrt *any other* info one would have about y.

Note that $\frac{1}{2}(\text{bcost} + \lg \text{ccost}) = \frac{\text{pcost}}{\text{ccost}}$! which is the criterion I had been mainly using!

For prefix codes: To construct all of them w. $\frac{pc}{cc} > \alpha_0$!

I think all we do is examine all codes until w. a particular code, one of t. fully occurs:

a) we know t. $\frac{pc}{cc} \not< \alpha_0$

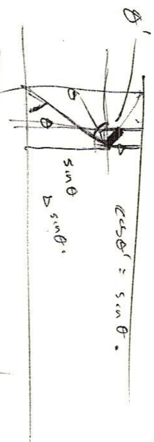
b) T. string is a code for t. corpus, w. $\frac{pc}{cc} > \alpha_0$

c) " " " not a code for t. corpus.

~~For how prefix codes: to construct all of them w. cc > α₀~~

For a very long corpus, hvr, this $\frac{pc}{cc} > \alpha_0$ is not adequate: If t. corpus is 1000 bits long & we have an # of .1, we still have 100 bits of code, & $2^{100} \sim 10^{30}$ is too large. [Hvr, after 900 bits has been coded rather well, t. no of possl. continuations of each good code can be rather small]

On t. other hand, if there really is no good way of searching, then $\frac{pc}{cc} > \alpha_0$ would seem to be best. It [redacted] could speed up ordinary random search by an enormous factor — so we would be able to deal w. n values perhaps 10x' as large as w.o. this technique.



T. problem of devising a suitable Machine that really incorporates all of one's past data: Say one is trying theorem proving. T. "machine" should include all of known Math, in as compact a way as possl. ~~Constructing~~ Also, all known heurs. would have to be included. Constructing such a machine would be equivt. to making t. most "elegant", "beautiful" synthesis of all of Math that has ever been done.

Q: How well would a machine be for thrm. proving if it wasn't that "elegant"? — i.e. say it just was fairly good at expressing commonly used Mathl ideas?

My impressn: A really "very elegant" Machine, incorporating much of Math (including thrm proving heuristics), would be a sort of TM, tug seq. soln. — in t. sense of it being ≈ equivt. to t. way a Mathematician would solve t. final NP problem — i.e. by using all trials in order of apri likelyhood — (taking into account t. cc as well as pc).

For each code length n, there are 2^n codes & each is allowed a cc $\in 2^{-n}$, so for each n t. Σ cc is constant. So t. Σ search time will then be $\propto n$.

Say we have done a search for $\frac{pc}{cc} > \alpha_0$. Then we want to continue w. $\frac{pc}{cc} > \frac{\alpha_0}{2}$ (≈ doubling search times). We could just redo t. entire search — & it wouldn't waste more than $\frac{1}{2}$ of total cc (superficially).

1·13·79 Levin — for $\alpha = \alpha_0$

Hvr., say we kept track of ~~[illegible]~~ each of those trials that didn't converge & did only those trials for $\alpha = \frac{\alpha_0}{2}$. If there were only a small ~~[illegible]~~ fraction of such trials, we would save much by only doing them: on t. other hand, t. cc of memory & access may override t. virtue of keeping track of them.

Another trick is to use random code ~~[illegible]~~ trials w. α = random variable w. suitable distribn. , 10

SN Trick to keep me from talking too fast to L.: use .1 sec. or .05 sec/ delayd acoustic f.B. !

Use of Chaitin Complexity v.s. ordinary $\lg P_i$ doesn't make much difference in t. genl. character of t. result. for ΣH^c we could use ordinary code strings ending in "δ"; t. cost of Δ is then $\approx \lg n + 2\lg\lg n$ or some similar expressn.

Apart from t. "short-cuts" (like 12.35 – 13.10), t. main Q is how to include t. desired "pre info" & what to include, & what approxns to use.

My impressn. is that if this "pre-info" is large, that t. nature of t. original [illegible] exerts very little effect on t. final result.

An impt Q in a Mathical "Pre Corpus": How much wt. should be gn. to each type of Example? This Corpus is largely "Made up" by t. Mathematician — so it can have any desired amt of repetition of any examples.

Just what would be t. best way to prepare a "proper Precorpus" for Math Σ?

$n + \lg n + \lg\lg n$

$\lg n$

$\lg\frac{n}{T}$

$= \frac{n}{\lg T} + \lg + \frac{\lg\lg n}{\lg T}$.

$n + \lg n + \lg T < \alpha$

$= n + \lg\frac{n}{T} < \alpha$

$-n - \lg n - \lg T > \alpha$

$\frac{2^{-n}}{n \cdot T} > \alpha$

$T < \frac{2^{-n}}{n}\alpha$

$2^n T < \frac{\alpha}{n}$

$\sum_{n=1}^{M} \frac{\alpha}{n} \approx \alpha \ln M$

actually, if we used $n + \lg n + 2\lg\lg n$, this Σ would converge for $M \to \infty$. But for small "M", or any other more accurate expressn. it may not make any difference i.e. t. constant for $M \to \infty$ may be quite large.

.1 sec, 7 kbs / samps/sec → 700 samps

1.11.79 Levine 14

Random Notes & ideas:

1) IPC : I IPC of a ROM is identical to problem of IPC of a adder or Muliplyer or divider.

2) IPC: Much of my thot has been about t. ~~Apc~~ IPC of arith. elements, & how to do arith. at minimum cost. RAM, hvr, I don't know how to deal w. It is usually hard to Multiplex it or get good usage out of it. Hvr., by using a heirar. memy w. suitable costs/bit for various levels, it is possl. to get a memy that → ∞, w. finite cost. It may be that such a memy is just a constant & it is small compard. to t. cc of properly usd. arith. elements.

3) IPC : In chess, no long term memy is needed. Each move branches out, say, 10 ways, & recruits 9 new computers into t. problem, each w. t. same initial board position, & a difrnt. move.

.18

---

4) IPC & RS: To do a fast search ~~for~~ for short induction codes: use t. idea of 3) : each time a branch in t. input code occures, a new machine is recruited & t. entire (fast) memy of t. [illegible] old machine is copyd onto t. new recruit. This copying can be done fast, because t. memy is very hyly interleaved.

This reduces necessity for long-term Memy.

If we use something like a Turing., t. extreme interleaving corresps to a very long word — which corresponds to a very large alphabet on t. tape.

For hy utilzn factor, we would want perhaps only 8 or 16 very long words in t. memy (perhaps a "STACK" Machine).

Perhaps only 2 or 4 words — ideally only 1 word!

Perhaps one could work borward t. ideas for 1 word, by trying very long words (say 256 of them) & trying to utilize them maximally. — Just how do t. various sub-machines interact, that are simultly working on various sections of a long word?

T. practicality of having 10^6 or 10^9 difrnt fast/ sub-machines for induction isn't too unreasonable — because one could have enuf induction probs of various kinds to keep them busy!

1 15 79 Levin!

complexity of soln. of any Math prob wrt. t. fixd corpus of (all Math info), is ~~only~~ within a "consto factor" of complexity of same problem w.r.t. < null corpus >.

That a UMC that was designed for hype / low cc wrt for a large part of Math, would also be good ~~devices~~ (yeald lo complexity solns. in reasonable time) for various NP complete probs, is ~~not~~ a matter of faith at present — Tho it certainly seems likely. Craig 367·1696 before 11pm.

.10 (Consider ~~so~~ t. Prob. in t. form: find y ∋ $A(y, x) = 0$.

While in general, t. soln. may require $\sim 2^{k\ell(x)}$ steps, in fact, t. x's that usually occur in R.W. are of low [redacted] Alg. Complexity — ~~in fact~~ (i.e. only a small fraction of all x's that can occur) ; for those x's, y can be found in ~~far~~ fewer steps. I.e. it may be that complxty of y is small if (or only if) complexity of x is small

T. thing that suggests this is ~~the~~ Shannon's experience w. Relay nets.

(This is perhaps ≃ identical to L's prob 3 in "Univl. Seql. Search Probs" Tho Shannon treats t. "Simplifcn. problem") Sh. said that in t. genl. relay net, much simplifn. is not possl., but in practical nets that occur in R.W., Simplifn. usually is possl.

Perhaps Similarly; No! Karp showd that for trav. slsmn. prob., using randomly constructed pts., (+ then by ~~Beardwood~~ Beardwood et al.) a certain simple algorithm usually yeilds near optimum Solns. { Sci Amer Art: Jan 78, p109 }

Chaitin's condl. complexity $H^c(s/t)$ is relevant here!

$H^c(s/t) = \min |r|$, $r : (U(r, t^*) = s$

t^* is th. shortest string ∋ $U(t^*, \Lambda) = t$

U is prefix machine! t. first args form a prefix set for each value of t. ~~2nd~~ argt.

$t^* \;\; : \;\; U(t^*, \Lambda) = t$

$|t^*| = \min |p| : U(p, \Lambda) = t$

$(U(p, \Lambda) = t) \supset |p| \geq (t^*)$

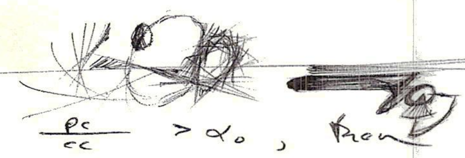
Hvr., this is not exactly an example of t. use of a random set of pts., assures us of ~~hy~~ complexity in "x", rather than low! — But maybe t. "example" isn't that relevant, since it is a soln. for a good approxn. not an exact soln.

Shannon's experience seems more relevant — since he found almost all " relay nets were not much simplifyable.

1·15·79 Levin:

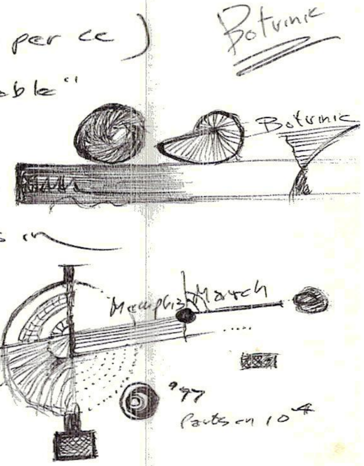
L said that this NP [redacted] "soln" was ~~wort~~ worth working on (even if ~~[illegible]~~ success was not very likely) because it was so impt. if it succeeded.

Hvr: T. main problem is to design a machine so that t. "constant ~~[illegible]~~ term" in complexity is ~~[illegible]~~ small (≈ "minimal"). This "design" problem in itself is of much importance for ~~a~~ any kind ~~of~~ of ~~[illegible]~~ practical soln. to t. induction problem.

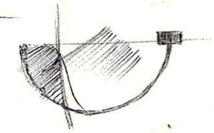


SN If we want to do optimal coding wrt. $\frac{pc}{cc} > \alpha_0$, then we will have to include heuristics in t. coding in some way — since ~~[illegible]~~ (if heurs are by defn., ~~[illegible]~~ search methods to obtain hyest pc per cc) for a given α_0, certain regularities will not be "observable" w.o. these heuristics — since w.o. them $\frac{pc}{cc} < \alpha_0$.

Hvr. note 11:30 — in which hy $\frac{pc}{cc}$ is obtained by testing a large set of possl. codes by a single technique — as in linear regressn codes.



Alternatively, perhaps we can arrange it so that this "set of strings" is decrbd by a single string or single concept." Whether or not this idea is directly translatable into CBI, I think it is pretty much t. sort of thing that I want



These "heuristics" could be part of t. machine itself.

Well: Consider linear regressn.: Consider it as a means of obtaining a very large set of codings for t. corpus in "one fell swoop" — so we get (often) a large $\frac{pc}{cc}$ for that total operation. We would like this operation ~~*~~ called "try Maxm" to be equivalent to trying an input ~~[illegible]~~ pgm for t. machine! Hvr, ~~[illegible]~~ as ordinarily done, we will not know untill Maxm has been more or less completely applied, whether $\frac{pc}{cc}$ is $> \alpha_0$ or not — infact, ordinarily, we mite expend rather large cc before we find $\frac{pc}{cc} < \alpha_0$. / Perhaps there are some "soft", approx versions of Maxm that ~~[illegible]~~ take only a little time, that will give some $\frac{pc}{cc} > \alpha_0$, then by expending more cc we still get $\frac{\Delta pc}{\Delta cc} > \alpha_0$.

There may, for some operations, be a min Δcc that we can try, below $\frac{\Delta pc}{\Delta cc} > \alpha_0$ — so we can't always have

1.15.79 Levin (1)

~~the slope of pc vs.~~ $\frac{d\,pc}{d\,cc}$ be $> \alpha_0$. T. smaller t. quantiza. level of cc we can use, t. less risk we take, i.e. t. less Σ cc we have. On t. other hand, there may be some rather larger Δcc's w. $\frac{pc}{cc} \gg \alpha_0$, that we'd want to try.

For humans — I think previous events in t. corpus would tell us whether

.10 we want to automatically risk cc on Maxm, say. So what such events do is to $\uparrow$ t. pc of t. Maxm operator — This automatically takes cc into account, since our corpus was coded using t. "CB"

.15 of $\approx \frac{pc}{cc} > \alpha_0$.

I think .10 – .15 may work (its not a new idea) — but what I also want is a way in which Maxm could have been derrd. This derry may involve Σ t. equivt. of "experimentation" — i.e. for going ~~temporari.~~ gain for a short while w. t. expectation of much greater gain in t. future (like R_nTM)

Quick Random notes:

1) Perhaps explain to L. that impt. thing about probys is not probly values, but relative probys. This is impt in OOL idea also:

No!!

2) Use of low complexity ~~adv~~ strings for DES search. Data Encription Standard $n \approx 56$

.25 3) One [complexity + $\lg T$ < search]

This $\equiv$ $\frac{2^{-n}}{T(n)} > \alpha_0 (\equiv 2^{-c})$.

so $T(n) \leq \frac{1}{\alpha_0} \cdot 2^{-n}$: If $T = 1$ sec. for $n=1$

$T = 10^{-9}$ sec for $n = 30$.

20×20

2^{16} = 64k sec $\sim$ 16 hrs

$T = 10^{-6}$ sec is not much time for a modern Machine — but could be not too bad for a hy speed, hyly 2) modern machine

$\sim$ 100 M ops/sec.

.30 So 1 sec for $n = 1$ → 1 μs for $n = 20$.

Note that total search time $= \frac{1}{\alpha_0} n$.

If we use search time $\propto$ $k \lg n$ or n^2 we mite do longer searches for single large n strings — so > 1 μs, for $n \gg 20$.

for n^2: $t = 20$ sec $n = 20$ $t \sim 1 \mu s$ for $n = 24$ or 25.

4) Not such a good idea, since t. keys could be chosen ~~at ra~~ w. Bernoulli randomness ($p = \frac{1}{2}$)! If t. codes could be inserted in a difrnt. way from t. normal (standard) way, it mite be possl.

— But I don't know enuf about t. Algorithm to tell if this is possl.

Presumably, t. amt. of info in t. key would not be degraded at any pt., by t. Encription Algorithm.

1.16 79 Levin

18

.01 1) Re: $\alpha \equiv \lg_2 R(x)$ v.s.

$\beta \equiv \min |p| \quad M(p) = x$

He says Peter Gacs showed that ~~≠~~ β is within $\lg(\alpha)$ of β.

[or was it within $\lg |x|$ of β)?

.06 Peter also showed it was > a constant greater, but found an error in his proof.

this mite be a cons of Peters early work on showing PC signifity $> 2^{-bc}$

2) For a "Process Machine" selecting codes for which $\frac{2^{-|n|}}{T} > 2^{\alpha}$

He got a total time of search of $< 2^{-\alpha}$.

$T < \frac{2^{-|n|}}{\;} 2^{-\alpha}$: I guess $\leq 2^{-|n|} \leq 1$ for t. codes that were tried.

.15 Well ok. — since they do form a prefix set ⟶ 19.25

.16 3) L has this optimal Algorithm for finding Solns to NP problems. (An NP prob. is one which, takes $\lesssim n$ to test a possl. soln.). T. induction problem is a NP problem (?) No. since if p is a code for t. corpus, we can quickly test it (?).

Anyway for NP probs., we want to test x values in order of a kind of complexity (~ Time limited Complexity)

3x4

.27 This is actually a kind of complexity $K_t(P/x) \equiv \min(\ell(q) + \lg(t_{A(q,x)}))$; $A(q,x) = P$

(t is part of K_t's name: tho t is not a number, hvr.)

A is a univl. algorithm w. 2 args.

$t_{A(q,x)}$ is t. time needed for A to produce p, w. q & x as inputs.

I think he wants to try these p's in order of complexity as possl. Solns. of T. N. P. problem $A(x, y) = 0$.

.30 (X is known, Y is to be found) — so p (I think) are trial values for y. ⟶ 19.30

4) I didn't mention to L, t. trouble of 17.25 – .30 :

5) Dan Scott: L. gave Scott's system of expressing all of Maths as an example of t. sort of thing he'd like to consider in devising Univl. machines $S_0 = 0,1$; $S_{n+1} = S_n^{S_n}$; S_∞ is t. set of interest & denumerable. (what does $S_a^{S_b}$ mean? (S_a & S_b are sets)

2^{S_2} means t. set of all sub-sets of S_2. perhaps S_2^n is t. n fold cart product of S_2 w. itself

6) L. feels that Alg. complexity theory is in much better shape than Computational Complexity Theory. That t. idea of a Univl. machine & t. intertranslations betw. them gives a "universal solution". I mentioned M. Blum's basic paper — but he said this just got optimality within some function (of n, I guess) that wasn't anywhere nearly as good as a "a constant factor".

7) wasn't successful in explaining 11.30 to L. Also no success w/ explaining 15.10.

I didn't try 17.25 – .30

1.17.79 Levin

8) Re: 5) Perhaps $S_a^{S_b} = S_a^{\alpha} \times S_a^{\beta} \cdots$ where $\alpha, \beta \cdots$

are t. elements of S_b.

If so, then $S_0 = 0, 1$, $S_1 = S_0^{S_0} = (0,1)^0 \times (0,1)^1 = 1 \times (0,1)$

$= (1,0),(1,1)$; $S_1^{S_1} = \{(1,0),(1,1)\}^{(1,0)} \times \{(1,0),(1,1)\}^{(1,1)}$ these are single pts.

I don't have a clear idea of this:

Also, in what what would 2^{S_a} then = all t. subsets of S_a ?

If $2^A = (\Lambda, A)$ where A is an element of a set,

then if $S_a = A, B, C, D \ldots$; $2^{S_a} = 2^{(A,B,C)} \ldots$

$= (\Lambda, A) \times (\Lambda, B) \times \cdots$ which is perhaps t. set of all subsets of S_a.

— But in a wierd way!

what would 3^A mean?

From Birk & McLane (1946 edn.) p 345 Discn. of β^α, where β & α are infinite cardinal nos: I get t. idea that if β & α are sets, then

β^α is t. set of all functions that go from t. elements of α to those of β.

T. no. of elements in such sets ↑ rapidly w. n. $S_1 = (0,1)$ so $N_1 = 2$, $N_2 = 4$, $N_3 = 4^4 = 16$, $N_4 = 16^{16}$, etc.

While 2^{α} (set) doesn't seem to be in t. spirit of that defn.,

t. idea of functs of sets of functs, etc., does seem to be much in t. spirit of what L. wanted.

.25 18.15 → See RS 1.3.78; 1.17.78; 4.10–.40 for a discn. of prefix codes & why this looks like a nice soln. to t. problem.

.30 : 18.30! L. said that he had this optimum (within a const factor) soln. to all NP probs. He gave this Thrm. but no proof (which he felt was not diflt). T. proof may go something like this!

.33 If $y = f(x)$ is any algorithm to solve some NP problem, then, to show his listing of y trials in order of complexity (wrt x) would do ≈ t. same thing:
using t. notation of 18.16–30 t. "y" trials are $P_i = A(q, x)$.
These P_i are ordered wrt $|q|$. Suppose ∃ an algorithm that ($y = f(x)$) can find y as a funct of x that solves t. NP prob.; then, indep of $|x|$, (even for $|x| \to \infty$) P_i will have bndd complexity wrt x.
i.e. $y = A(q_x, x)$ where q_x is ∋ $y = f(x)$
let q_{x_j} be a seq. of such algorithms in some order of computation time. Perhaps L's Theorem would automatically pick out

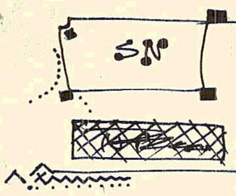
1.19.79 **Levin**

Th. lowest cc. algorithms!

Perhaps relevant: (in a negative way): one has to be careful about such things: sometimes there exists no way to do such a search: e.g. say one has a binary t.s. There is no recursive way to ↑ cc used w. n (≡ seq. length) so one can be sure that eventually / mean Σ sq. err will → 0 (if t. stoch source is recursive).

Anyway, this "optimum soln. method" would then be wrt. t. umc, **A** that defined t. complexity, $K_t(P/X)$ (see 18.23)

I guess t. cc of t. machine, $\ell(x,y)$ that defines t. NP prob., is almost irrelevant — this machine has only a "Yes-No" output & is usually not univl.



Perhaps if t. algorithm $\ell(x,y)$ is in some sense "univl", then that NP problem is "NP complete". — But I think Levin's paper reduces "NP completeness" to something like this.

from ≈19.33: say $y=f(x)=A(q,x)$ umc. — for some pgm, q is a least "Time" soln. of this problem.

.18 I think that $\exists q_0 \ni \forall x$, $A(q_0,x)$ is a optimum soln. to this problem.

I want to show (Firstly) that t. method of ordering trials, y in complexity order, would only take a constant factor longer than $t_{A(q_0,x)}$.

Write: George Levin Abel.

.20 Consider finding all $q \ni (|q| + \lg t_{A(q,x)}) < \lg T_0$

such a search take ≤ time T_0 (see RS 1.3.78, 1.17.78; 4.10–.20 for proof).

q_0 will be found in such a search w. $\lg T_0 = (\lg T_{A(q_0,x)} + |q_0|)$

.25 I.e. $T_0 = T_{A(q_0,x)} \cdot 2^{|q_0|}$: $2^{|q_0|}$ is a constant indep of |x| & |y|.

$T_{A(q_0,x)}$ is t. time to compute $A(q_0,x)$ By using t. T_0 doubling method of ibid 4.22;

.27 this search will take $< 2T_0 = T_{A(q_0,x)}\, 2^{|q_0|+1}$ or > optimum by factor of $2^{|q|+1}$.

This is t. time it takes to order the $A(q_0,x)$'s:

Each such trial value of $y_{trial} = A(q,x)$ must be tested.

Secondly How much Time is needed for testing all of t. trial y's? Consider t. search for q's of complexity < $\lg T_0$.

$$\frac{2^{-|q|}}{T_{A(q,x)}} > \frac{1}{T_0}$$

$$T_{A(q,x)} \cdot 2^{+|q|} < T_0 \;;\quad T_{A(q,x)} < T_0 \cdot 2^{-|q|}$$

.34 consider

We want $\Sigma k|q|\,2^{+|q|}$ ≈ all q's tried. $< \Sigma k |q| \frac{T_0}{T_{A(q,x)}}$

Wrong → See 21.25 — also 21.36!

I guess $T_{A(q,x)}$ is at least $> K_1|q|$ since t. machine takes that time to read q. so $< \Sigma k T_0 \frac{1}{K_1}$

.40 total search time = Hvr, $\frac{K}{K_1} T_0 \Sigma$ can be very large! Σ = t. total no. of trials.

This may be unnecy. q_0 can be a fnct of x; $q_x (\equiv q_0$ fn$)$ ∋ $y = A(q_x,x)$ is a soln of $\ell(x,y)=0$ b) q_x is t. value of q giving minimal $T_{A(q_x,x)}$ ⇒ this is satisfied.

.18 would ... $q \approx y$ + constant. would then solve this: I think t. original .18 way be right.

Levin:

This last result says that if testing time is $\propto |q|$, then the total testing time is within a const. factor of requiring ~~a constant~~ the same testing times for each trial.

Let's try to find the largest $\underset{\geq 0}{q}$ tested.

Since by 20.20, $|q| + \lg T_{A(q,x)} < \lg T_0$, $\quad |q_{max}| < \lg T_0$.

$2 \cdot 2^{|q_{max}|} < 2T_0$. so we will need $< 2T_0$ trials with a constant time (20.40) of $\frac{k}{k_1} T_0$ per trial!

This is dimensionally wrong, but it can be fixed up: the trouble starts here. Its o.k. here

Woops! so $\frac{k}{k_1} T_0^2$ is total time!

from 20.25 this is $\propto \left(T_{A(q_0,x)}\right)^2$ — which is not $\propto$ optimum time!

Also on 2nd thot, letting $\lg T_{A(q,x)} = 0$ is ambiguous — It depends on the unit dimension of T — i.e. how big is the smallest step? — More exactly, what is a lower bnd. for $T_{A(q,x)}$? Call this T_{min}:

.20 $|q_{max}| < \lg \frac{T_0}{T_{min}}$ $\qquad 2 \cdot 2^{|q_{max}|} < 2\frac{T_0}{T_{min}}$.

from 20.25 $T_0 = T_{A(q_0,x)} \cdot 2^{|q_0|}$ so $2^{|q_{max}|} < \frac{T_{A(q_0,x)}}{T_{min}} \cdot 2^{|q_0|}$

$2^{|q_{max}| - |q_0|} < \frac{T_{A(q_0,x)}}{T_{min}}$

$q_{max} \leftrightarrow T_{min}$

$q_0 \leftrightarrow T_{A(q_0,x)}$.

.25 Ah so! error on 20.34: The $\leq$ sign should be ommitted.

One upper bnd is $\underbrace{k|q_{max}|}_{<\text{ Time for trying}} 2^{|q_{max}|} < k|q_{max}| \cdot \frac{T_0}{T_{min}}$ (constant).

q_{max} is $\approx$ constant since $\underbrace{|q_{max}| - |q_0|}_{\text{const}} < \underbrace{\lg T_{A(q_0,x)}}_{\approx \text{ const}} - \lg T_{min}$

≈ const: is ~ $\propto |x|$

well, $|q_{max}|$ is $\approx \propto |x|$

$T_0 = T_{A(q_0,x)} \cdot 2^{|q_0|}$

.30 $|q_{max}| < |q_0| + \lg T_{A(q_0,x)} - \lg T_{min}$

Also, this is not an obvious upper bnd for testing time of y's!

I think, by definition of NP problems, Testing times $\lesssim |x|$.

so upperbnd for total testing times is $k|x| \left(2 \cdot 2^{|q_{max}|}\right)$ | from .20 $2^{|q_{max}|} < \frac{T_0}{T_{min}}$ (const.)

$< \frac{k|x|}{T_{min}} 2 \cdot T_0 = 2 \cdot \frac{k|x|}{T_{min}} \cdot T_{A(q_0,x)} \cdot 2^{|q_0|+1}$ (from 20.25)

Total time for ~~...~~ selecting y trials and testing y's $\Big\}$ $\left(\frac{k|x|}{T_{min}} + 1\right) \cdot 2T_0$ (20.27)

More exactly: $\left(\frac{T(y)}{T_{min}} + 1\right) 2 \cdot T_0$ | $T(y)$ is the time needed to compute $\mathcal{S}(x,y)$.

1.21.79 **Levin:**

O.k. let's cleanup this proof of **L's 2nd** Thrm: (y ≡ soln to NP probs) My discn. starts ~ 19.30

The "NP" problem: given an algorithm $\mathcal{A}(x, y)$ from pairs of finite strings into $(0, 1)$ ($\equiv$ yes, no). ~~This known the~~ x is gn., to find a $y \ni \mathcal{A}(x,y)=1$

.04 we know that $|y| < 2|x|$, say and we want an Alg $A(q_0, x) = y$ that computes y in minimal time for all x. q_0 is t. desn of this Algm. wrt Umc, $A(\cdot,\cdot)$.

.07 Usually $T_{x,y} \gtrsim k|x|$ ($|x|$ is t. length of x; k is a const.). ~~So it doesn't take long to test a candid~~ $T_{x,y}$ is t. time needed to compute $\mathcal{A}(x,y)$ ... so it doesn't take long to test a candidate y.

Suppose q_0 is a soln. (.04).

.18 We will find q_0 by listing all ~~given the~~ candidate y's in order of a special "mixd" complexity k_t.

$$k_t(y, x) \equiv \min_{\text{(over all } q)} \left(|q| + \lg(T_{A(q,x)}) \right) \qquad y = A(q, x)$$

A is an Umc; $T_{A(q,x)}$ is time needed to compute $A(q, x)$.

To do this listing, we first select a T_1, then we list all

q's $\ni$ $|q| + \lg(T_{A(q,x)}) < \lg T_1$: $\quad \dfrac{2^{-|q|}}{T_{A(q,x)}} > \dfrac{1}{T_1}$

~~The total~~

.22 These q's form a prefix set → see RS 1.3.79 4.20, 1.17.79 –.40 → or $T_{A(q,x)} < T_1 \cdot 2^{-|q|}$.

So ~~Since~~ sum $\sum T_{A(q,x)} < \sum T_1 \cdot 2^{-|q|}$

$< T_1 \sum 2^{-|q|} < T_1$

.28 so listing all ~~[illegible]~~ y's $\ni$ $k_t(y,x) \le T_1$ takes $<$ time T_1.

we then list all y's w. $k_t < 2T_1$, then $4T_1$ .... etc.

If we list up to $T_0 \equiv 2^n T_1$, then total time usd is $<$ $\underbrace{2^{n+1}}_{\approx}(T_1) < 2T_0$.

There exists some soln. to t. NP problem, q_0.

The mix ed complexity k_t of that soln. is $|q_0| + \lg(T_{A(q_0,x)}) \equiv \lg T_{q_0}$

we ~~must find~~ have found q_0 if we allowd T_0 to be $\geq T_{q_0}$.

i.e. if we allow ~~any~~ y ~~search time~~ total selection time to

.37 be $2T_0 = 2T_{q_0} = 2^{|q_0|+1} \cdot T_{A(q_0,x)}$

Since $2^{|q_0|+1}$ — see 29.01 – .10 for how to elim. this factor of 2. for (total?) — is a constant, indep of x, this selection time is $\propto$ a const. times $T_{A(q_0,x)}$ t. min. time for an Algm to compute y.

1·21·79 Levin:

Hvr, we must not only spend time selecting y's, we must also test them. Since it takes $< k|x|$ to test a candidate y (22.07) and there are $< 2 \cdot 2^{|q_{max}|}$ candidates to test, this testing takes $< k|x| \cdot 2 \cdot 2^{|q_{max}|}$ time.

.04 $|q_{max}| + \lg T_{min} \leq \lg T_0$; $\left(T_{min} = \min_{\substack{\text{all } x \\ \text{all } q}} T_{A(q,x)} = \text{const indep of } x, q. \right)$

so $2^{|q_{max}|} < \dfrac{T_0}{T_{min}}$ and total test time is $\therefore < k|x| \cdot 2 \dfrac{T_0}{T_{min}}$

Adding to t. y selection time: $2T_0 + k|x| \cdot 2\dfrac{T_0}{T_{min}} = \left(\dfrac{k|x|}{T_{min}} + 1\right) \cdot 2T_0$

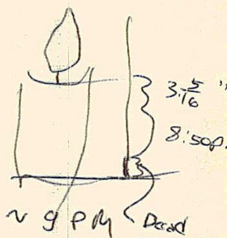
.10 $= \left(\dfrac{k|x|}{T_{min}} + 1\right) 2^{|q_0|+1} \cdot T_{A(q_0,x)}$ } This $(a|x|+b)$ factor is t. substance of L's thrm 2. See 28.20 for a better soln!

For large x this is $C \cdot k|x| \cdot T_{A(q_0,x)}$ minimal time.

i.e. $2^{|q_0|+1}\left(T_{A(q_0,x)} + k|x|\right)$ $\boxed{\text{Also shorter proof}}$ or a const. times $k|x|$ times t.

If minimal time is $\propto |x|^N$ (polynomial time) — This will give $\underline{\propto |x|^{N+1}}$

.20



poor measurement of wats, ruler.

$d \sim 3\frac{3}{16}$ ~ 9:30p

$d \sim 3"$ 9:40p

$\Delta T_{kit} \sim 1.5°F$

30 5

If 30° outside 5%/hr means $\tau = 20$ hrs.

$4\frac{1}{2}$" candle.

Say $\frac{1}{4}$"/hr.

→ $\underline{18 \text{ hrs}}$!

In t. forgg. demo, t. time for testing y's was dominant over time for selecting y's. (for $|x| >> 1$). Can we possibly trim down this "testing of y" upper bnd? Strange that testing time should be $\underset{\sim}{\propto} \dfrac{k|x|}{T_{min}}$! we ought to be able to make T_{min} arbly large! $2^{|q_{max}|} \cdot T_{min} \leq T_0$ — well, if we somehow made T_{min} larger, and correspondingly ↓ $|q_{max}|$ so to satisfy $\lg T_{min} + |q_{max}| < \lg T_0$ we might cut out t. $q = q_0$ soln! ——

Another possl. trick: That perhaps there are $<< 2^{|q_{max}|}$ trials with code length $|q_{max}|$. For long $|q|$, $T_{A(q,x)} \geq c|q|$ (c const) i.e. $q > 60$ may be (tho we may never get large enuf for this to be true! beyond present computing machinery) ?

$|q_{max}| + \lg(c|q_{max}|) < \lg T_0$

$2^{|q_{max}|} \cdot c|q_{max}| < T_0$ → $2^{|q_{max}|} < \dfrac{T_0}{c|q_{max}|}$

$2^{|q_{max}|} < \dfrac{T_0}{T_{min}}$ is also true so $|q_{max}| < \lg T_0 - \lg T_{min}$

$|q_{max}| + \lg c|q_{max}| > \lg T_0$ (for large enuf $c \cdot q_{max}$) $\dfrac{1}{c|q_{max}|} > \dfrac{1}{c(\lg T_0 - \lg T_{min})}$

Hvr, on second thots, even if $|x| \to \infty$, q_0 need not $\to \infty$.

1.21.79 Levin:

well as $|x| \to \infty$, q_0 may not $\to \infty$, but $|q_{max}|$ may $\to \infty$

from 23.04: $|q_{max}| + \lg T_{min} < \lg T_0$

" 22.37 $T_0 = 2^{|q_0|} \cdot T_{A(q_0,x)}$ $\therefore \lg T_0 = |q_0| + \lg T_{A(q_0,x)}$

$\therefore |q_{max}| < \underbrace{-\lg T_{min}}_{const} + \underbrace{|q_0|}_{const} + \lg T_{A(q_0,x)}$

which will ↑ w. $|x|$.

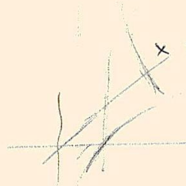
Hvr., $2^{|q_{max}|} < \frac{T_0}{c|q_{max}|}$ is a logit. inequality:

$2^x < \frac{k}{x}$ It fixes x as < some function of k.

say $k >> 1$; $2 < \frac{k^{\frac{1}{x}}}{x^{\frac{1}{x}}}$

say $k = 2^r$ $(r>>1)$ $x < \frac{r}{\lg x}$; $x \lg x < r$

$x 2^x < k$ so, solve $x \lg x = r$. ; $x = \frac{r}{\lg x}$ say $r = 30$



$x = \frac{r}{\ln x}$ $x_0 = r$

$\frac{30}{\ln 30 = 3.4} = \frac{8.8}{\ln 8.8 = 2.7} = \frac{3.6}{\lg 3.6}$

$\frac{30}{\lg 8.8 = 2.17} = 13.8$; $\frac{30}{\lg 13.8 = 2.62} = 11.5$ $\frac{30}{\lg 11.4 = 2.44} = 12.6$

$\boxed{y < \frac{k}{\ln y}}$ $(y \equiv 2^x)$ | $e^x = \frac{k}{x}$; $x_{n+1} = \ln k - \ln x_n$

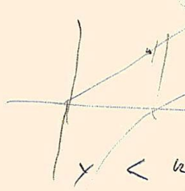
$y = \frac{k}{\ln y}$ $y < k$ $x_0 = \ln k$

$\therefore \ln y < \ln k$

$y > \frac{k}{\ln k}$ $x_1 = 0$ ~~oscillates.~~ diverges

$y < \frac{k}{\ln k - \ln\ln k}$ $x_0 = \sqrt{k}$

$x_1 = \ln k - \frac{1}{2}\ln k = \frac{1}{2}\ln k$



$y < k$ so

$y > \frac{k}{\ln k - \ln(\ln k - \ln\ln k)}$ $x_2 = \ln k - \ln \frac{1}{2}\ln k$

$y <$ $= \ln k - \ln\frac{1}{2} - \ln\ln k$

This sequence is provable — each ineq. is. $x_3 = \ln k - \ln\ln k$

$y > \frac{k}{\ln k}$ $\therefore$ $\ln y > \ln k - \ln\ln k$ $+\ln$

o.k. so $2^x < \frac{k}{x}$ $y \equiv 2^x$; $x = \lg y = c \ln y$

so $y < \frac{k}{c \ln y}$ $= \frac{\alpha}{\ln y}$ $\alpha \equiv \frac{k}{c}$.

if ~~$y < \frac{\alpha}{\ln y}$~~ $\therefore y \ln y < \alpha$ (say y is known to be $> \alpha$)

if $y \ln y = \alpha$

$y < \frac{\alpha}{\ln\alpha - \ln\ln\alpha}$ $\therefore$ if ~~$y < \frac{\alpha}{\ln y}$~~ $y \ln y < \alpha$ this is a fortiori true

same argt works for $y < \frac{k}{\ln y}$

gives $y < \frac{k}{\lg k - \lg\lg k}$ | $\underbrace{2^{|q_{max}|}}_{y} < \frac{\frac{T_0}{c} = \alpha}{\underbrace{|q_{max}|}_{\lg y}}$

$2^{|q_{max}|} < \frac{\alpha}{\lg\alpha - \lg\lg\alpha} = \frac{T_0}{c} \Big/ \lg T_0 - \lg c - \lg(\lg T_0 - \lg c)$ .40

1.22.79 Levin:

T. idea of 24.40 might help a bit:

Another possl. (small) help might be by using a mixd complexity slightly difrnt. from $K_T = |q| + \ln T_{A(q,x)}$. As it is, t. testing routine takes $\sim |x|$ times as long as t. "selection of y" routine. — So ideally, perhaps we can ~~be~~ worsen t. testing routine & better t. selection routine so they are both / a factor of only $\sim \sqrt{|x|}$ (say) greater than optimum.

(71583)

He may have meant that finding an induction code was not hard. — perhaps even finding one of Min Leost was not hard .... But getting codes of Max total pc on minimal cc is more difft.

.10 L. mentioned that $\approx$ "induction was an easier prob. than most NP probs".

If we use this routine of searching for induction codes, how well will we do? Well say q are trial codes for t. corpus, c: we want short codes, $q \ni A(q) = c$.

.15 we ~~order~~ order them via $|q| + \ln T_{A(q)}$.

If q_0 is a [illegible] minimum code, then by total time $2 \cdot 2^{|q_0|} \cdot T_{A(q_0)}$, we will find that code, using t. routine of 22.18–.37.

.20 There is, hvr., t. Q of 18.01–.06. How good are min. codes for induction? (This may not be a critical Q., but if t. "> a const factor" is true, ~~[illegible]~~ or say $\ln|q_0|$ is true, then how is induction accuracy affected?)

Also, while t. result of .20 is within a constant factor of optimum, it is a very large constant factor (☺). There may be ~~much be~~ ways to get a much smaller "const. factor" — or even a "non-optimal" method that is usually far better!

How does a "$< T$" search compare w. .10–.20? Well, we don't know how many codes there are with $T_{A(q)} \leq T_{A(q_0)}$ — well $c|q| < T$ [since it takes time $c|q|$ to read t. input, so $|q| < \frac{T}{c}$ — $2^{\frac{T}{c}}$ is perhaps very large:

Also the ordering of .15 gives ~~[illegible]~~ something like "max pc pervec —" which is sort of what we want.

In t. case of ~~[illegible]~~ a $\approx$ cpm defind by a finite set of continuous params., (like linear regressn), q_0 ~~[illegible]~~ increases w. $|c|$: Woops! This is true about ~~[illegible]~~ most common ~~[illegible]~~ true cpm's!

We would like to try to list & try all cpm's in some Mixd complxty order; The cpm's aren't linearly orderable, but I don't know if this is a ~~[illegible]~~ devastating diffy! Probly FOR's & certainly W.FOR's are not recursively enumerable. ~~Could we ever~~ We'd like to try "partial enumeration" ("partial" as in "partial recursive"). Would it help to have suitable non-normzd cpm's? Perhaps look at Zvonk & Lev.'s dscn. of "$\approx$ cpm's"

–27.01

On the notion "comparable": if, for some k,

I_1 $\left(f(n) \le (g(n)+2)^k\right)$; I_2 $\left(g(n) \le (f(n)+2)^k\right)$: say both $f(n), g(n) \ge 0$. Say both $< \infty$ for finite n

1) If both $f(n)$ & $g(n)$ are bounded by, say 8: Then they are comparable

for $k = \lg_2 8 = 3$. I. worst case for I_1 is $f(n) = 8$, $g(n) = 0$

2) If for ~~all~~ $f(n) = 0$ pts., $g(n)$ is unbnd then f & g are not compbl.
 e.g. say $f(n) = 0$ for all n, $g(n) = n^2$.

3) If $f(n)$ is bounded on some infinite set $\{n\}$, & $g(n)$ is unbnd. on that set, then f & g are not compble.

4) if $f(n)$ & $g(n)$ are polynomials in n then they are compble

5) if f & g are $2^{\alpha n}$ & $2^{\beta n}$ resp. $(\alpha, \beta \ge 0)$ then they are compble

6) if f & g are 2^{n^α} & 2^{n^β} / if α, β both ≥ 0, but $\alpha \ne \beta$ they are not compble

eg. 2^n & 2^{n^2}.

7) n^n, 2^n, incomparable:

8) n^n, $n!$ / are I think, comparable. $n! \approx \frac{n^n}{e^n}\sqrt{2\pi n}$.

He only uses the term "in time or in length comparable to n": so this means polynomial in n or $P(n)$. $\sqrt{n}$ or $x \lg n$ or $\lg\lg n$ or whatever.

9) n & $\lg n$: are not comparable.

10) n & $n \lg n$ are comparable: also n & $n \lg\lg n$, [illegible]

The funct. defind to $\equiv 0$ where otherwise complex.

.01:25.40. **In** doing induction, we often consider cpm's. T. code for t. corpus consists
.02 of a prefix code dern. of t. cpm followed by a dern. of t. corpus wrt that cpm.
After we have found a cpm that works rather well for a small part of t. corpus, we want to try that cpm on t. rest of t. corpus.
If we don't use this division of t. code into 2 parts, then t. minimal code length of t. corpus is too large to be found by exhaustive search.

T. cpm's that satisfy a certain mxd complexity criterion, are orderable wrt. ~~Basic crit~~ that criterion.

Consider Willis' defn. of a cpm; ~~we~~ A machine when we insert a string (w. end marker) into it, we get a binary fraction output telling t. proby. of that string wrt that cpm. There are certainly many machines that will give binary output w. a suitably "ended" input — but these outputs will not define a cpm. We could take any such machine, treat its outputs as binary fractions & normalize them so they did, indeed, define a cpm.

Suppose we had a machine which for every finite string input, emitted an infinite binary output. So each input string → P with P a real on [0,1].
We can renormalize it using t. conditional proby defn:

probty of 1 following X is $\dfrac{P(X1)}{P(X0)+P(X1)}$. T. normzn eq. is, I think t. same as my eq. (6) in CBIS (1978). p. 423

To get t. normed proby of X(n), it is necy to get t. unnormzd probys of about n+1 finite strings! While this "factor of n" is not very large, it is non-t. less desirable to elim. it. — There may be much better ways to generate cpm's (i.e. faster).

Dropping t. "partial listing" prob. for t. moment:
We can get a set of optimum q_0 for a cpm + t. first 20 bits of t. TS. Then try this set of v.g. q_0's for t. next 20 bits (at very low cc.). This is like t. "1000 best Pams" approach.

Hvr, I'm not sure that this approach of .01 → .02 really is as "slow" as it seems to be: work it out in more detail:

q_0 is (a t.) code for t. Pam (prefix code). P_0 is t. proby of t. corpus wrt Pam$_0$ (Pam$_0$ has dern q_0). T_0 is t. time needed to compute P_0. It includes t. time needed to "construct" Pam.

So we order trials via $\dfrac{2^{-|q|}\times P}{T}$; $\dfrac{2^{-|q_i|}\cdot P_i}{T_i}$.

Say, using t. method of 22.18 – .40 we get all $q_i \ni \dfrac{2^{-|q_i|}P_i}{T_i} > \dfrac{1}{T_1}$

so $T_i < T_1\, 2^{-|q_i|} P_i$. | Now I think $\sum 2^{-|q_i|} P_i \leq 1$

— so $\sum T_i < T_1$

1.23.79 Levin:

we continue to double / redouble T_1 . . . untill, say $2^k T_1 \geq T_0$.

— now $\dfrac{2^{-|q_0|} p_0}{T_0}$ ~~xxxx~~ $= \dfrac{1}{T}$; $T = T_0 \cdot \dfrac{2^{|q_0|}}{p_0}$.

so $2T$, t. total search time ~~xxx~~ will be very large! $\gg T_0\, 2^{|q_0|}$.

If it were $T_0\, 2^{|q_0|}$, it mite be tolerable, but not w. t. ext. factor $\frac{1}{p_0}$ — which is exponential in corpus length.

.08 Suppose we listed trials in order of $\dfrac{2^{-|q_i|}}{T_i}$) — by time ~~xxxx~~
$T_i \leftarrow$ Time to generate Pcm$_i$ $\doteq$ compute Pcm$_i$(c).

.09 $2 \cdot T_0 \cdot 2^{|q_0|}$ we will have found Pcm$_0$.

.10 SN Est. NP soln. of 22.01 — 23.20: say we order trials by

$|q| + \lg\left(T_{A(q,x)} + T(q)\right)$; $T(q)$ is t. time needed to test $y = A(q,x)$)

so if ~~x~~ $\lg T_{q_0} \equiv |q_0| + \lg\left(T_{A(q_0,x)} + T(q_0)\right)$ where q_0 is t. soln,

then we will find soln. if $2^n T_1 > T_{q_0} \equiv 2^{|q_0|}\left(T_{A(q_0,x)} + T(q_0)\right)$

Σ t. largest T / limit needed

$T_{q_0} > T_{q_i} = 2^{|q_i|}\underbrace{\left(T_{A(q_i,x)} + T(q_i)\right)}_{\equiv T_i}$

$\therefore$ ~~xxx~~ $T_i < T_{q_0} \cdot 2^{-|q_i|}$

so $\Sigma . T_i < T_{q_0}$.

and $T_{q_0} \equiv \boxed{2^{|q_0|} \cdot \left(T_{A(q_0,x)} + T(q_0)\right)}$ — Which is better (usually) than

Time needed to test t. correct y.

.20 t. soln. of 23.10 This soln. sounds more like t. one stated in L's paper. ~~It~~ is better I think, than t. soln of 23.10.

In a "sense" then $T_{A(q_0,x)} + T(q_0)$ is a minimum soln. time, ~~xx~~ since it includes testing. But one could say that it is possl. to have an algm. $A(q_0,x)$ w.o. testing it, if one knew it was correct. seems reasonable

.29

.30 ~~x~~ That the q_i form a prefix set is t. thing that assures that

$\Sigma . T_i < T_0 \Sigma 2^{-|q_i|}$ converges to $< T_0$. ~~xxxx~~

The q_i (each code for t. corpus) also form a prefix set, but it is unnecy to use such a long code! So t. total search time is only $\boxed{2^{|q_0|} \cdot T_0}$ ← which is fine: I think usually $|q_0|$ will be small — say < 10 usually or < 20 almost always!

Time needed to compute P_0.

Great!

It would seem that there wouldn't be a shorter method of listing t. possl. Pcms. If there were, one ~~xx~~ should incorporate this idea into a method of omitting certain q_i trials, or a problistic method of reordering th. trials by giving certain coding / concepts hy pc's.

1·23·79

Levin:

To gain a factor of 2 in search speed; In t. T. doubling method of 22.28 we recompute many $A(q,x)$'s for each doubling. If we stored t. ~~partial results~~ unconverged results on the shortest 10 unconverged q's, ∴ started at that pt. for t. next doubling" round, we would save a relatively large amt. of time. Almost all of the time spent is, I think, on the short q's: By saving partial results on t. shortest 10 q's that never converge, one would, indeed save about $\frac{1}{2}$ t. Σ computing time.
.10

One does, here, use storage cost: An impt. Q is whether one saves cc. $\approx$ S.T ≡ storage × Time

.11 How to (partially) list all possl. cpm's: Take a large set of cpm's that has been very useful in prediction in t. area of interest. Factor these pcms into a complete (univl.) set of concepts — then use these concepts ~~for coding~~ (w suitable pc's) to code new trial pcms. This listing will be partial, because some of t. things coded will not converge or will be somehow meaningless — but this meaningless can be determined only after ∞ time, for some of them.

.20 Another way to save a ~ $\frac{1}{2}$ of time: Instead of doubling each time, mult by 4: $1+\frac{1}{4}+\frac{1}{16}\ldots = \frac{1}{1-\frac{1}{4}} = \frac{4}{3} = 1\frac{1}{3}$ rather than $1+\frac{1}{2}\ldots = 2$

Hvr, in multiplying by 4, one runs a greater chance of "overkill". Also, if one has a maximum total amt. of cc available, in "mult by 4," when one is interrupted $\frac{1}{2}$ way thru t. last stage of search, one loses (on t. average) more than being interrupted in a "×2" search
.24

.30 One thing not included in this approach, is th. use of info "from other modalities" (like of a PMTM) to modify pc's of concepts usd to derv. candidate pcms: impt.

E.g. T. invention of linear regressn. techniques. Here, I think both cc ∴ pc are impt: T. use of soln. of system of linear eqs. to get rite coifs. is impt. — ∴ is a much shorter way than simply trying all pssl. sets of coifs.. I would think that it would be necy. to have a tng. seq. that would ~~make arith~~ include arith, algebra — (including soln. of ~~simult.~~ sets of linear eqns). Also, perhaps, RW problems that would give t. needed concepts hy pc. → 30.25 spec

1.23.79 Levin:

In t. search of 22.01 –.40 suppose we use a difrnt. wt. for T,

t. ordering is w.r.t. $|q| + k \lg T_{A(q,x)}$.

$|q_0| + k \lg T_{A(q_0,x)} \equiv \lg T_0$

$2^{|q_0|} \cdot T^k_{A(q_0,x)} = T_0$; $T_{A(q_0,x)} = T_0^{\frac{1}{k}} \; 2^{-\frac{1}{k}|q_0|}$

If $k > 1$, $\sum 2^{\frac{1}{k}|q_i|}$ may not converge.

If $k < 1$ it must converge; But total search time $= T_0^{\frac{1}{k}} \sum_i 2^{-\frac{1}{k}|q_i|}$

.10 If we make $k \lesseqgtr 1$, will this product ↑ or ↓?

The Σ factor will certainly ↓ & t. $T_0^{\frac{1}{k}}$ factor will ↑

say $k|q_0| + \lg T_{A(q_0,x)} \equiv \lg T_0$; $\sum_i T_{A(q_i,x)} = T_0 \sum_i 2^{-k|q_i|}$

$T_0 = 2^{k|q_0|} \cdot T_{A(q_0,x)}$. so again; if we ↑$k$ T_0↑, but this factor ↓ —

& we don't know which does more! T. Q is; does $(2^{|q_0|})^k \sum_i 2^{-k|q_i|}$

↑ or ↓ as k ↑? Well! The shortest q_i in will contribute most to

t. sum. Say $\sum 2^{-k|q_i|} \approx 2^{-k|q'|}$ (q' is t. shortest code used)

$2^{|q_0|k} \cdot \sum 2^{-k|q_i|}$ Then $\approx 2^{|q_0|k} \cdot 2^{-|q'|k} \approx 2^{(|q_0|-|q'|)k}$.

Since $|q_0|$ is almost certainly $>> q'$, $k > 1$ will ↑ this product —

.22 Which we don't want. So I guess we'd best let t. wts be as is!

What is t. effect of scaling T? $|q| + \lg T_{A(q,x)} + k \approx \sum_i \lg T_i$. → 41.03.

I suspect it does nothing.

.24

.25: 29.40 →: A common way to do induction (or part of a common way ⓒ).

Corpus is broken into sequential segments: $C_0, C_1, \ldots$

C_0 C_1 C_2

Segment C_0 is worked using concept pc's as of 29.11. Then t. Pcm used to code C_0 is factord & t. pc's of t. concepts (including new concepts if any) is updated. Using this new updated set of concepts & pc's, C_1 is worked, then t. concepts & pc's are again updated, etc.

T. sequence of "problems" or "sub corpi", $C_0, C_1, \ldots$ constitutes a "tng seq" & can be used to bring a machine to any desired level of competance!

Note that if t. concepts started with are v.g. Then it will take a short search (say $|q_0| < 10$) to code C_0. In general,

$2^{56} =$
$10^{16.8}$
$= 10^{12}$.
$10^3 \cdot 10^{1.8}$
$10^{.8} \approx 8$
$2^{56} =$
$2^{50} \cdot 2^6$
$= 10^{15} \cdot 64$

if t. subcorpi ~~=~~ C_i, C_{i+1} are "heuristically close", then t. soln. of C_{i+1} will have small $|q_0|$. Hvr., t. smaller t. $|q_0|$'s are, the more "info" ≡ "Guidance" we have to put into t. trng. seq. — ∴ t. more "personally biasd" t. trng. seq. is.

Some Big Problems:

.10 1) How To list good Probs. & factor them: (29.11)

2) How to use info from other Modalities (in particular, how to "invent" linear regn) —29.30

3) How to construct & pgm. real machines for Minimum cc [The IPC problem see IPC file]

4) How to allow TM to "look at t. whole $\overset{sub}{}$corpus" befor trying to code it: (31.25) [31.25 — 32.35 looks like a v.g. approach!]

It will be necy to keep t. T limit (≡ CB.) t. same (or only slowly varying) for consecutive C_i segments. This is because t. pc's of various concepts depend on t. C.B. e.g. certain concepts used w. hy cc, will be impossible to implement, & get zero pc, if cc limit is ~~so~~ too small. This "T limit" will also have to depend on t. "length" (or total bcost?) of t. sub-corpus being coded, in order to be comparable to ~~those~~ "T limits" of other sub-corpi

.25 Note that t. Technique of 30.25 – .40 operates very apri wrt. t. corpus. If one were allowd to look at t. corpus, entirely difnt. cpm's mite be implemented. E.g. linear regn. (as ordinarily done) does involve looking at "t. corpus as a whole". I don't know just how this (very impt.) technique [redacted] is (in general) done, nor how it can be integrated into t. method of 30.25 – .40.

.32 What we want is an algm. in which we put in any finite corpus, &

.33 we get, as output a cpm of hy pc that gives that corpus hy pc.

Actually, this could be done by t. ~~[illegible]~~ apri method of 30.25 – .40 (& specifically 28.30ff), but in t. case of linear regn., $|q_0|$ would be too large. ~~[illegible]~~ This apri method would, essentially, examine all sets of coefs. For 4 coefs & 8 bits accuracy, this gives a $|q_0|$ of 32. (8 bits accuracy means $\sim (2^8)^2 = 2^{16} = 64k$ data pts.) — well 10 coefs & 5 bits accuracy is 1000 data pts & $|q_0| \approx 50$ — which is too large — much > that which one should be able to get if one was able to look at t. corpus.

.01 Well, we could order all algms like 31.32–.33 (& assign cc's to them if nec(?))
Then, we could obtain pc's for each of them, so we could try them in ≈ order
of likelyhood or order of $\frac{pc}{cc}$ (Here we have estimate of cc & use it for
prelim ordering of trials: but we discontinue a trial if t. $\frac{pc}{cc}$ threshold for that
round is too low — [illegible] Remember pc is t.
pc. of t. alg. being used, & is known apri: cc is sometimes known to some approximation,
so we can save some time by trying later in t. apri ordering, algm's
.10 that are likely to be too low for t. $\frac{pc}{cc}$ threshold of that Round.).

SN I've written a very lot on $\frac{pc}{cc}$ in ~~search technique~~! Some of it
Much in t. spirit of Stochastic Pert ....?!! Perhaps try to make a
Bibliog ~~of Some of That~~ work: There may be a review of Some of it in "Rev".

.15 .01–.10 looks like a reasonable approach to t. prob. of 31.25. Each algm. is
coded by a seq. of symbols (w. difrnt pc's). Each symbol is a difrnt "concept".
After many sub. corpi have been coded, we look at t. code seqs of t. algms.
that have been successful, & we try to find regys in this set — we
can treat it as but another sub-corpus if t. probs. of t.
previous corpus where at all ~ to t. problem of recognizing similarities
.21 & regularities in dcrns. of predn algms. This is a sort of TM_2
approach, but in t. present case I may well have automatically included
cc into t. entire process! — So that devry of heuristics is a natural outgrowth of t. methodology.

→ See 36.20 for some relevant remarks

One thing is that all pc's are wrt a gn. cc threshold, so that
if a concept has a hy pc., it is w.t. understanding that it is to be used
in constructing a pam within the common cc (≡ C.B). [Think
this automatically includes heuristics (≡ search speeder uppers)
.30 as things that t. technique looks for.

→ See Rev, (Levin) 3.01 for Moravcsik. of Heurs.

Another impt. idea is .01–.10: it is a less cl. idea of
coding & predn. than t. purely sequencial methods discussed
.35 befor 31.25.

damolished, hwr. !

A difty: We do have to be sure that t. dcrns of t. algms of .15 ff
are prefix codes: This is so we can use the idea of ~ 22.22 ff
Hvr., since t. Algms. of interest are a set of finite objects, they must
have chaibinish dcrns., that tell when t. dcrn. has been completed
— so t. dcrns must form a prefix set!

1.23.79 Lev.

One thing I got into t. last time I was involved w. "looking at t. whole corpus" before choosing a fcn., was t. possy of ~~cheating~~ by humans. I Think that in Th. machines, I can fix it so There is no possy of cheating — In Humans, "cheating" can occur if t. human assigns spuriously hy pc to certain [redacted] "regys" occuring in t. corpus. Since Those pc's are assigned subconsciously, we can't criticize The method by Which t. human pc assignments were obtaind. In machines That are "open" to us — we can make such critricisms — Moreover, we can speacifically design t. machines so That pc's are assigned correctly ..... no possy. of cheating!

They will do This because t. human's goals are not nearly ~~[illegible]~~ good predn.

→ also, t. humans may use some gross approxn. methods

So: A Big Problem: To apply t. factoring idea of 29.11 to creating algms. for $\boxed{31.25 - 32.35}$ i.e. "look.at.t.whole.sub-corpus" algms.

I Think many (if not Most) algms used in statistics (& elsewhere) are of This sort — So it shouldn't be hard to find examples.

An impt. problem (but perhaps not at present a bottleneck) is how those algms were invented — how They could be gn. reasonable hy pcts using ideas from "other modalities".

.20 Some ~~Predn.~~ Global inpt. Algms: (1) Maxm & any linear regressn (2) Clustering of various kinds: Here one looks at t. set of pts., looks for "clusters": one has various a pri ideas as to what legit. "clusters" might be.

(3) Ngram definitions ~~4~~ (e.g. t. use of defns for predn. in Z & C II)

(4) Devry of Grammars by looking at a set of a's.

(5) One looks at corpus (1 or 2 or N dim.). One looks for "configurations of elements" That occure w. "unusual frequency" ... This is a kind of Genzn. of (3)

.... In fact, I Think it is rare to do "sequencial predn." (i.e. looking only at t. past of t. corpus). Perhaps it is never done. T. main value of t. "seql. predn." concept. is That it enables us to correctly assign ~~[illegible]~~ probys via various predn. methods (sequencial or (usually) "global").

.35 Using The ~~Sum~~ $\left(\frac{pc_i}{T_*}\right)$ criterion to determine when to cutoff a trial algm. is certainly fine for algms. That may possibly not converge (or do so in an unreasonably long time) — but if we were certain That algm; would, indeed, converge w. a $\frac{pc}{T}$ of $> \frac{1}{T'}$ ~~[illegible]~~ (& $\frac{1}{T'}$ was $\leq$ t. Threshold at That time) — Then if we were $\frac{1}{2}$ way Thru ~~a algm~~ algm; — perhaps we should discard it at "That round". But if we know it would take ~~[illegible]~~ need $\frac{1}{T'}$, Then we should have not started it on t. $\left(\frac{T'}{2}\right)^{-1}$ round! Well suppose we know → 34.15

1·23·79: Lev.

$\boxed{\text{SN}}$ In my work on OOL, I got t. eq. $\frac{T_0}{P_0} > T_{available}$

In which $T_{av.}$ was cc available (product of time & mass) since start of life on earth ← & perhaps energy or entropy.

T_0 " " needed for t. successful trial

P_0 " probty (pc) of t. successful trial being made

T. eq. is t. cond. under which (I felt) random creation of life was a reasonable hypoth.

Turns out this is true if an optimum search strategy is usd. — Did I have any such strategy in mind? Well: Actually, I may have assumd T_0 about th. same for all trials — in which case random trials are about as good as an optimum strategy.

·15: 33.40 it would take betw. $\left(\frac{T'}{4}\right)^{-1}$ & $\left(T'\right)^{-1}$: Then on t. $\left(\frac{T'}{2}\right)^{-1}$ round we find it takes $< \left(\frac{T'}{2}\right)^{-1}$. Well, I think this may be solvable: In t. next $\frac{T'}{2}$ seconds we will certainly get $\frac{\Delta pc}{\Delta T}$ expected, so $\frac{\Delta pc}{\Delta T}$ would be above threshold — so we should continue. [Considerations of expected future $\frac{\Delta pc}{\Delta T}$ should govern such decisions.] — we want to operate ~~so that we~~ get max $\frac{\Delta pc}{\Delta T}$ or max Δpc per unit time. Note that in trying to maxz. $\frac{\Delta pc}{\Delta T}$, we have to decide on a minimum ΔT (≡ "grain size" or ~~"quantum~~ "minimum quantum"..)...for ΔT smaller than this Δpc can be smaller or

20 even zero.

First, I want to characterize the "Global algms." Each has a prefix set dern. Its input is t. corpus; it outputs t. probty of t. corpus wrt t. algm.

How can we keep t. algms "honest"? e.g. An Algm. could conceivably assign probty 1 to all inputs. → One way would be for t. Algm. to have as output, (rather than probty) one or several codes for t. corpus. There may be variations on this (essentially foolproof) method to insure "honesty".

→ An object equivt. to a set of codes for t. corpus is a cpm, that is able to assign probys to all conceivable corpi.

Actually, all t. algm need have for output is a ruff approxn. of t. probty of t. corpus. This could be like just t. best code or a bunch of "best" codes. After we have found t. set of best Algms, we can spend more time obtaing more exact probys from them.

What about linear regn? We note on 31.32 – .40 that w. 10 coifs, 1000 data pts ≈ 5 bits accuracy, $|g_0| = 50$. If we are allowed to "look at t. entire corpus", this situation is not improved — t. pc of t. best set of coifs is $\sim 2^{-50}$, and would be found only after a search of $2^{50} \times T_{test}$ seconds (T_{test} is t. cc of creating this linear regn. param. & applying it to t. corpus)

The description of how to obtain t. linear regn. coifs from t. corpus could be << 50 bits. Is there someway that these bits could be used to control t. search? — I think this is essential!

O.K. so that's what we do. The ordering will be on t. basis of t. dern. of how to obtain t. pam from t. corpus. We use $\frac{pc\text{(of that dern)}}{T_{(test)}}$ as an ordering param.

As always, t. criterion of "goodness" of one of these pams is ∝ t. pc of t. pam times t. pc of t. corpus wrt that pam.

It looks like there's some confusion here, (but not too bad!)

In t. cases of pams like linear regn., t. derns are broken into

.18 2 parts: ① T. name of t. method e.g. linear regn. ② a listing of t. params e.g. ③ (no. of coifs), $\underbrace{15.10, -1.35, +2.11}_{\text{coifs}}$, $\overset{\sigma^2}{.11}$. T. first part can be

Hvr, see .30

a specially defined "symbol" (≡ "concept") if it is commonly used, or, it may be a dern. of how this pam type is formed.

T. entire code of t. corpus consists of ①, then ② of it, followed by ③, the list of predn. errors in gaussian error notation. ; ①, & ② are prefix codes for finite objects. ③ is a prefix code, but only because it is a "minimal code" (in t. sense used in CB1S P424, col 1, near middle)

I guess that usually ③ (or a set of sort ③'s) would not be gn., but only t. probty of t. corpus w.r.t. t. pam dervd by ①, ②.

So: In doing predn. of a T.S., we would probly first try all of t. commonly used methods of statistical predn. — t. ones that had "names" — they go into the construction of new types of pams.

.29

.30 Hvr: Impt. Q! ~~Suppose~~ Consider ordinary linear regn., in which any no. of coifs can be usd. Since t. no. of coifs is specified by t. pam (& is not computed by t. pam as it is for Markov) t. no. of coifs usd is part of t. "Name" of t. pam.

A peculiar phenomenon occurs here! .... we would do trials for a large no. of diffrnt. nos. of coifs. The drop off point for testing would not depend on t. final Gorc, but would depend only (initially) on t. product of t. p.c. of t. name "linear regn" & t. pc of the (prefix coded) no. of coiffs. T. first factor is constant, t. 2nd factor drops more rapidly than $\frac{\sim}{2}(\lg m)$ ← t. no. of coifs.

– less rapidly than $\frac{\sim}{2}(\lg m + 2 \lg \lg m)$, ect. so $< \frac{1}{m}$ but $> \frac{1}{m} \cdot \left(\frac{1}{\lg m}\right)^2$.

I think $< \frac{1}{m} \frac{1}{\ln m}$ but $> \frac{1}{m}\left(\frac{1}{\ln m}\right)^2$ (Actually, any convergent seq $f(m) \ni \sum^{\infty} f(m) = 1$ is fine!

1.24.79 Lav:

.01 T. reason This is peculiar, is that what a human searcher would probly do, would be to stop increasing m, when it began to seem very likely that t. final Gcvc was going down & was very unlikely to ↑ w. further ↑ of m. It is conceivable that t. Gcvc would be monotone ↓ after a while.

Very IMPT! I did write something about cross coupling betw. parallel codes of a corpus — but I don't remember just what it was! I think it was within t. last year. I may have mentioned it in t. Reviews, since it seems rather impt. .01 is an example of induction while searching. I think that Humans do it a lot, but I can't (immediately) justify it. Another common technique of this kind, is hill climbing on t. params of a set of pcms (wrt t. Gcvc).

78 PW 1.01 – 25.40 7 10 78 – 7 24 78 has some stuff on ll coupling. "Cross coupling" 24.01 – 25.40 in particular.

It may be that this kind "learning while searching" does, indeed, give one greater pc. for a gn. cc, but it is more likely to BIAS t. search.

If a Pcm has only 1 param (like, for linear regn., t. no. of coefs), then searching over them is not so bad; but if it has 2 or 3 params, t. search space rapidly becomes very large. This gives "no parameter" methods like Maxm a rather large apri wt.! Trouble is, I suspect that methods like Maxm will not be tried untill T.M. has learned an enormous amount of Math. & things other than Statistics!

.20 This "speeding up" of t. search looks like a TM_2 problem. Hvr., recently (32.21) I had a way to do "TM_2" as a regular TM_1 problem in a very direct way!

.25 One way to implement .01: Perhaps within t. present system! When t. human in .01 notices this, he is essentially devising a new Pcm. — E.g. Consider lin. regn.: we want to find good values for m rapidly. So we devise a new Pcm, which automatically finds the proper m by considerations like .01. This new Pcm does not have an m specification in its dcrn., so it is of hyper pc — Tho it can have a much larger cc than linear regn. pcms w. specified m, since it will usually be nccy to try several m values befor an optimum one can be found. Perhaps it takes $\lg_2 m_0$ trials to find t. optimum m_0.

.35 So contrast $\dfrac{\text{pc of Pcm w.o. m.}}{T \cdot \lg_2 m_0}$ v.s. $\dfrac{\text{pc of pcm w.o. m} \times (< \frac{1}{m})}{T}$ (pc. of de'rbng Pcm with m.)

So this technique is better by factor of $\approx \dfrac{> m_0}{\lg_2 m_0}$.

for $m_0 = 16$, this is factor of >4
for $m_0 = 32$ " " " " >6.4 } so! worth while!

Not BAD!! The Genl. idea of .25 ff does seem v.g.

Actually, in .35, t. left hand expression would be chosen first, anyway because its apri pc. is larger.

because its Pc. is larger: & Pc is ~~[illegible]~~ t. first thing usd to order trials → see Rev, (Levin) 3.25 for more discn

T. forgo. stuff looks like a nearly complete outline of a practical soln. to t. induction problem! Write a somewhat detaild review, giving list of unsolvd problems, diftys. ← see 31.10 for some. | 1.1481 |

This system seems to solve a rather genl. difty that I had w. "R.S.": In linear regn., there was an enormous initial bcost of deriving th. system — including the coifs & vars. For any search system to invest this much time in an enormous no. of difnt possl. Pams of equal bcost, seemd a rather unproductive approach. Th. present system seems to avoid this nicely, & furthermore gives a good upper bnd. on needed search cc.

Furthermore, t. method that it does linear regn., is probly very close to how humans do it, & in T.M., this "discovery" of linear regn. can be brought about via a tng. seq. That would be perhaps adequate for Humans also!

.18 In R.W. t. corpus isn't so neatly broken down into sc. of = size. Just how can this be dealt w. in a TM that's fed RW data?

.20 This present method does have a kind of assoc. "$\alpha\beta$ heuristic": Say t. sub corpus being wrk. on is c_j. Say one has already found a code (or / pc assoc assignment) w. pc $= 2^{-50}$. Then, in evaluating a new / trial algm, t. pc is down to 2^{-60} & one hasn't gotten thru t. whole corpus, so one discards this algm. & tries t. next. How much time can be savd this way if t. minimum algm. is chosen early — say first? — well, say $T_a = T_0\, 2^{|q_a|}$ is being usd.

so for a trial, w. dern. $|q_i|$, wo'd normally allow time of $T_a \cdot 2^{-|q_i|}$ for construction of Pam$_i$ & evaln. of c_j by Pam$_i$. This $\equiv T_0\, 2^{(|q_a| - |q_i|)}$.

Say these sc's are long, so t. principal search time is in evaluating t. pc. of t. sc wrt t. various Algms, rather than t. "≈ fixd time" for constructing t. algm.

...... well, I suspect that this will not save much! That t. most time that will be spent will be on ~~relatively sho~~ Algms of short $|q|$, that do not converge — so we never get around to knowing if they give a hy or low pc to part of t. sc. — we just spend our alloted time on

.35 their non-convergence.

.36 I think that impt. theoretical Q's are: 1 Whether all types of heurs can be automatically implemented within this system — & just How. 2 How to deal w. sc's of difnt. lengths. → see 40.25 for one possl. approach (perhaps Adequate) 3 How to Legitimately implement 36.01 by 36.25: (This is t. idea of noticing things during search & having these noticed things control future search to some extent.) — see 40.30 for possl. approach — (seems reasonable.)

354 9769

T. present system has as output (1) a seq. of pc's for t. seq. of sc's
(2) A seq. of "successful" Algms: 1 or more for each sc.

After doing each algm., t. seq. of successful Algms, is regarded as a T.S., & is usd to give an apripd for t. Algm. that will be t.d. for t. next sc.

This idea is a good genzn. of t. idea of "updating t. pc's of t. old & new abss. used to create Algms".

.08 So there are sort of 2 induction problems: One w. t. seq. of sc's itself, & one with t. seq. of successful Algms.

.10 One way to obtain t. heur. device of 36.01 & 36.25, is to regard the actual process of search as something that is observable by t. TM as an aux source of info: Another mode of PMTM that is usd in extrapolating t. sequ. of "Good Algms".

,15 = Primary T.S.

Re: t. 2 time Series' idea: T. seq. of Algms., as a T.S. can be worked just like any other problem in t. primary T.S. — in fact, it can be (a regular problem) (i.e. a sub-corpus) in t. primary T.S.

.195 An impt. difty here, is that it doesn't have t. same no. of symbols as t. s.c.s.

Def .20 Anyway, when viewed in this way, we construct a special Algm.* ~~that is good at~~ extrapolating t. seq. of Algms. This Algm* is usd to create a set of trial Algms (in aprip order) to work t. next s.c. —

— It may be desirable for Algm* to produce trial Algms in Random, Monte Carlo fashion, if this is easier to implement, hardware-wise.

The idea of simply updating pc's of concepts using frequencies of their use, is a particularly simple kind of Algm*.

Aside from t. difty of .195; T. Algm$_i$'s may have difnt syms. than t. sc's.. Algm$_i$'s will be a's in some generative grammar, say — T. sc's usually will not be. One ~ necessary way to deal w. this is that t. sc's must be ~ to t. Algm$_i$'s if they Algm$_i$'s are to be extrapolated like t. sc's — so if we want to do it that way, (w. this type of $TM_2 = TM_1$), we will have to have some of t. primary sc's be problems in (finding) generative/stochastic grammars — or finding whatever kind of thing t. Algm$_i$'s are.

Note that for a long time, I can be Algm*, then later, Algm* can be fairly fixed, with occasional help from me — and a year later, TM_1 can try its hand at improving Algm*.

ABC
ABC ABC
ABC

1.27.79: Levin:

Phone discn: He thinks it likely that there is an optimal algorithm to derv. all of Math. — That t. "constant" (complexity) relating it to other derns. of Math may be small ~ 10.

I mentioned to him that I didn't believe that abs. probys were of any ~~importance~~. — That it is almost always relative probys that are of interest.

He said that (dimension-less) consts were usually small: I mentioned astronomical constants; He said that ratio of elect to grav. force was ≈ ratio of size of proton to size of universe; That these ratios were ≈ because determined when stars would form & when life (ours) could exist, to observe "present" size of universe. So this ratio mite determine a Time, This time mult by vel of lite mite give ≈ radius of Universe.

But he felt it likely that this large dimensionless constant had a explanation in terms of simpler constants.

Tho this may be, it still should be possl. to proceed w. work w.o. having to derv. this (or even hypothesize it?).

ABC
α α
α α
α α
ABC

1.27.79: Later at Party: Discn about Religion: Russian Official "Atheist Materialism" line is that all ethical problems have been solved & one can get t. word by reading Marx. He feels that there are imp. probs unsolved, & he looks to religion as having possl. ideas on them.

Nutrition: That accepted nutritional requirements are probly not too far off for most people, so by eating a reasonable mixd diet, one would most certainly get enuf of needed components. — This is about t. standard AMA line. That individual requirements differ widely from t. mean is not considered.

Also that "Natural Vitamins" seem to work better, suggests that cross products w. other (some unknown) components may be impt. This would also account for wide variations in t. usual methods of determining minimum daily Requirements of components.

.01 In trying to create t. next Algm$_i$, t. first set of trials will be t. previously successful

.02 Algm$_i$'s (in backwards order). This will save some time in t. manner of 37.20 – .35

— if, indeed, there is much

.03 Another ~~approach~~ Entire approach, is that there is only one Algm$_i$, that is trying to extrapolate t. primary T.S. After working each ~~sc, we try~~ incremental sc, we try to modify Algm$_i$, so that it will be better for t. entire corpus — but in evaluating trials, we can just test them on selected parts of t. corpus, so as to save time, ~~ ~~ How we would get suitable statistics (or a suitable corpus) for Algm* is unclear: perhaps t. sequence of successful modifns of Algm$_i$. This system then becomes t. same as t. previous ones, except that here, t. corpus Algm$_i$ operates on is always t. concatn. of all sc$_j$'s to date. There is t. problem of all

.17 corpi being of diffrnt. lengths, hvr. — see it .25 will help here. → Also, see 42.18 →

T. 3 Impt. ~~~~ problems: (≈ 37.36)

1) How to deal w. sc's of diffrnt. lengths. → see .25 That t. heuristic properties should work out properly is of much import.

2) How to implement 36.25 – .40, 38.10 – .15 ~~~~ (watching TM, try to work problems ~~~~ part of primary (or scory) corpus)

3) ~~~~ 37.36. Are all types of heurs implemented in this system?

.25 ① One rough & dirty way to deal w. sub. corpi of diffrnt lengths!

Hvr., Note that a "Rough dirty" soln. may be Quite adequate.

Do .01 – .02 on t. new sc. By observing t. time needed for those old Pam's to evaluate t. new sub corpus, & comparing this time to that need to evaluate old sc's (take geometric means by taking logs), we can obtain an expected ratio of cc of t. new sc. & ~~~~ if To was t. CB used on all those old sc's we will mult To by t. appropriate factor to get t. new CB. An impt. Q is: Just how are heuristics effected?

.30 ② On watching TM. work probs: (36.25 – .40; 38.10 – .15) Including this as part of t. primary corpus (& looking into t. secondary corpus. = one of t. sc's) is O.K. It just means that we have to have other sc's in t. corpus that are like that, & also have abss, concepts, types of Algms (e.g. proper stoch. Grammar types), etc.

③ Are all heurs implementable in this system? How to deal w. this Q.: Consider specific Heurs, then classes of heurs, then, perhaps, t. most genl. class of heurs.

This is a Main Problem now. → 42.01 (spec)

Would it be possl. to have T.M. automatically choose a C.B. for any particular aspect of t. search? Is there some way to build this feature into t. system in a "Natural" way?

.03 Re: use of $\frac{pc}{cc}$ as ordering for search: (30.01–.24 is about this)

Other ways of ordering:

1) use of $\frac{(pc)^k}{cc}$: see 30.01–.24 seems not to work ~~for~~ $k \neq 1$

2) use of $T = T_0$ limit, then search in order of q. This makes Σ time easy to estimate upper bnd of, but we don't know how large q_{max} should be for a gvn. T.

3) order $= f(cc) \cdot g(pc)$ xfms to $cc \times f(pc)$ or $f(cc) \times pc$.

so, for one run; $cc < \frac{T_0}{f(pc)}$. i.e. $T_i < \frac{T_0}{f(q)}$

Hvr, we usually cannot put an upper bnd on T_i unless $f(q) = 2^q$ or 2^{kq} with $k \geq 1$.

.17 (Say $f(q)$ is any funct. of $q \ni \Sigma \frac{1}{f(q)}$ converges — then ΣT_i can be estimated.

Hvr., see .28 for a more genl. view of this

4) for ordering of $f(cc) \times g(pc)$: .17 is perhaps t. most genl. soln.

say, more generally, $f(cc, pc) < T_0$

for each value of pc, there is a value of cc (that is a funct of T_0 as well as pc)

that cc must be < so $T_i < f(q_i; T_0)$.

For ΣT_i to be badd & estimatable, $\sum_i f(q_i, T_0)$ must be estimatable w/o upper boundable. We do know that $\Sigma 2^{-q_i}$ is badd, so it is certainly easy to use that — but still, is it clear that that's t. only possy?

— say just $\sum_i f(q_i)$ is bnd. : we know only that $\Sigma 2^{-q_i}$ is bnd.

.28 O.K. so let $2^{-q_i} \equiv y_i$ Σy_i is bnd. what $\Sigma f(y_i)$ is bnd?

Well, any funct. $f(y_i)$ that is $< y_i$ for large ~~e~~ enuf y_i will do.

I think perhaps any funct $f(x)$ ~~$f(x)$~~ $\ni \lim_{x \to 0} \frac{f(x)}{x} \to \infty$ will not do, because there are (perhaps) distribns of $q_i \ni \Sigma y_i$ converges arbly slowly — so that this $\Sigma f(y_i)$ wouldn't not converge.

So this (perhaps) non-convergence argt. mite assure us that $f(x)$ is less than / within a const times x for small x.

. 30.10 — .22 suggests an argt. that $f(x)$ shouldn't be much less than x for small x.

.35 So t. result is that (perhaps), we do best (in terms of minimum expected search time), by using $\frac{pc}{cc}$ for ordering trials.

.36 T. forgg., if true suggests that if $\frac{pc}{cc}$ is t. ~~best~~ best ways of ordering, then $2^{q_0}\left(T_{(A(q_0,x)}^{+} \int T(\text{testify})\right)$ may actually be t. best one can do —

.37 that t. cc is, indeed, large, but one can't do better. — then

Trouble w. this latter argt. is that for many NP probs., a polynomial soln does exist which has a small constant of proportionality $\ll 2^{q_0}$ . . . .

so t. conjecture of .36–.37 couldn't be rite!

While t. conjecture of .36–.37 is wrong, still, .28–.35 suggest that there's something especially optimum about t. $\frac{pc}{cc}$ ordering.

1.29.79 Lev:

.01 : 40.40 (spec) : "One Heur": Linear Regn. can be "solved" by considering all codes in U. This is, hvr., very time consuming. T. usual way of obtaining correln. matrix & solving simult. eqns., is much shorter, but can be made to yield t. same result : **It is**, in this sense, a "Heur.".

—— This is, hvr, a very complicated heur, that requires much aux mode. info.

Perhaps try to find a simpler one.

---

2) Well, ~~to~~ coding t. Bern seq. by counting freqs.

3) counting digrams to obtain new definitions for coding.

4) General methods of observing freqs. of various possl. "regularities" & comparing them w. "random freqs".

T. Bern seq. is perhaps simplest & very "fundamental" — so work on it first.

Some **RANDOM** Thots:

.18 1) T. idea of 40.03–.17 is not so bad! It is more like a person working probs. in **RW**. — "Only One" Algm. for all problems, but it is modified for each new SE. 25k $25 → .1%

2) **In** either version of TM (40.03 or t. previous device — in particular), One new Algm. that can be very fast, is one that looks at t. corpus briefly, then decides which of t. regular **Pams** is most likely to work well..... t. **initial** observation reorders t. Algms. w.r.t. their a prips. 1000 $20, 4¢ = .2% $\frac{.04}{2000}$ = .2%

RE: HEURISTICS : What is a "heuristic"? Well, one way to look at it:

We have this $Algm_1$ operating within CB_1, To. It is able to get a certain pc_1 for a corpus. $Algm_2$ ~~would~~ needs CB_2 ($CB_2 > CB_1$) to operate, but it gets pc_2 ($> pc_1$) for that corpus.

.30 A "Heuristic", is a trick for modifying Algms, so they take less cc; so we can get an $Algm_3$, that is identical (or very close) in function to $Algm_2$, but which will operate faster, — i.e. within CB_1.

.32

So, it would seem then that if Algm* (38.20 Def) is looking for $Algm_i$'s within a ≈ fixed CB that have max pc, then, if Algm* itself has large enuf CB. (presumably >> that of t. $Algm_i$'s), it will find better Algms that are within t. needed CB.... so Algm* is a heur.

Hvr., lots of "speed-up ~~new~~ ideas" are not of hy aprip (i.e. they are of low pc) for Algm*, so they will be found only after much search. This search can be much reduced if Algm* has a suitable tng. seq. — bringing to it impt. ideas ~~...~~ relevant to speeding up Algms.

1.30.79 Lov:

T. ~~essential~~ feature of a heur, is that it modifies t. pc ordering of t. Algm_i's for Algm*. That it ~~does~~ should do this in t. manner of 42.30–.32 (i.e. by speeding up otherwise superior Algms) is ~~unrelevant~~ interesting & defines a certain type of heur., & ~~may~~ suggests ways to find them! But t. ~~ess~~ impt. characteristic of heurs. that we are interested in, is that they are modifns. of Algm*. that enable it to find better Algm_i's in a given time (Algm*'s "given time").

— They ↑ $\frac{pc}{cc}$ for Algm*'s work. — so th. sequence of Algm_i's chosen by Algm*, gets a greater pc for a given total cc of Algm*'s. ⟶ on Heurs ⟶ 46.01

.14

.15 If we allow TM_2 to be TM_2 (by letting ~~[illegible]~~ t. Algm_i's be part of t. normal corpus — so t. Algm_i's are SE's) then we are really not doing it ~~ap~~ t. best way — since TM_2's job is optimizn & TM_1's job is straight prediction. Hvr, since we (presumably) give more wt. to those Algm_i's which in t. past that have been most successful, Algm*, as a predictor would tend to cluster its trials "about" t. v.g. Algm_i's of t. past. It is a kind of "H.C." approach to optimization: it can work, but it gets stuck in local maxima, & lacks a "~~whole~~ picture of t. whole problem" — which it can never really "understand" from its narrow viewpt. We would like a TM_2 that really understands what t. maxzn. problem is & so it can make global theories & global efforts.

.27

.28 I had expressed t. extremzn. problem as a ~~[illegible]~~ problem that reduced to normal seq. extrapoln. — but I forgot how I did it: I think it is outlined in "PRD" — I'm not sure t. reduction that I proposed was particularly good, hvr.!

... I think one approach was this: Given a set $[I_i, R_i]$ ($[I_i]$ are objects, R_i are scalar results), to devise a I_0 with max expect R_0 (or some other criterion of optimality). ← (This is t. problem.)

One kind of soln: Take some hy value of R & use this data set to get a proby distribn. over objects that could get that result. (i.e. Its an inversion of t. $[I_i, O_i]$ set, so instead of trying to extrapolate what ~~[illegible]~~ output some new input would give, we look on $[R_i]$ as t. "inputs", & try to get a distribn. over possl. ~~[illegible]~~ I_i's.

.01 Anyway, one way to do this: we have the Algms's & Algm* & t. corpus of subcorpi of ≈ length (perhaps length corrected by 40.25). Initially, I am Algm* & I try to devise a mechanized Algm*, using various heurs I think of. By using suitable problems in t. main corpus for SE's, we eventually get Algms's that can work problems like Algm*'s, & we allow this to be done. Because t. resultant TM_1 is an extrapolating (rather than optimizing) TM, it will have t. serious

.10 diffys of 43.15 – .27 — but it _will_ climb hills to some extent.

T. object of .01 – .10 _will_ be an extrapolating TM of sorts & can be used, (After a reasonable amt. of tng) to work optimizn. probs in t. manner of, say, 43.28 – .40. Or, we can perhaps at an earlier stage of development, when we use TM_1 to do TM_2, we can use t. extrapolating capabilities of TM_1 to do an optimizing job for TM_2, using t. technique of 43.28 – .40.

Ngms → Algms ! ~~Algms~~ ~~Algms~~

Incidentally, one can _Bootstrap_ any "passably good" optimizing T.M., & by giving it t. problem of finding a good (optimum) algorithm ~~for~~ doing optimizing. After a few example Algms. (with their associated, observd efficacy in optimization (for some corpus of optzn probs), we ask it to optz itself — i.e. optz. ~~its~~ _own_ algm.

| SN | _Q_ : Did L. have a proof that for every cpm ∃ a machine ∋ the proby measure assoc. w. t. machine ≡ t. proby measure assoc. w. that? ?

.27 Perhaps essentially t. substance of my (perhaps Thm 3) in "U pm".

.28 T. forgoing ideas of search are relevant to t. following (more human-like)

TM! A Tng. _Seq._ of problems — each sharply defined from t. rest. Each problem is either of an NP type — so t. soln. is B.W. or a "Grey" soln., in which an R value is obtainable for any proposd response (a ~~history~~ R, TM (history of 1)). Induction problems are of this second type.

TM is, at ~~[illegible]~~ t. end of sc;, dscbd by $U(q_i, x)$

U is a umc., q_i is t. dcrn. of TM's Algm., x is TM's input, $U(q_i, x)$ is its output.

On t. basis of t. past set of q_i's & their degrees of success, we want to make a set of modifns on t. present q_i that

1.31.79

are most likely to solve t. present sc ($\equiv$ problem).

Usually, I expect trial new q_i's are [redacted] q_{ij} with "complexity" $C(q_{ij}|q_i)$ being small.
Search is orderd by $\frac{2^{-C(q_{ij}|q_i)}}{T_{q_{ij}}}$ q_{ij}'s are "small" modifns of q_i.

In some sense, we may want $U(q_{ij}, \cdot)$ to solve not only sc_{i+1}) but all of t. previous sc's.

I think $q_{ij} \ni C(q_{ij}|q_i)$ is small, is wrong. If q_i somehow included all of t. info. in t. trng seq. & info on what mutation types tended to be useful, would this be o.k.?

— A Q is ••• Would it work at all?

Perhaps [redacted] retain a set of q_i's that derb. t. TM at end of sc_i.

Well, for each sc_i problem, I want an algm. that will solve it best. I should think that such an algm. would be close to one that was able to solve all of t. previous probs., rather than just the single previous prob. 225

Say q_i ($i = 1|n$) are t. codes for t. ~~solns~~ Algms. solving $[sc_i]$. 20 mc 92 lab

.18 Perhaps we want q_{i+1} trials $\ni C(q_{i+1} | q_1, q_2 \cdots q_n)$ [redacted] is small. \$ 125

Chaitin may have a defn. for One reasonable defn: 5852 + 628 = 4292

$C(q_{i+1} | q_1, \cdots q_n) \equiv C(q_{i+1} | q_1 \Delta q_2 \Delta q_3 \cdots \Delta q_n)$: here Δ is a punctu. symbol (or equivalent)

& $q_1 \Delta q_2 \cdots q_n$ is a single string. Note that t. order of info $q_1 \cdots q_n$ is retaind,

& I want to retain this info.

The earliest approach (Rev. (Levin) 1.01 – 3.40) was sort of like .18 :
T. initial problem was about t. same; there, we tried to (i.e. Algm*) predict q_{i+1} from knowledge of $q_1, q_2 \cdots q_i$. T. info. in t. trng. seq. was stored (as far as this method was concernd) in t. seq. of successful Algm$_i$'s, which are characterized by $q_1, q_2, \cdots, q_i$. On .18 we have t. same info situation; again t. corpus is "summarized" by t. sequence $q_1 \cdots q_i$. It may well be that .18 is t. same as this, But this seems to be t. more obviously good approach — i.e. we are, indeed, trying to predict what q_{i+1} will be like on t. basis of past known $q_1, q_2 \cdots q_n$ Successes.

...My impression is that this device could be made to work, tho it is probly not near an optimum design, it mite be good enuf to demonstrate t. goodness of t. genl. principals involved.

Algm* would (perhaps probly) be a stock. PSG. — for assigning probys, there may be some diffty w. parsing — e.g. if we have context dependence or other complexitys.

1·31·79 Lov: 46

.01: 43.14 spec.

I. problems of heuristics: One big problem is variations in sc. length: these can be dealt w., to some extent, using t. idea of 40.25

Hvr., t. basic idea here, is that pc's change when t. CB is changed — if we mean by "pc", t. pc. of an entire $Algm_i$, then for each difnt. CB, we have an essentially difnt predn. distribn. for t. next $Algm_i$. If we ↑ CB much, we have various essential new methods of predn. that become available.

.07 If we have a $Algm^*$ that has been trained on CB = T_0, then if we use a $T_1 = 10T_0$ for a search, we will find perhaps more new predn methods that were not thot of by t. $Algm_i$'s created by $Algm^*$ in CB = T_0. Hvr., the search for CB betw T_0 & $10T_0$ will not be very near optimum, because this CB region contains devices that $Algm^*$ knows very little about (tho he has some idea, since they are related to t. techniques usd for CB = T_0.)

On t. otherhand, if we use that same $Algm^*$ & use CB = $\frac{T_0}{10}$, we will often do rather poorly, because many of t. predn methods will actually require ~ CB = T_0 in order to be workable.

.18

.19 What we could do: Have a set of $Algm^*$'s: $Algm^*_m$ is for CB betw $T_0 2^m$ and $T_0 \cdot 2^{m+1}$ (or $T_0 \cdot 10^m$ & $T_0\, 10^{m+1}$). Each time we run a search, we use t. appropriate $Algm^*_m$ to obtain pc's of $Algm_i$'s. Also, when we run a search by doubling T_0 progressively to obtain larger CB's, we credit t. proper $Algm^*_m$ for each "best" soln. obtaind within that CB. — So if we hunt at T_0, then $2T_0$, then $4T_0$ then $8T_0$ ... etc. at each search we credit t. proper $Algm^*_m$. m = 1, 2, 3, etc.

We should look at how $Algm^*$ varies w. CB, to get some idea of how much spacing to use betw CB's (i.e. t. no. α in $T_0 \cdot \alpha^m$). Also whether we want to run a search w. CB = T under t. $Algm^*_m$ for $\alpha^n < T \leq \alpha^{n+1}$, under m = n or under m = n+1 (.07 — .18 is perhaps a beginning of an analysis of this problem.) If CB = T > any used befor, we → 47.15

.32.

.33 There are at least 2 aspects of this: ① variation of corpus length — which can, to some extent, be dealt w. by 40.25 More generally, hvr, various corpus lengths will to some extent modify t. pc of t. $Algm_i$'s assoc. w. a gn. C.B.

② Variation of C.B.: changes t. predn. techniques available: ∴ changes t. pc of $Algm_i$'s a great deal. (.19 — .32 can deal w. this effectively).

Just run it under t. $Algm^*_m$ of largest available m (or large?). Note that SSZ may be small for $Algm^*_m$ of small m, — so we may have to mix data of lower m values somehow ? (see ...)

1.31.79 Lev:

40 pp, 25 d.

46.33 – .40 Discusses 2 ways in which a sc may be inappropriate for a Algm^*, & tells 2 ways to try to fix things. ~~It also shows how~~

Another way Algm^*'s may be inappropriate: The sc. may be very difrnt from previous sc's: T. only ways to deal w. this I can think of

(1) avoid t. situation by using a suitable tng. seq. leading up to t. otherwise "far-out" sc.

(2) use a very large CB.

I guess (2) is more desireable if possl., since by designing tng. sequs., we are biasing TM's education w. our own ~~prejudices~~ prejudices.

.15 : Re: 46.19 — .32 If we assign each Algm^*_m to a range of CB's, (say T_0 to $10T_0$) then credit for "finds" will have to be made for that Range of CB's. ~~Alternatively~~ I guess we don't want to credit say, Algm^*_j with all good Algm_i's found for $CB \leqslant T_0 \times 10^7$; Algm^*_j gets credit for only those Algm_i "finds" w. CB betw., say $T_0 \times 10^7$ & $T_0 \times 10^8$.

This brings up t. Q. of SSZ. Certain Algm^*_m's will get very few data pts. & so t. low SSZ will give their pc evalns little accuracy: We must find a good way that effectively pools data ———

.24 so pc of a particular Algm_i becomes a function of both q_i and allowable CB .... which sounds peculiar (not so peculiar!) — but it would seem to be t. ~~only way~~ most general (& best) way to pool data for various Algm^*_m ranges.

So perhaps we want a super $\overrightarrow{\mathrm{Algm}^*}$ that expresses pc's of Algm_i's as a funct. of allowd CB, on t. basis of all available data on all Algm_i scores.

Perhaps we want $\overline{\mathrm{Algm}^*}$ to predict expected "score" (≡ pc of sc wrt Algm_i) ~~as a funct of~~ and cc of any arby Algm_i. It would do this by giving a joint proby distribn. for cc & "score" for each possl. Algm_i. As a function of SC length also

This "score" idea is impt., because in animals ~~human~~ that have problems that are not purely predn. probs, this "score" concept can be far more general (& impt.) than simply pc of a sc wrt a proposd Algm_i.

Also include length of sc here)

See OSTM 4.12 ff (to at least ~5.40) for reasonable discussn.

Dropping these refinements for a while! Let us consider a TM w. sc's of all t. same length, & t. same CB for all of them. Could I get some interesting, (perhaps even useful!) behavior from such a TM?

I am thinking of a machine to do tag. seqs like t. Chicago 1962 paper: Like 40.03–17: One / single Algm. that characterizes TM's behavior at any particular time. It uses this on sc_{i+1}. If it doesn't work, it does modifns. of $Algm_i$. What is t. "modification metric"? — well, it is mutation operators that have been useful in t. past. A possl. refinement of keeping only one $Algm_i$, would be to retain, say 100 of t. best Algm's thus far.

Another impt thing: When we modify $Algm_i$ to work sc_{i+1}, we would like t. modifn. to be able to work all previous sc's. If not, t. new $Algm_{i+1}$ is not necly much better as a starting pt. for sc_{i+2} than $Algm_i$ is.

— This fact make t. older approach of Rev (Lev.) 1.01 – 4.40 seem better — i.e. the info about t. entire tag. seq. is stored in Algm*.

[SN Algm* should look on t. sequence of $Algm_i$'s it is extrapolating as an ordered seq. .... so Algm* could not be simply a stoch gramm. — since Gramms. give probys for unorderd sets of objects.]

One reasonable aproach: Devise a tag. seq. that seems reasonable for a ≃ human. Then devise a TM to solve this Tag. Seq. The idea is, that if a tag. seq. is reasonable — i.e. if a human can work it, then I should be able to devise a machine & search strategy that can work t. tag. seq. in reasonable CB. probly using methods not too far from Human.

Here, I would start out w. some idea of what TM looks like & what t. tag. seq. looks like. T detaild construction of both TM & t. tag. seq. would be developed by me, in parallel — by a process of ll growth.

On Human Problem Solving: T. way it is usually done! Say I've solved various probs. in t. past — each w. a difrnt algm. I get a new problem. I first try to see if it is "much like" one of t. previous probs. solvd. If so, I use t. correspg. formerly successful algm. for a first trial. If it doesn't work, I try modifying that algm. in a way that depends on both t. new problem (& possibly on t. old problem that t. Algm successfully solvd) & upon t. Algm. itself. → 49.15 (spec) ...

2.2.79 **Lev:**

I <u>do</u> have a lot of work on Tng. Seqs.: Mainly in Math learning.

So t. work consists of devising a tng. seq.; of making a set of primitive abss. w. assoc. pc's, & estimates of their cc's. From this, I can use $\frac{\Sigma cc}{\Sigma pc}$ $\frac{cc}{pc}$ to get an upper bnd of search cc for each step in t. tng. seq.

—— Actually implementing t. machine is not absly necy. — <u>much</u> useful work can be done w.o. pgmg. <u>anything</u>!

Pgmg. would give Σcc's much < t. upper bnds (presumably), so we mite be able to work tng. seqs. w. larger heur. jumps than one would think by just "hand analysis" of t. tng. seqs.

Also, t. possy. of serious errors in Hand analysis can be reduced by pgmg.
Also, It is likely that one will derv. "<u>creative</u> new solns." by pgming.

.15: 48.40: Usual way to solve problems: I look at t. problem & from initial observations, decide that a certain algm. that I've used successfully in t. past, is relevant. I apply t. Algm. & it works.

Alternatively, I decide that previous algms are usd are not applicable, or I can't remember any of t. old algms that are certainly applicable. So I try to derise a new algm.

Usually I have this genl. technique of looking at probs. & deciding which algms are applicable. <u>This</u> is a partial rec. funct. from probs. into "applicable algms."

So I <u>factor</u> t. genl. problem prob. solving algm into (a) A/selection algm, — that looks at probs & decides what $algm_2^i$ to use (b) a set of $algm_2^i$'s that are to be applied to a narrower range of probs than $algm_1$.

1000 10.000 10^7

If $algm_1$ has a "nullish" output (i.e. no $algm_2^i$'s seem applicable), then $algm_3$ is called — which devises <u>new</u> trial algms. T. new algms. all (perhaps) contain a recognition part to decide if t. algm. is applicable, and an Operator part that is usd <u>only if</u> t. (operator algm.) is applicable.

The nature of t. various tngg. algms. is somewhat vague & t. method of finding new trial algms is even <u>vaguer</u> — but this is just to be a <u>framework</u> to be <u>filld in</u> after an analysis of some <u>actual tng. seqs</u> that <u>humans can do</u>.

T. really <u>neat breakthru</u>, is that all I need to do is calculate $\frac{\Sigma cc}{\Sigma pc}$ $\frac{cc}{pc}$ for each step in t. Tng seq. Very Good for t. Proposal!

Another <u>very impt</u>. idea is that of trying lots of codes at once as in **Maxm**: — so we get <u>very large</u> $\frac{\Sigma cc}{\Sigma pc}$. Perhaps finding <u>one</u> code of a particular kind (I say "central") in this case makes it easy to find near by codes of hy pc "near by"

2.2.79 Lev

That Arithmetic → Alg → Calculus, etc. Tng. seq. looks attractive

I should read stuff I have about it on how to Genz. it to more diflt. probs: — But I suspect that that tng. seq. could lead to soln. of perhaps truely diflt. probs.

At any rate, it looks like an interesting "study problem".

Of INTEREST: That we may be able to get fairly good learning of tng. seqs. w. a very **simple Algm*** ~ say something like Bern seq. plus "**definitions**". **In such a case, most of t. real "work"** is done **in finding t. Algmi's.**

.18 Algmi's : Algm* seem to be difrnt. sorts of devices

Hours in Algmi's v.s. Hours in Algm*. How to present (info for learning Hours) to TM.

Algmi's assign probys (or induction codes) to sc. sequences.

Algm* generates Algmi's in ≈ order of aprip.

so: for Algmi's, we insert a string into t. machine; ~~its~~ t. output is t. probby of that inputstring (or a suitable good (set of) induction codes).

For Algm*, we insert an integer (N) & its output is t. Nth most likely Algmi.

.25 Optionally, Algm* could have a stochastic output; Its input is a random no.. Its output distribn. is that gen. by Algm*.

Because of this, it may not be easy to get t. Algmi's to try to work Algm*'s problems.

Is t. Cover — L. Cheung method of going from Algmi-type to Algm* (of .25) type; relevant?

Not directly: Algm* produces finite objects as opposed to.

A possl. soln: Consider t. sequence of "Good" Algmi's. We want to devise an Algm* that could have generated them w. hy pc_1; & have ~~this~~ t. derv. of this Algm* have hy pc_2 also (we want t. t. product, $pc_1 \times pc_2$ to be hy.). So here, we have an Algm*' (a special section of Algm*?) that generates Algms that are generators of ~~Alms~~ Algmi's.

Well that doesn't seem to help much — because how do we findout what ~~aprit~~ pc this Algm. would have assigned to t. corpus ~~[illegible]~~ of t. known set of Algmi's?

.37 Perhaps we want an Algm*' such that its input is t. set [Algmi], & its output is (a short code (or pc value) for t. [Algmi] set) and a new Algmi of hy pc. (or ~~[illegible]~~ an ordered seq. of Algms — in order of pc.)

52.01 spec.

2.3.79: (Lou)

Mathematics & Intuition: To what extent has t. devel. of Math been ~~[illegible]~~ influenced by t. Physics & t. characteristics of t. R.W. that Mathematicians Live in?

One view (Lou) is that Good Math is fairly free of influence of R.W. That there is a "natural developement" of Math that is indipendent of RW., & is that good Mathematicians can sense t. proper direction of this developement.

Another view (my own & prob'y Minsky's): That t. direction of developement of Math ~~[illegible]~~ would be fairly arby — except, that it is limited to those concepts that t. mind of the Mathematician finds "intuitively ~~[illegible]~~ acceptable" — they are concepts that he can deal with effectively thru his sub-concious mind. That the limitations of this sub concious mind condition t. Mathematician's concepts of Beauty, Unity, eligance.

The mathematician is, in principal capable of dealing with only 2 types of problems: (1) Non-creative problems, requiring few new **concepts** -- developements of old ideas

(2) **New beautiful Concepts** needed; The only kind the Mathematician can think of are "beautiful, unifying ideas" — which are in principal conditioned by t. capabilities of his sub-concious mind.

This sub-concious mind has been trained to deal w. RW probs of certain kinds — & its ideas of Beauty are so conditioned.

28: 2.7.79: from discns. w. Lou, I got t. idea that he likes th. idea of "optimal," "universal" theorems. **In math it's** often possl. to take an object & find <u>some</u> criterion for which it's "optimum". However, in any ~~[illegible]~~ RW application, it's nacy that t. object used **be optimum** for t. <u>correct</u> <u>criterion</u>. ~~[illegible]~~ Perhaps people use things that are optimum for one criterion as a choice for situations in which the optimality criterion is <u>quite</u> difnt.

The idea of 56.02 (that t. $\frac{pr}{cc}$ ordering is optimum), of their being a single best UMC for all math, of t. use of R because it is optimal, & <u>not</u> R' because there <u>is</u> no optimal normed scpn. — these are all in t. same direction. L asks "If you were world dictator, how would you tell scientists how to work?" — i.e. again t. idea that there is an optimal method. → 55.30

2^{3k}

$k = \lg N$

N^3

Random Notes!

1) Levin notes that ≈ there is no Universal normed semi computable proby measure.

I think a more exact statement is, that of t. Univl. normalz ~~...~~ proby measures (that are limits of normed cpm's), there is none that is ≥ all others within

.05 a const. factor. This means that if P_1 & P_2 are 2 UPMS, then ~~...~~

.06 for any constant, $c > 0$, no matter how large, $\exists\, x(n) \ni P_1(x(n)) \geq c\, P_2(x(n))$.

If both P_1 & P_2 are derived from Univl. scpm's can this be true?

If R_1 & R_2 are 2 univl scpm from which P_1 & P_2 are derived, then $\exists$ a constant > 0 α ~~...~~ $\ni$ for all $x(n)$ $\alpha R_1(x(n)) > R_2(x(n))$ & $\alpha R_2(x(n)) > R_1(x(n))$; ~~...~~ so $R_1(x(n))$ & $R_2(x(n))$ are within a const

.15 factor of one another: $\frac{1}{\alpha} < \frac{R_1}{R_2} < \alpha$

Still, hvr, P_1 & P_2 can differ by very much, since t. normalization factors can be unbounded (arbly large).

Well, maybe not: In fact t. normzn. constants must be bndd! — since a > 0 fraction of all infinite strings do have probys > 0 — e.g. any recursive infinite string has a proby > 0.

.20 Hvr., I'm not sure this argt. is correct: it may be based on my earlier normzn idea — which was incorrect — i.e. $P_1(x(n)) = R_1(x(n)) \Big/ \sum_{i=1}^{2^n} R_1(x_i(n))$ ($x_i(n)$ is t. ith possl. string of length n.)

See .30 for t. rest of this argt.!

It would seem that if t. normzn factors are bnd., then .15 implies

$$\frac{1}{k\alpha} < \frac{P_1}{P_2} < k\alpha$$

: where k is t. largest value t. normzn. constant can have. (This const is always ≥ 1.)

This implies .05 - .06 must be false!

Well; I suspect that t. error is that of ≈.20: that t. correct normzn. const. ~~...~~ is, indeed unbndd, for some x. This is an impt. result (if true): I had infnt. string.

.27 previously that ~~...~~ that t. normzn. const. was ~~...~~ bndd for all x.

See 2.25 — ~~...~~ 3.03; 6.01 – 6.09; 2.39 suggests that t. normzn. const. must be bounded for a set of strings of total measure 1. — but (among other things) what proby measure is this "measure 1" wrt?

.30 T. erroneous normzn of .20 would make t. normzn. constant bnd., because t. const. was a funct. of n only (i.e. t. same for all strings of length n). If t. constant $c_n \to \infty$, this means that a fraction $\frac{1}{c_n}$ of all infnt. strings converged (i.e. had measure > 0) — so t. measure of all strings that converged was zero — which is impossl., since any recursive string ~~...~~ has measure > 0.

These results suggest that contrary to my earlier conjectures, even for large n,

$$\frac{R(x(n)\frown 0) + R(x(n)\frown 1)}{R(x(n))}$$

may differ appreciably from 1. — i.e. it need not → 1 as $n \to \infty$. I'm not sure if this is true for x's of measure > 0 or just what, hvr. — but for at least one x, t. product of these things must be zero (i.e. $\prod_n$ ≈ of). — so t. sum of t. deviations from 1 ~~...~~ must diverge for at least one x.

55.09
58.15

.01: (50.40 spec)! 50.37 is not bad: For t. sequence of Algms to be a suitable training set for Algm*', t. problems of t. Algms's should be not only assignment of pc (or short codes) to sc's, but also some kind of extrapolation like Algm*' has to do. We then have to use $\frac{\text{cc of evaln, of sct cc of creation of prodn}}{\text{pc of Algms}}$ as an ordering criterion

.05 for all of t. Algms as well as Algm*' trials.

Hvr., In general, I think t. best (certainly for long range) approach is ≈ PMTM. T. various (cross-coupled) modes assure that all manners of heurs are usable. T. problem of Algms & Algm* (50.18ff) being difrnt kinds of machines would prosbly be easy to solve this way: its just "another kind of problem", & PMTM has ways to deal w.it.

I have written a lot out development of a very general TM — perhaps GIM ≡ General Induction Machine — & on development of its tng. seqs & heurs.

At one time in particular I was very Manic about t. capabilities of t. system & I felt I had it pretty well "MADE"!

Go back to some of this earlier stuff. It may be v. G.

I think t. present search techniques may be adequate!!

But t. General idea of a very general purpose device to solve all kinds of probs. in ≈ t. same way, seems V. G.

I told L. that I didn't see why his ~~proof of~~ thrm. applied only to t. system of Kolmog–Uspenskii Algms. He said that in that system something was $\propto N$: In other systems $\propto N + \lg N$: I will ask for more details on this: See 28.10–.29 (& perhaps 22.01–23.20 for Backgnd) see 57.10 for kind of explainn.

$\Sigma \quad x\, n.$

~~term~~

$n = n+1$

$\text{terms} \frac{\text{term}}{n}$

$\Sigma = \Sigma + \text{term}$

$\left(\left(x+\frac{1}{2}\right)x+\frac{1}{3}\right)x+\frac{1}{4}$

$(((x+1)x+2)x+3)\ldots$

$(((x+1)2x+1)3x+1)$

$((x+1)\frac{x}{2}+1)\frac{x}{3}+1$

Preliny dcrn. of elementary Induction Machine:

string of sc's; $sc_0, sc_1, sc_2, sc_3 \ldots$

We want to devise algms $U(\underset{\text{unv}}{q_j}, sc_i)$ w. 2 outputs:

a) short codes a/o proby of sc_i

b) a set of continuations of sc_i that are of hy pc.

~~{sc_i} can~~ The sc_i can be sets of finite objects to be extrapolated, a/o strings to be extrapolated.

We want a algm (dcrbd by q_j), $\ni$ $\underline{2^{-q_j} \cdot pc_{ij}}$ is max:

pc_{ij} is t. proby assigned to sc_i by q_j.

We will try th. q_j in order of

$$2^{q_j}\, T_{(\text{to generate } U(q_j \cdot) \text{ \& calculate } pc_{ij} \text{ \& generate t. predns})}$$

\- smallest, first.

This search technqe gives us best algms for gn total time of search.

.20

.21 After doing a few initial sc_i's we modify U so that it expresses the associated best found algms, with shorter codes than before, & we continue t. search.

Periodically U is modified so as to make t. codeg of t. v.g. Algms thus far, as short as possl.

.26

.27 Another way to do .21 - .26: After solving a small no. (say n) of initial sc_i's, we look at $q_1, q_2, q_3 \ldots q_n$ & we try to extrapolate it. This gives an apriPd on q_{n+1}. We use a rather

.30 simple algm to get t. q_{n+1} distribn — using freqs of symbols & perhaps

.31 defns of ngms. This proby distribn on q_{n+1} is used to control t. subsequent search, $(P(q_{n+1}))^{-1} \times T_{(\text{gen, calc, gen})}$ being used for ordering.

As we solve more sc_i's we base $P(q_{m+1})$ on a larger body of data — so t. probys would be better if t. proby model were adequate. Hvr., even w. t. simple model of .30-.31 I think we would be able to follow reasonable trg. seqns.

We can, of course, use arbly complex algm. for basing $P(m+1)$ on t. seq. $q_0, q_1, \ldots\ldots q_m$.

After suitable trg. w. suitable kinds of problems, we ~~mitr~~ let $q_0\ q_1 \ldots q_m$ be one of t. sci's & allow t. regular prediction scheme work out $p(q_{m+1})$ & make trial selections for q_{m+1}, $(TM_2 = TM_1)$ since TM_1 was designed to be able to do both of these.

---

1) Description of initial trg. seq. say with learning: Then ~~[illegible]~~, neg. nos., then fractions, Then solns. of linear eqns., solns. of simult linear eqns. Then sq. roots (radicals)

$1+3=4$; $5-3=2$; $5\times7=35$, etc.

2) Design of initial UMC, U; First, partial design of Trg. Seq., Then devise instruction set for computer so as to minz. "total instruction length" of solutions. Note that "Total instruction length" is not usually found by length of instructions, but by probability of ~~[illegible]~~ the solutions. So we want a Probty measure that assigns hy probabilitys to the set of solutions that we have for t. trg. seq.

simple qual eqns
general quad eqns,
cubic, quartic, etc.
complex nos.
Trigonometry
dif. Calc.
Int. Calc.
try to get it to solve diff. probs. as soon as possl. — since there are diff probs even in simple Algebra: e.g. Math Tripos

.20 So: we write "reasonable" trg. seq. Then we write set of solns. to t. seq. (>1 soln. to each sc. if possl.). This is done in any good language; Say APL, with suitable "extensibility additions" if necy. Then we look at these solutions & try to design a new instruction set that

a) Expresses t. solns. w. short codes
b) Will be likely to express solns. of future problems w. short codes.

.28

Note: I suspect that even w.r.t. $TM_1 = TM_2$ concern. This device may not have all human hears available to it. PMTM may be necy — also other kinds (?!) of info input.

After some experience with t. problem .20-.28 one may also be able to work on problem of designing a v.g. Umc for Soln. of NP probs.

2·6·79 LBVS

Dison. w. L.: Big Arguement w. L.: R vs. R_n (call this $\bar{R}$):
L. feels that R is ~~correct~~ t. optimum thing. : That he is not really open to discussion:
That even if I did have a demo. to show $\bar{R}$ was better, he would simply try to
show t. demo. was n.g.

Cat 220 44110

Jan 27

His reasons for this: ① R is optimum within a constant factor, but
there is no such optimum Univl. SGPM.

·09:51.40 ② $\bar{R} \div R$ is very small any way; something like: t. proby that $\frac{\bar{R}}{R} > k$ is $< \frac{1}{k}$.
·10: — Tho I'm not sure it's just that. He says it's "an easy proof". → 58.15

③ say α is an n bit random seq. Then
t. proby that $R(X/\alpha)$ is $> R(X)$ is somewhat similar —
So he feels that $\bar{R}$'s being $> R$ is of no more signifc. than $R(X/\alpha)$ being $> R(X)$.
[Seems that $R(X/\alpha)$ has an extra param: t. no. of bits in α — which
·18 doesn't occur in $\bar{R}$] → 60.25

④ I mentioned that $\bar{R}$ used the [illegible] known info. that t. seq. would not halt. That such info. often was available — as when we actually knew (after it had occured) that t. seq. did not stop.

⑤ L. feels that absolute probability is of importance.

⑥ In t. case of t. Betting problem, he formulates it in such a way that R_T is t. only soln. I.e. t. betting funct. must by defn. be a non ↓ funct. of T. When I told him that $\bar{R}$ was always $> R$ he asked for a proof — which I sort of supplied, & he admitted it was true, but said that even if $\bar{R}$ was $> R$, he could always find a R' that was better than my $\bar{R}$! These arguments seem rather not to the point.

. 30 : 50½.28 → My impressn. of L. is that he is fascinated with the ideas of absolute truth — with finding "unique" "best" answers to Q's. He feels that the law of conservation of energy hasn't been changed for > 100 yrs. — that t. 2nd law of thermo has also remained unchanged since it was proposed.
That all impt. Univl. constants are "small".

By "Library" I think he may mean α in $R(X/\alpha)$ . . .
i.e. α, t. "library" is aux. info involved in computing "$R(X)$".

L. asks if folg. is true: we have this Algm ordering trials by $\frac{pc}{cc}$.

Does there exist an Algm A (probly) ∋ for all Algms B,

.02 $t_A \leq C_A t_B + d_{BA}$?

Another way to ask this: is $\limsup_x \frac{t_A(x)}{t_B(x)} = \infty$? and above ?

Here t_A is t. total time needed to solve a NP prob. by Algm A

" t_B " . . . . B

C_A is a constant, characteristic of A. (L. thinks C_A should be "small").

d_{BA} is a const. char. of A & B.

L. thinks maybe there is, but it would be useful to prove yes or No.

Proby we can show that if t. order of trials depends only on q_i & t_i, then A is optimum (the .02 is for all NP probs. indep of size of N, constancy of C_A means that t. "indep of B" is concerned only w. large values of "N". (essentially $N=\infty$)).

L. feels that this Q. is very relevant to t. problem of optimum search —— that he has considered things like tng. seqns., but he'd really feel that we would be on firmer ground, if we knew .01 - .01 - .02 were true or false.

⟶ 58.25

.23 He is interested in t. Q of ~~[illegible]~~ in $\frac{pc}{cc}$ search, whether we usually end up with a final $q \gg \lg T$ or $\lg T \gg q$.

I think he's talking about NP problems. Is one much more frequent than the other? Also under what conds is it possl. to ↑q a bit & ↓T very much?) see 58.01

L. has been getting involvd w. Schransky case: In Oct 78 CACM was letter saying Schransky was no special big deal — so L. is going to talk to Peter Elias (who has been involvd w. such things) about it.

L. also says a letter explaining how Anti Semitism in USSR world was partially printed in Science ← (I think this is what he referd to) but in vastly attenuated form

---

L. Got subs. to Science: is disappointed: too much Biochemy.

He asked what were good Mags. I suggested Sci. Amer. Perhaps suggest he read "The New Scientist" also. New York Times --- of some interest. WSJ is worth glancing at occasionally.

2·6·79: Lev.

Re: In his Thrm. solving all NP probs: He mentions Kolmog, Uspenski algms. This amounts to a kind of machine that has much better mobility of info than a 1 tape Trmc.. T. Kol. machine is an expanding network of nodes w. edges that can be arbly connected & disconnected. He says it amounts to a RAM, but one in which access time is not ∝ lg No. of addresses.

.10 Anyway, T. reason his thrm didn't work directly for Trmcs, is that a Trmc. needed an extra tape to count time, to limit t. algms running time. If we consider such 2 tape machines, then they would

.15 need another extra tape to time them, & so on for more complex trmcs.

T. Kol. machine does not have this diffy.

.17 Anyway, I'm not sure I understand .10 – .15 but.... → 61.06

Also some Russian Guy has invented a theoretical ‖ machine that can simulate any other ‖ machine within lg T & lg N (or $(\lg N)^2$) T in time, N in space. L. says he had devised a much simplified version of this, but didn't publish it ("noise pollution of Journals").

T. Kolmog-Uspenski algm (or machine) has a non-dimensional network topology, rather than a 1 dim. tape. It can create or destroy "edges" betw. arby nodes! A RAM is something like this, but has access time ∝ Log N — T. Kol. machine doesn't have this limitation

Importantly Refs t. Comp. Comply.

I have { → 1) Blum JACM (1967) 322–336

→ 2) Rabin (1960)

3) Hartmanis & Stearns: Trans Amer Math Soc 117 (1965) 285–306

L. says Blum's speed up thrm. shows that for a great no. of algms (involving infinitely long seqns), it is possl. to find algms speeding them up by an arby recursive function. L. says he has a ~~many~~ much more generalized version of this thrm. He says those speed up thrms (Both B's & L's) are for pretty arby forms of cc.

Hvr. if these "speed up thrms" are true, they must apply only to very long sequences — otherwise they would be used in computation. Hvr. they don't seem to have even made a dent in estimates of time needed to solve various probs., as functions of N. I'm not sure t. "speed up algms" all apply to all kinds of probs — only certain kinds.

L. has been refering to costs as "computing" or "dollar" costs (tho usually time) — perhaps I got thru to him on this point!

2 15 79 Lav:

L. consider T as being essentially dimensionless: ≡ t. no. of steps in t. compute. Are other forms of cc equally dimensionless?

— Anyway, this makes t. Q of whether ~~proof T is low~~ q or lg T is larger in 56.23 meaningful.

In computing $U(q, S_i)$, say, what is t. expected distribn. of computing times for, say $S_i \hat{0}$ & $S_i \hat{0}$? L. feels this is impt. Q. In general, I guess it has impt. bearing on using $\frac{pc}{cc}$ as [redacted] search ordering criterion.

.15 : 55.10 Anyway say t. probty that $\frac{\bar{R}}{R}$ is $> x$ is $\sim \frac{1}{x}$

so t. probty that $\frac{\bar{R}}{R} < x$ is $\sim 1 - \frac{1}{x}$

T. probty that $\ln \frac{\bar{R}}{R} < \ln x$ is $\sim 1 - \frac{1}{x}$.

T. expected value of $\ln \frac{\bar{R}}{R}$ is $\int_1^{\infty} \ln x \; d\left(1 - \frac{1}{x}\right) = \int_{1}^{\infty} \frac{\ln x}{x^2} dx = 1.$

$= 1$ so $\bar{R}$ is about e times as large as R.

$y \equiv \ln x$
$x = e^y$
$dy = \frac{dx}{x}$
$= \int_0^{\infty} \frac{y}{e^y} dy$
$\int_0^{\infty} x e^{-x} dy = 1.$

This idea seems impt., hvr.! It gives us some idea as to how much time will be spent on unconvergent trials. Hvr, this must be gone into — try to get L's proof so I really understand what's going on! → 67.25

In comparing $\bar{R}$ & R, I think this should be done for larger values of n (of x(n)). This is because x spends most of its time "Being Large"

If this is a proof, then t. whole thing is silly!

67.19 looks like a proof!

.24

---

1 AM start of Noise.

.25 Re: [redacted] L's conjecture of 56.02:

How big is α — Get some examples — some orders of magnitude — some limits, some bounds

Consider linear regressn., d coefs! N is length of corpus!

$q \propto d \lg N + \gamma$ so $\underline{\underline{pc}} = 2^{-\alpha d \lg N} = [redacted] \cdot N^{-\alpha d} \cdot 2^{[redacted]\gamma}$

$\approx \alpha d \lg N + \gamma$

constant gives structure of linear regn.

Time needed to compute t. pc of t. corpus wrt. t. algm is probly $\propto$

$d \cdot N. \quad \approx \beta d N.$ so $\frac{cc}{pc} = 2^{\gamma} \beta d N \cdot N^{\alpha d} = 2^{\gamma} \beta d N^{\alpha d + 1} = \boxed{\delta \cdot d N^{\alpha d + 1}}$

= Upper bnd.
= Approx. time for search using L's method.

Using ~~[illegible]~~ usual correlation matrix method, hvr!

.35 Time needed to do d lags for correln. matrix $\propto N \cdot d$ (No! Accuracy $\propto \lg \sqrt{N}$ is needed: say $\left(\frac{1}{2}\lg N\right)^2$ operations needed for 1 $\lg\sqrt{N}$ digit multiplcn. so $\underline{d \cdot N \cdot (\lg N)^2}$ may be more accurate. → see 59.15

Time needed to solve correln. matrix using t. best known way: $\propto$ d or d lg d or d^2 mult by no. places. Also we need accuracy $\propto \sqrt{N}$ ∴ $\frac{1}{2} \lg N$ digits so soln. time is maybe $d^2 \cdot \left(\frac{1}{2} \lg N\right)^2$ for t. equations. So Nd usually dominates

.40 Compare $N d \lambda$ w. $(\lg N)^2$ $\delta \cdot d N^{\alpha d + 1}$; ratio is $\propto (\lg N)^2 : N^{\alpha d}$

2·7·73 Lev:

so in this case, it would seem that t. $\frac{cc}{pc}$ search method is worse by a factor that is $\approx$ a polynomial in N.

Note, hvr., that linear Regression is not an N.P. problem — [The we can easily
.04 devise a N.P. prob. from it. E.g. "Find predn. code with < 2.3 MS
.05 error" or various other error criteria can be devised. In general, there will be $\gg$ one soln — which reduces t. search time somewhat: we want to find t. soln. of $\approx$ ~~lowest~~ largest $\frac{pc}{cc}$ — within a factor of 2, say.
— This problem does have to be examined more closely]

.15: On t. Q of Accuracy needed in various arith. operations.
T. corpus can be generated in several possl. difnt ways:
.17 1) a) Infinite accuracy usd to produce a T.S. using d coifs & some varc, σ^2.
b) This seq. is then truncated to m places of accuracy.
.19 2) The truncation can occur in various parts of t. generation of t. T.S.
e.g. d initial values are taken, w. m places accuracy. X (d+1) is computed
.21 using d exact coifs & exact σ^2; t. result is then truncated to m places.
I think I found a particular method of truncation that was particularly tractable for CBI analysis. .19–.21 may have been this way.
Hvr., I don't know if .19–.21 is easy to analyse w. correln. matrix, etc. —
— T. method of .17 may be easier to analyse this way.

I suspect that t. result of 58.40 is in t. rite direction — that t. ratio will be $\approx$ 1: $N^{\alpha d}$ even after I look at it more carefully. I really would have to do a detaild analysis of linear regn. (which I have probly already done — but don't know exactly where it is) to be certain of this result.

[Perhaps t. problem of finding t. mean & varc. of a seq. of indept. gaussian ordinates would be a good counter example to t. conjecture of 56.02. The independence of t. ordinates makes truncation easier to deal w.

.30 (To get t. mean & varc by conventional means; N squares + N additions.
Say $\frac{1}{2}\lg N$ digits of accuracy so Time: $\propto N(\lg N)^2 + \beta N \lg N$

.32 using $\frac{pc}{cc}$ search: $q = \alpha' \lg N + \beta' \lg N$ for mean, varc.

$\tilde{T}$ $\propto N$ operations.
(to compute error of a particular mean, σ^2 selection)

so $T \cdot 2^q = N^{\alpha'+\beta'} \cdot \alpha N (\lg N)^2 + \beta N \lg N$. $T(\;)$ probly $= \alpha N (\lg N)^2 + \beta N \lg N$

So $\frac{pc}{cc}$ search time is greater by factor of $N^{\alpha'+\beta'}$. If T_0 is t. time to compute mean & varc. via .30, then it takes 2^q times as long to do it via .32, and q increases w. N. $2^q \sim N^{(\alpha'+\beta')}$: $\alpha'+\beta' > 0$.

Note that if we make lin. regn. an N.P. Prob. — using t. methods of $\approx$ 59.~~04~~ .05,
The accuracy needed will be stipulated when t. problem is given.

.03 One example: Given N params x_i [$i=1|N$] and given A.

To find a real y [$0 \le y \le 1$] $\ni$ $\frac{1}{N}\sum_{i=1}^{N}(x_i - y)^2 < A$. | i.e. $\sum (x_i - y)^2 < AN$.

If it takes time To to compute $\sum (x_i - y)^2$.

say ~~that x_i are all between or~~ $\forall i$, $0 \le x_i \le 1$ and we have m bits accuracy for each.

Then, it takes N m bits subtractions; N m×m mults, & N 2m bit additions

to find $\sum (x_i - y)^2$. so $\sim 3mN + \sim m^2 N \;.\equiv T_0$

The accuracy needed for y is at most, set by $\approx \sqrt{N}\cdot\sqrt{A}$. $= \sqrt{NA}$.

In such a case we need y to accuracy $\sqrt{NA}$ and $2^q = \sqrt{NA}$.

Total search time = "2^q" $\times T_0$ = $\sqrt{NA} \cdot \underbrace{N(\sim 3m + \sim m^2)}_{T_0}$.

T. time needed to find mean & var. is about T_0

.18 so $\frac{pc}{cc}$ search takes $\sqrt{NA}$ times as long.

T. example of .03 — .18 would seem to disprove L's conjecture of 56.02.
Hvr. that example, is for poly time. Perhaps t. conjecture is true if t. NP soln. is exp. time .... but that a NP problem is exp. time is usually very hard to prove. (usually it hasn't been done).

L. mentioned int. NP soln. paper that there are difft NP probs. in which it is not known that soln time is appreciably > verifn. time: E.g. finding proofs in logic v.s. verifying proofs.

.24 → 61.10

.25: 55.18: On $R(x/\alpha)$: L. said that this was $> R(x)$ w. t. same probly that $\bar{R}(x)$ is $> R(x)$ (α being a finite random no.) I think not: $R(x/\alpha)$ is not only larger than $R(x)$ when $x = \alpha$, it is smaller than $R(x)$ when $x \neq \alpha$. On t. average, they are probly t. same! (Tho I'm not sure whether an arith or Geom. mean should be taken.)

He also said that if $\bar{R}(x) > R(x)$ then that he could easily find another R' $\ni$ $R'(x) > \bar{R}(x)$. I doubt if this is true

In fact, it must be false — No R can be > a normed proby measure — since R is sub-additive

i.e. he would probly have lots of trouble finding such a R' ...

On t. other hand, t. normzn of R is not altogether trivial —
It uses a fair amt. of compn. — to find each $R(x(i)\bar{\ }0)$ and $R(x(i)\bar{\ }1)$) — which is not need if R alone is used in betting. T. cc of $\bar{R}$ betting is about twice that of R betting ~~NO~~ on second thot — I think not they are about t. same. for R betting, both $R(x(i)\bar{\ }0)$ & $R(x(i)\bar{\ }1)$ need be known.)
So with a very small amt. of extra computation, I can (usually) get ~ e times as much betting yeald.

2·7·79 Lau. 61

If L. could find another $R' > \bar{R}$, then I would get another factor of e by using $\bar{R'}$; — clearly this cannot go on indefinitely. At some pt. L. will be unable to find a $R^{\cdots\cdots n+1} >$ my $\bar{R}^{\cdots\cdots n}$.

.05 If L. can do this even once, it should be done to ↑ betting yield. → 65.01

.06 : 57.17 One impt. implication of this is that for studying computational complexity, ordinary Tms are not t. ideal device, because it is not always possl. to get them to simulate other machines w.r.t. computation time — or "cc".

.10 : 60.24 : Other NP problems in which it is easy to disprove 56.02

Solve $|Ax - B| < 2^{-N}$ • i.e. A & B are binary integers of N bits, $A > B$; we want a binary fraction $x \ni$ this is true.

To do long division / to N bit accuracy requires maybe N^3 operations.

To find x by $\frac{pc}{cc}$ search involve $2^9 \approx 2^N$, which is $>> N^3$.

Also other probs. in solving equs. in algebra — or transcendental equs, etc. $x e^{-xy} = 1$

Hvr. see 63.10 for error

Perhaps I'm missing t. pt. ! i.e. maybe we only look for / a single, finite q that solves t. problem for all N. This seems "more likely", but not so easy to state!

In fact, t. real sense of t. 56.02 is that $\frac{cc}{pc}$ search is near optimal for all problems, if one looks at it rite! Usually one solves such problems using what appear to be non-random "heuristic methods" — but these methods are prob'ly really ordinary random or exhaustive $\frac{cc}{pc}$ searches using a suitable tug. seq.

Hvr., it would seem that this would be dif'f to prove as a Thm., because of t. critical dependence on "Library content" (= previous tug. seq.) — Maybe not i.e. see .35–.37

L. had a particular way of stating t. conjecture in terms of $\limsup \frac{t_{A(x)}}{t_{B(x)}}$ being bnd above by a rather small no. — like, say, 2 (see 56.02 for notation).

I think $\limsup f(x)$ is t. ~~smallest~~ largest value, A, $\ni A - f(x)$ is < arbly small ∈ for infinitely large no. of values of x. ~~[illegible]~~

I don't know if x has to have ~~[illegible]~~ countable no. of values or whether it may be continuous.

.35 Anyway L.'s conjecture may be about t. same as my conjecture — (i.e. t. "R.S" conjecture). that $\frac{cc}{pc}$ search may be optimum for induction, even w.o. tug. seq., but that it (or any other method) is not / ordinarily practical unless a suitable tug.

.37 Seq. is usd. — ≶

1:30 PM

6:15 PM

(216) 531-9771 Mrs. [illegible]

Room 220

Beach Manor

16101 Lk shr. Blvd

Clev 44110.

2.10.79 Lev:

I suspect that what L. has in mind in 56.02 is that $\frac{dc}{pc}$ search is optimum w.r.t. to info. available for that search. This includes any heur. info obtaind by previous solns. of "related" probs. It would seem that this is quite diflt to formulate exactly: i.e. by suitable selection of an Umc., any
.05 problem can have a short soln (i.e. small(q)) & sometimes (time to solve) may also be small by suitably selecting Umc.

Well, Look at it this way: Say t. Umc is fixd — thereby fixing apri info.

Another (perhaps identical) approach: Try to derb. "info. content" of an NP problem: Start w. "Zero" apri info. [Tho this may be impossl.: one may always have a Umc. around or some (not nécly unvl.) machn to define things, in which case one has an apriPd defined by that umc.]

In one case say one has to solve $AX = B$ within $\pm\epsilon$. Say X is known to be on $(0,1)$. One could try binary sequences for "x" trials into a black box with "Yes – No" output. One mite also have a timer to measure decision time of Black Box.

More exactly, for N.P. probs, one is gn. A, B. — or, a set of A, B pairs; & one must find a single $q \ni U(q, A, B) = y$; & $A(y, A, B) = 0$.

or, $q(A,B) = y \ni A(y, A, B) = 0$. i.e. McCarthy's "Inversion of functs. defnd by Trmcs".

Actually, we are given $A(\cdot,\cdot,\cdot)$ Funct. form: we have to find $q \ni$ for all permissable A, B, $y = U(q, A, B)$ satisfies

Using L's defn: gn. x, $A(\cdot,\cdot)$; to find $y \ni A(x,y) = 0$.

constraints: Time to compute $A(x,y)$ is poly in $|x|$; $|y|$ is "poly" in $|x|$.

.25 L's soln.: Try $y = U(q, x)$ for q's in order of $[T(U(q,x)) + T(A(x,y))] \times 2^{|q|}$.

If $y = U(q_0, x)$ is a soln, then it will take $< 2 \times [$ " q_0 $] \cdot 2^{|q_0|}$ "seconds", total $= (\text{Time to compute } y + \text{time to test } y) \times 2^{|q_0|}$.

One trouble here: L's soln. in .25 is for a given single x. T. NP problem is to find one q_0 for all x. Hvr., t. search he proposes does work for arby x. For each x, we go thru t. search again! still this gives a soln. for all x.

A short cut! we know $\exists$ an algm $q_0 \ni$ it's a soln. for $\forall x$. So if we are given some x_0, to find a suitable y, we start w. $x = 1$, then $x = 2$ then .... up to $x = x_0$. for each value of x we use, we can find a q that will solve it, but usually it will take less time because both $U(q, x)$ and $A(x,y)$ will take less time than for t. larger x.

.33 $\boxed{\text{If } T(U(q,x)) + T(A(x,y)) \text{ is monotone} \uparrow \text{in } |q|,}$ then we can try t. q that workd for $x-1$, on x; if it doesn't work, we just continue t. search with $\uparrow$ in q.

Hvr., while $T(U(q,x))$ may be monotone $\uparrow$ in $|q|$, $T(A(x,y))$ may not be. (tho this is also not so certain) (tho this may be << this But anyway, .33 may be a bottleneck in this shortcut that makes it usually inapplicable.

Hvr., even if t. bottleneck sometimes occurs, it may usually not be impt. — so t. "shortcut" may be usually good.

2.11.79 Lev:

I think I'm still looking at L's. conjecture of 56.02 incorrectly!

Say $|X|$ is large (which is what t. conjecture is about)

Then 2^{q_0} (T (generate y from x & q_0) + T (checking y)) may not be so large.

.08 No ... That's still not it. Perhaps t. Q is: can we make t. search factor $\ll 2^{q_0}$?

.09 Any device that could elim. certain trials would reduce this factor.

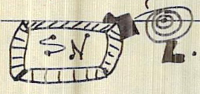
.10 Perhaps t. counter-examples of 16.10 (& previous examples) are wrong, because $\frac{cc}{pc}$ search would always obtain a q that was t. same for all A, B. This q describes t. "divide A by B" algorithm. T. counter examples all show that t. $\frac{cc}{pc}$ search is worse than optimum by a factor that ↑ w. $|X|$. — so they must miss t. pt.!

.14 I think .08-.09 is in t. right direction. Normally, t. algms of interest have, I think, a sufficiently long descrn so that $2^{|q|}$ would place t. soln. outside of human capability or $< 100 \times 10^{10} = 10^{12} = 2^{40}$ trials/sec. or $\approx 2^{60}$ trials/yr.

Humans use fag-seqs to reduce t. q's considerably. This is effectively done by partial F.B. as one makes trials..... essentially "subgoals".

T. effect of achieving these sub-goals is that t. umc is modified, so t. $|q|$ for t. algm. wrt t. latest (updated) umc, can be not very large. But t. search factor is always $2^{|q|}$.

.22 Can we show that if any of t. 2^{q_0} trials are ommited, we have some possy of ommiting t. optimum soln.? i.e. that t. only way to prove a value of q is N.G. is to try it?



L. repeatedly mentions $|q|$ as being more impt than T, since t. ordering is $|q| + \lg T$. This sounds like poor reasoning! Later, hvr., he was concerned w. whether t. final solns. had larger $\lg T$ or large $|q|$ .... so in spite of this "error", he seems to be able to do o.k. in at least one situation!

Well, This is probly true for some values of q — ... but is it true for most, or, more impt, "almost all"? Well even if it's true for as few as 10% of t. q's, we could only save by factor of 10, at most.

Perhaps, Then t. conjecture says that t. problem of finding q is exponential in $|q|$ & there is no shorter way. So far, hvr., few probs. have been proved to be exponential. It would be interesting to show that this problem is "NP complete"!

Well..... I don't think "NP complete" is relevant. It refers only to cases in which there is a set of probs of arbly increasing $|X|$, & concerns those w. very large $|X|$.

In t. present case $|q|$ is always finite & usually not very large.

2.12.79 LEV:::

Well, look at it this way: Given some N.P. prob., that I know t. soln. of, like

.02 $|A \cdot x - B| < 2^{-N}$; I can solve it quickly because I have this apri knowledge. Such apri knowledge can be designed into t. reference Umc.

If all apri knowl must be designed into t. ref. Umc., then perhaps by defn, exhaustive search is t. only way to solve t. problem. If any other means exists for a shorter soln., than one has, by defn., apri knowl., & there exists a way to modify t. ref. umc. by designing this apri knowl. into it.

.10 Hvr., there are other defns of "apri knowl" — are these t. same as t. forgo?

This is rather peculiar! An impt. Q seems: How can one be "free" of mathmcl knowledge?

Try t. define problem: Say I'm gn. some N.P. prob like .02 : $\left(\begin{array}{l} A, B \to X \\ X \to y \end{array}\right)$

I'm gn. a Umc, U' and X & I want to find a $y \ni A(X, y) = 0$

I try y's in order of complexity wrt X using $\frac{\epsilon c}{pc}$ for search.

.17 $U(q, x) = y$.

.18 Shouldn't y have low complexity w.r.t X & $A(\cdot,\cdot)$? —

i.e. $A(\cdot,\cdot)$ also contains impt. info. In t. simple $y = U(q, x)$ we don't use t. info in $A(\cdot,\cdot)$.

.20 Try $U(q, x, A(\cdot,\cdot)) = y$. Well; if $A(\cdot,\cdot)$ contains "no info" on t. structure of A : i.e. $A(\cdot,\cdot)$ is a "Black box", then, perhaps this will not help

For .20, let us do a search w. q of increasing length using (Time to compile + Time to test) $\times 2^{|q|}$ for search ordering, and .20 to generate y's.

25:68.24 SN Re: Proof that $\bar{R}/R$ > n has proby $< \frac{1}{n}$; Look at "Law of t. iterated log" see Cover's paper on "T. probability of a no. being rational"! | This is an impt concept. → see 67.19 for $\frac{\bar{R}}{R}$ proof

For suitable U, there is a $q = 1 \ni \underbrace{U(1, x, A(\cdot,\cdot))}_{A(x,y)=0} = y$ gives L's normal search for y ordering trials by $y = U'(q, x)$ & $2^{|q|} \times (T_1 + T_2)$.

T. ~~total~~ T for this $q=1$ is, hvr., t. same as t. total T for L's normal search, but there is only 1 trial y ~~isoftend~~ (for $q=1$) & that y fits!

} so this defn of t. prob. may give t. same soln. as L's defn of t. problem.

.30 .18 ff is interesting, but go back to .01 – .17: Try to state exactly what t. diffy is.

O.K.: I have a vague idea as to what "ignorance" is (total "lack of info", lack of intelligence) in trying to solve $A(x, y) = 0$.

It means I have a "standard" apripd, which is not very likely to be much related to solving this.

I think this apripd is $\approx P(y \mid X, A(\cdot,\cdot))$. or, related to complexity of y, given $x, A(\cdot,\cdot)$. "Info" = "Intelligence"

37 Next, to approach t. problem: what does "non-ignorance" look like? One form of "intelligence". A better way to look at X & $A(\cdot,\cdot)$ & derive an ordering of trials for y: In particular, using early trials to help form a better apripd for later trials. One way to do this (of unclear validity): Use earlier trials

} See 65.20 for a new tac on this!

$n\sqrt{2} \simeq int$

$n^2 \cdot 2 \simeq int^2$

$\frac{16^2}{11^2}$ $\frac{256}{121}$

7.071 ·
9.899 ·
16.97 ←
24.04
33.94 ·
41.012
57.9827
82.024 ←
98.9949
.995
140.007
.005
239.0017 ←
.998
337.9964
379.00849 hyssa reached
2^{39} ← prime
$\times 169$ = 1.4142011
12 = ... 135
$(\ )^2$ = 1.9999647
48 - .99

$2^{68} \times \sqrt{2}$

132

2·13·79: Lev:

as part of t. corpus for later trials. i.e. say q_1, q_2, q_3 are first 3 trials.

$y_1 = U(q_1, x, A(\cdot,\cdot))$; Then y_2 (2nd trial) $= U(q_1, q_2, A(\cdot,\cdot))$. Hvr, here, for trial 2, we want to preserve (perhaps) all "info" that was created in trial 1: This "info" includes q_1, y_1 & t. path going from $q_1, x, A(\cdot,\cdot)$ to y_1. Much of this info will probly not be usd — but apri, we don't know what part of it is impt.

Note that usual ideas of "info" do not apply, since they assume $CB = \infty$. For finite C.B., — The "path" from $q_1, x, A(\cdot,\cdot)$ to y_1 has info. "for $CB = \infty$, t. path has no more info than $q_1, x, A(\cdot,\cdot)$ (& t. form of U, t. reference umc), since it is computable

.12 from those params. Hvr., note that when, using finite CB, one stores this entire "path" of compns, there is much cc assoc. w. that storage — so one will

.14 probly want to be selective & store only part of that "info".

Even if we assume (as perhaps w. humans) that all of t. info is stored, we still have t. cc of retrieval (much stored info by humans has to be computed from a few stord params).

Hvr., it may be that almost all of t. potentially useful info in t. past is expressible as

.19 short codes of that past.

20: ~~64.37~~ 64.37 $\boxed{2·14·79}$ (mildly) new Tac: Suppose that t. "apri" intelligence consists of a soln. to t. gn. prob. w. t. same $A(\cdot,\cdot)$, but difrnt. x.

Say greater |x| than t. present one (so we suspect it would take no more time).

.23 I think L. mite put this info. int. "Library": $U(q, x, Lib, A(\cdot,\cdot))$.

— Again t. cc of info. retrieval arises as an impt. factor!

— Well, O.K., t. machine can learn to do I.R. efficiently — i.e. within t. C.B. gn. by $\frac{cc}{pc}$. If it does not learn this (or be gn. this info apri), it will have very poor use of t. library info .... it will too often require > t. CB to obtain needed info.

Th. exact mechanism of its learning & using various IR techniques must be gone into: This is an example of a learnd "heuristic" & ∴ is of more general interest, also.

Another Q is again: What to include in t. library? (.12 – .19: .12 – .14 in particular) Perhaps include everything in TM's past (including his path of calcus); but t. pc of most of this is rather low hvr., so put it in low cc, slow access storage. Suitably "indexing" it & deciding what to put in rapid access storage, are

.34 some of t. probs. of "I.R.".

.35 So: One possl. interpretation of L's 56.02 conjecture:

"That if there is no other info (an unusual situation that is perhaps of little interest in itself) then $\frac{cc}{pc}$ search is t. best one can do." Hvr., I don't see how he defines "no other info". He did mention t. possy. of including other info. in t. "Library" (as .23), but how this works isn't clear: e.g. Just how does t. machine learn I.R.? What is t. cc of storage? Just what info is to be stored? (.23 ff discusses some of these apts. a bit).

2·14·79 Lov:

.01 : 61.05 (Spec) Another serious advantage of $\bar{R}$ over R, is that for at least the x, $\frac{\bar{R}}{R}$ is unbnb. If this were not so i $\frac{\bar{R}}{R}$ were bnd, then t. fact that $\exists$ an optimum R implies there also exists an optimum normzd scpm — which is false. More generally, $\bar{R} >> R$ in certain (rare) cases, i for certain problems, this can be very impt — e.g. say we are interested in t. probly of a rare event — like t. malf. of a Nuc. reactor, or a very bad result from a recomb. DNA expt. That x's w. $\bar{R} >> R$ are unlikely is impt. to know, but the x's we are interested in are not a "random selection". Hvr

[This latter (non-random selection) idea may not be relevant. X is never randomly selected. — any non-randomness has to be included in t. defn. of t. reference Umc.]

That x's w. $\bar{R} > 100\,R$ have measure $< \frac{1}{100}$ is of interest, but in any particular case, we can (sometimes) easily look and obtain both $\bar{R}$ i R i see if this is t. case.

→ 67.19

.15

Other than L's very a.H. betting situation, there may be one actual practical applicn. of R: that of probly of OOL: Here we want t. probly of a self reproducing molecule. Tho we are really interested in relative probys, but th. alternatives are $P_{\bar{r}}$ (a non-reproducing molecule is produced) i P_Λ: t. machine either stops befor completing a molecule, or enters into an infinite compn. befor completing a molecule. We are interested in $\frac{P_r}{P_r + P_{\bar{r}} + P_\Lambda}$: which is what R gives us. From a practical pt. of view P_Λ would seem to be included in $P_{\bar{r}}$, so perhaps in this case, R is automatically normalized i $\equiv \bar{R}$ — Hvr. I'm not sure of this last. Th. whole situation would have to be analysd in more detail.

Situations in which t. products of condl probys is needed (like in Betting) are not so common. Usually we are interested in condl. probys, where $\bar{R}$ i R tend to be close — particularly if t. seq. is long.

Anyway wrt L's 56.02 conjecture: Ask him if t. interpretation of 65.35 is correct. If so, then perhaps t. problem is this: given $U(\cdot\cdot\cdot)$, to find an algm $U(q,\cdot,\cdot) \ni \forall$ NP probs, $x, A(\cdot,\cdot)$, ⓐ $U(q,x,A(\cdot,\cdot))$ is a value for y that is a soln. (i.e. $A(x,y)=0$) ⓑ this soln. is obtained in less time (for arbly large $|x|$) than any other algm $U(q,\cdot,\cdot)$.

$\frac{cc}{pc}$ search satisfies (a); t. Q is, does it satisfy ⓑ or is it even close to optimum?

.35 T. meaning of t. conjecture: $\limsup_{\text{wrt } x,\, A(\cdot)} \frac{T_{U(q_1,\,x,\,A)}}{T_{U(q_2,\,x,\,A)}}$ is a small constant (≥ 1)

q_1 is t. $\frac{cc}{pc}$ algm.

q_2 is any other algm.

.01 Then 66.35 seems false: Say q_2 is t. following: "look in t. following ~~finite~~ table of algms $[A_i, q_{A_i}]$ (algm, soln.); if A is in that list, then use t. soln. listed. If A is not found in t. list (using a simple matching method), then use q_1 ($\equiv \frac{cc}{pc}$ search).

If A is in t. list, then t. lim sup of 66.35 is very large, i.e. $2^{|q_{A_i}|}$ (see 28.20⅓ for discn & proof of L's $\frac{cc}{pc}$ thrm) — otherwise it is 1. Here $U(q_A, x)$ is t. truly optimum algm that computes y from x; q_{A_1} is t. soln. for "A_1". (q_A will usually be many bits. (A_1 is in t. list)

.10 Actually t. ratio 66.35 is even larger: $T_{U(q_1, x, A)} \tilde{\gtrsim} 2^{|q_{A_1}|}(T_{A(q_{A_1}, x)} + T_{(q_{A_1})})$ ← Time needed to test y.

.12 If $A = A_1$; $T_{U(q_2, x, A)} = T_{A(q_{A_1}, x)} + C$ — constant time needed to find A_1 in t. table.

$T_{(q_{A_1})}$ (t. testing time) is usually an ↑ funct of $|x|$, so for large $|x|$ t. ratio of these is $\tilde{>} 2^{|q_{A_1}|}$

.16 if A_1 is in t. table & if testing t. correct soln. takes time that $\to \infty$ w. $|x|$

Note: however this (weakens / softens) arg't. considerably — but still makes it very likely → I could probly make lower bnds on $T_{U(q_1 \ldots)}$

Well, one could say, that if A_1 is in t. table, then q_{A_1}, t. dern. of its soln., should have small q_{A_1}. In L's system, I think this is not true, since in that ($\frac{cc}{pc}$) system, t. ordering of t. algms is indep of A.

10:25P dog started barking. 11:50 still going. 12:15 stopped. 12:25 started again

.19 : 66.15 : About this "measure": Presumably, R is used for this measure.

.20 i.e. x's for which $\bar{R} > kR$ are of measure $< \frac{1}{k}$ (measure w.r.t. R)

.21 ∴ since $\bar{R} > kR$ for those strings; " " " " $< k(\frac{1}{k}) = 1$ (measure wrt $\bar{R}$).

.21 is trivially true for all k; & I think .20 follows from .21!

Furthermore, if we substitute R' for R in .20 (in both places) it is still true, no matter what probly measure R' is! So it looks like L's "easily proved" thrm. is actually silly! A useful thrm. would make t. measure $\frac{1}{k}$ be wrt $\bar{R}$, or w.r.t. t. "uniform" (Bern, $\alpha = \frac{1}{2}$) distribn., but I doubt if L. would say that, since he feels that $\bar{R}$ is of no interest.

[If R is any/probly measure on infinite strings; it can be super/normal, sub/normal or normalized.

If $\bar{R}$ is any normalized/probly measure, not necly a unvl. measure, then t. measure of all strings x wrt R for which $\bar{R} > kR$, is $\frac{1}{k}$.

Check this w. L.!

I suspect that in any impt. Q., we'd be interested in t. probly wrt. not R, but t. relative probly. L's treatment in this case is like asking a person to bear witness to himself! → 73.01

.32 In .01 – .16, t. lim sup of t. ratio is larger if we require lim sup to be for an ∞ of x values. If we require it for an ∞ of x and A values, t. ratio is bounded, since we only have a finite no. of A_i's in t. table.

1) Can we put an inft. no. of A_i's in t. table? – if so, this would be done by expressing t. solns q_{A_i} as some recursive funct of $A_i(\cdot,\cdot)$ & x.

2) Is making "lim sup" for an infinite no. of both x's & A_i's possl.?

.36 Well, we can have $\limsup_A (\limsup_x)$ (or reverse order of x, A) ⟶ probly on hacy (see 68.30) ← No, it is hacy

One trouble w. 2): certain/single elements of t. table could represent an infinite no. of algms — by some sort of parametrizn. This idea is also related to 1) so $A_i(x,y) \equiv f(\alpha, x, y)$; $q_{A_i} \equiv g(\alpha)$. [here, f & g are algms & α is any finite string]

We then have some method for recognizing if A_1, say $= f(\alpha, \cdot, \cdot)$.

This method need not work every time [i.e. certain algms will be really identical to $f(3,x,y)$ but this method wouldn't be able to tell ... ~~in genl, its imp~~ in t. genl. case, its impossl. to tell if ~~2~~ algms are identical], but it does need to work in an ∞ of cases.

Anyway: This ~~[illegible]~~ makes it possl. to put ∞ of A_i in t. table, so t. lim sup would be large, even if ~~taken~~ taken over both A_i & X.

.07 If there are ∞ of A_i in t. table, then it will not take a ~~fixd~~ fixd amt. of time to use t. table (?)

Well, say t. table says "if A is of t. form $f(\alpha, x, y)$, then $q_A = g(\alpha)$".

.09 T. table could be arranged so that $f(\alpha, x, y)$ could be recognized for arby α in finite time (perhaps)

→ but constructing $q_A = g(\alpha)$ will take a time that is ↑ in $|\alpha|$.

.11 So perhaps this counter example wouldn't ~~—~~ counter-examp. $3.14159\overset{3}{2}6$ — $\frac{355}{113}$

.12 But anyway: say t. conjecture is true! It would have to be for t. double lim. sup. (i.e. over all x & all A ~~[illegible]~~) ← see 67.36 If so, this puts all wt. on large $|x|$ & "large" A — since only large $|x|$ & "large" A's can appear an ∞ of times as required by "lim sup".

[N.B. t. conjecture is false if A's are of bndd cardinality ← a/o of bndd "size". ~~Probly~~ Also false if $|X|$ is bnd. ~~[illegible]~~ — whether it is true for both A & X un budd — I don't know].

.20 Such a theorem would not be relevant to "t. case of usual interest": i.e. say we have a tng. seq. (as humans do), & we have had much experience w. ~~certain~~ a large (tho finite) set of problem types. Our usual "method of soln" for a new problem, is to "look it up in our table", to see if it is there or is close to a solvd problem of t. past. While this is a v.p. method &, indeed, solves most problems, t. conjecture does not deal w. this at all (because there are only a finite no. of problem types in t. table), & considers such

.28 a soln. method irrelevant; tho in practical cases, it is usually much better than pure $\frac{cc}{pc}$ search.

.29 Well, let's go back to .07–.11 to see what happends if we have ∞ A_i's "in our table".

.30 : 67.36 : Actually just "lim sup." taken over all A & all x will do t. trick. No! Remember lim sup. is t. largest value that we get arby close to, ∞ times — so if there is even one A for which $\limsup_x$ is large, t. Thm. is false. So, on second thot 67.36 looks rite, since in "counterexample 67.01 – .16," we have ~~[illegible]~~ a finite no. of A's for which $\limsup_x$ is large. $\left(\limsup_A \left(\limsup_x \right)\right) \not\equiv \left(\limsup_{A,x}\right)$ by any means!

$\limsup_x f(x) = \alpha$ means there are at most only a finite no. of values of x for which $f(x) > \alpha$.

→ Well, I can proby devise a specific example in which $f(\alpha, x, y)$ (of .09) could be recognized from table in bounded time, (indip of $|\alpha|$), & t. soln, $q_A = g(\alpha)$ could be constructed in linear or poly. time (in $|\alpha|$).

2.15.79 Lev

69

.01 : Then 67.12 becomes $\approx$, for A's that are in t. table:

$$T_U(q_z, x, A) = \underbrace{c}_{\text{T. time to find } A(\alpha) \text{ in table}} + \underbrace{a \times |\alpha|}_{\text{Time to generate } q = f(\alpha)} + \underbrace{T_{A(q_A, x)}}_{\text{Time to compute } y}$$

If $a \times |\alpha|$ is $\leq \underbrace{T_{(q_A)}}_{\text{t. time to test } y}$, then from 67.10 – .12, t. ratio becomes $\approx 2^{|q_A|}$

— which is large. ~~Actually if~~ $\cancel{T_{(q_A)}}$ ~~is linear in~~ $\cancel{|\alpha|}$

woops! $a \cdot |\alpha|$ & $T_{(q_A)}$ are not really comparable: $a \cdot |\alpha|$ depends on $|\alpha|$, t. "size" of A, while $T_{(q_A)}$ depends on $|x|$ or $|y|$.

Say $T_{A(q_A, x)}$ is $>$ linear in $|x|$ — or linear in $|x|$ but $\gtrsim$? — so in 67.10, $T_{(q_A)}$ is unimpt.

.12 so ratio $= \dfrac{2^{|q_A|} \times (b \cdot |x|)}{b \cdot |x| + a \cdot |\alpha|}$ ← $T_{A(q_A, x)}$ being $\approx$ linear in $|x|$

Note: $|\alpha|$ is t. length of t. problem (A) dcrn. $|q_A|$ is t. length of t. soln. to A.

So: t. ratio of .12 would $\to 0$ for large $|\alpha|$, so this example would not destroy t. conjecture. — unless $2^{|q_A|}$ ↑ more rapidly than $|\alpha|$ ... which is quite possl.!

Any way $\limsup_x (.12) = 2^{|q_A|}$

$\limsup_A \left(\limsup_x (.12) \right)$ probly $= \infty$.

$\limsup_A (.12) = \limsup_A \frac{a}{b} \cdot |x| \cdot \dfrac{2^{|q_A|}}{|\alpha|}$

probly $\limsup_A \dfrac{2^{|q_A|}}{|\alpha|} = \infty$; but if it is bounded & > 0; say it $= L$

then still $\limsup_x \limsup_A = \limsup_x \frac{a}{b} \cdot L \cdot |x| = \infty$.

Summ: $\limsup_A \limsup_x (.12) = \infty$

.30 $\limsup_x \limsup_A (.12)$ probly $= \infty$ (i.e. if $\limsup_A \dfrac{2^{|q_A|}}{|\alpha|}$ is some constant > 0)

→ TO .10 for much simplifn

Some things that need clearing up:

1) does $\limsup_A \dfrac{2^{|q_A|}}{|\alpha|} = \infty$ or > 0? One Q is (.12 Rt)!

How does α, t. dcrn of A, compare w. q_A, t. soln. of A? It would seem that I could find probs that were not exponential in there soln. lengths.

In fact perhaps t. most common situation is $|\alpha| \overset{\sim}{\propto} |q_A|$.

Anyway, it would seem to be of no practical import. Even if t. A, x $\limsup \limsup$ thm. were true, it would seem to not be useful, because of 68.20 – .28

It reminds me of relay contact networks. Practically all of them that one can dcrb. are incapable of being signifly simplified — yet practically all networks that arise in R.W. are capable of simplifn.

2.15.79 Lev

70

.02 Hvr., there seems to be a version of L's conjecture that is equivt. to t. optimality of R.S. (≡ "Random Search"). see 61.35 — 62.05 in particular. see 61.35–37, 62.01 – .40 ; 63.18 – 64.17 ; 64.30 – 65.34

Here, the idea is something like: "If we have "no other info" — $\frac{cc}{pc}$ search is best".
When we do have other info, we incorporate it into t. umc, so we have no info other than that in t. umc; so we do $\frac{cc}{pc}$ search wrt that umc.

Hvr, re read t. refs of .02 for possibly good ideas.

The meaning of "No other info" is unclear; In a sense t. statement that $\frac{cc}{pc}$ search is optimal is a sort of defn. of "no other info" ☺

876

8296

.10 69.30: An easier way to deal w. an infinite set of A's: $A = A(\alpha, x, y)$.
(α, x) can be that of of t. "x" & y is t. "soln." as befor. We contract this problem as "only one A", or, as a difnt A for each α."
If we like, t. soln. time can be restricted to be linear in both $|\alpha|$ & $|x|$ — i.e. $\propto |\alpha| \times |x|$. In this case t soln. could be quadratic in $|\alpha, x| \simeq |\alpha| + |x|$.

Anyway, it should be possl. to get t. results of 69.30 relatively rigorously, using these ideas.

An example of an NP prob, or set of NP probs, like this: $\alpha \cdot y = x$ ← consider exact integral division w.o. remainder only. — so not all α, x pairs are possl.

... ordinary/arith. division. α & x are as (.10).

The Args of 69.01 — .30 should be worked out in detail — but I'm pretty sure t. results are t. same: i.e. both t. limsup (limsup) [x, A]

& th. limsup limsup conjectures are wrong.
[A, x]

.25

Note that this "info" is not merely "statistical info" in t. usual sense:
A person who has seen a kind of problem would only once can then know how to work all probs of that type, henceforth .... "1 shot learning".

T. way I planed to do this "1 shot learning" was to have TM reorganize himself so that he would be able to work that problem quickly (low cc), & retain his

.30 ability to work other past problems. So: "Info about t. prob" amounts to low $\frac{cc}{pc}$ for t. lowest $\frac{cc}{pc}$ soln. to that problem. This assumes $\frac{cc}{pc}$ search hvr., & that assumption defines "Info about t. problem". If we use a difnt search technique, "infor about t. problem" would be defined wrt it. → Hvr. see 90.33 on this "1 shot learning"

Is there any "indip" way we can define "info about t. problem"? — or, can we

.34 usefully use this defn. in terms of an unstated search technique, & define it in terms of itself (recursively)?

→ 90.30 very impt. idea!

2.18.79 : Lav:

T. Process of ~ 20.01 — 37.01 ff (Note 50.18 ff ; 53.01ff) (see Rev. Clara 1.01 — 4.40; ~ 1.25.79)

~~Breaking of~~ Corpus is made of seq. of sub corpi (sc's). They are each coded as well as possl. each sc need not be coded sequencially, but t. seq. of sc's is coded sequencially & forms a Tng. seq. [illegible] T. code for sc_i is an $Algm_i$, plus t. code of that sc_i wrt. that $Algm_i$. We then look at t. seq. of $Algm_i$'s & try to make a $Algm^*$ that gives a apripd for t. next $Algm_i$.

One way of looking at t. forgg: These are merely ways to try to code t. entire corpus, by breaking it up into parts (sc's) so that partial f.B. is possl. thru achievement of sub-goals. The efficacy, the accuracy, the "errors" (of not finding certain regy types) of t. technique should be analysd in this lite — How can such defitys be patchd up?

Can this approxn. system designd for induction, be applied to NP probs. of more genl. type? (we would not have t. $TM_1 = TM_2$ feature, hvr.)

One very impt. characteristic of t. method. All of t. observd regys of sc_i are completely containd in t. $Algm_i$ (or set of $Algm_i$) ~~so~~ decided ~~on~~ for sc_i. For t. apripd. of $Algm_{n+1}$ As computed by $Algm^*$ t. corpus is t. seq. of Alg_i's only (Other info in t. Sc_i's is not used). i.e. $Alpm^*$ uses only info about regys of t. corpus that is expressd in t. $Algm_i$ to extrapolate t. $Algm_i$, & hence to extrapolate t. corpus. [76.10 - 18 gives a serious difty assoc. w. this characteristic.]

Note: in t. sequencial coding of a corpus by a UIO machine, we also have partial f.B. as we code, & this reduces cc of coding (perhaps a lot). But this aforementiond method of tng. seq. construction & breaking up t. corpus into sc_i's, seems to go much further in this direction — since t. initial sequencial coding f.B. is not enuf to make $\frac{pc}{cc}$ search practical, but t. "sc_i" method does (I think) make $\frac{pc}{cc}$ search practical.

Can we regard t. new methods as changing t. radix, so t. Sc_i's are t. "Symbols" in t. [illegible] corpus? → 79.16

.35

→

→ { 77.01 — 79.15 } Hps suggs. for improving t. model. possibly limitations.

→ 76.01

On t. "optimality" of $\frac{cc}{pc}$ search: Here we are concernd w. t. not-yet-rigorous conjecture that $\frac{cc}{pc}$ search is best. T. rigorous conjecturing in this direction so far (~ 68.01 – 70.40) ~~always~~ seem to be false.

T. "lack of info" in search, I guess can be defind by certain "standandard" aprip's for t. thing searchd for. This "lack" of info, is, than, simply a defn. of what info one does have.

If we ~~disallow~~ limit t. strategies to be considerd, to those ~~avoidin~~ in which ~~certain~~ choice of trial will depend only on pc & cc of that trial, than $\approx \frac{pc}{cc}$ search may,

.15 indeed, be provably best. (41.03 – .40 ,{.28 – .37 Rt} especially).

.17 This automatically excludes "learning as one searches". A simple case of this last would be a H.C. search, in which each trail depends on t. Gorcs of recent trials in t. past. This is sort of analogous to a RTM w. history of 1. Anyway, this sort of "learning from recent history" could be implemented if Algm* was updated more frequently

.19 than just after each final Algm's discovery!

.20 An even better, more general kind of search corresponds to RTM w. longer

.21 history. Here Algm* is more than just an orderer of trials by Aprip.... (and cc)...

In .17 – .19 this behavior coud be implemented by a Algm* that bases its aprip estimates not only on (t. corpus / t. set) of previous "successful" Pam_i's, but upon t. aprips, & cc's of recent trials.

.25 For t. device of .20 – .21, ~~[illegible]~~ ~~[illegible]~~ we would try to make Algm* an optimum search algm.... basing t. expected Gorc of any possl. choice of algm. on previous history of ~~[illegible]~~ trial search Algms*'s as well as t. corpus of successful Algm's.

For .20 – .21 as a $\approx$ ($RTM_{h \gg 1}$), certain trials could be made as "experiments"... to gain info, rather than as direct (candidates) trial(s) for hy Gorc.

It may be that one would (initially, at least), use a simple $\frac{pc}{cc}$ search

.30 to try to find good search algms of type $\overbrace{.17 – .19}^{h=1}$ or $\overbrace{.20 – .21}^{type\ h \gg 1}$.

T. $\approx$ "demo" that (R^{-1}) ~~[illegible]~~ $\times$ cc search is "best" if R & cc are used to contrl search $\rightarrow$ c $\approx$ 41.03 – .40 & .28 – .37 in particular) implies that ~~[illegible]~~ $(\frac{R}{P_m'})^{-1} \times$ cc search would be better, since $\frac{P_m'}{R} > 1$. Hvr., ~~[illegible]~~ → 75.01

Actually we use 2^{9mm} rather than $(R)^{-1}$. $(R)^{-1} \times$ cc search would be better than

. 35 $2^{9min} \times$ cc if we could easily compute R [since $R > 2^{9min}$. Similarly $(P_m')^{-1} \times$ cc would be better than $(R)^{-1} \times$ cc since $P_m' > R$, but similarly, we have t. addi. tional

. 37 cc of computing P_m'. → 75.25 ; 75.35 in particular.

T. "proof" of that cc/pc is optimal hinges on $\sum 2^{-q_i}$ being about t. largest function of q_i that will converge. No funct. can be significtly larger at ∞ & still converge ← (I conjecture).

(75.01 spac

.01: <u>67.32</u> On t. Relation of R to $\bar{R}$ $(\bar{R} \equiv P_M')$.

Def It would be useful to study $\bar{R}'$: $\bar{R}'$ is t. "improperly normzd" R of I & C I :

$$\bar{R}'(x(n)) \equiv R(x(n)) \Big/ \sum_{i=1}^{2^n} R(x_i(n)) \quad \equiv c_n\, R(x(n)).$$

.04 Since t. measure w.r.t. R of ~~a~~ t. set of recursive infinite strings is > 0 ($\equiv$ infinite strings

.05 w. finite descs),

$$\prod_{n=1}^{\infty}\left(\frac{R(x(n))}{R(x(n)\hat{}0)+R(x(n)\hat{}1)}\right)^{-1} > 0. \quad \text{so} \quad \sum_n \left(1-\frac{R(x(n)\hat{}0)+R(x(n)\hat{}1)}{R(x(n))}\right) < \infty.$$

These (.04 - .05) properties of R imply that condl probys of $\bar{R}'$ are very close to those of R, for large n.

[It may, then, be possl. to find some relations betw. $\bar{R}$ & $\bar{R}'$.

Perhaps a [redacted] quicker, rigorous way to look at relation of $\bar{R}'$ to R:
.04 insures that $\Big(\sum_{i=1}^{2^n} R(x_i(n))\Big)^{-1} \equiv c_n$ (t. normza const) is bndd. for all n. $c_n \uparrow$ w. n so <u>c_∞ exists</u>.
[Each infinite recursive string we code, gives us a larger lower bnd on c_∞]

.19 $\delta_n \equiv \dfrac{R(x(n)\hat{}0)}{R(x(n))}$ and $\delta_n' \equiv \dfrac{\bar{R}'(x(n)\hat{}0)}{\bar{R}'(x(n))}$ are condl. probys for R & $\bar{R}'$ resp.

.20 t. ratio of t. 2 is $\delta_n'/\delta_n = \dfrac{c_{n+1}}{c_n}$ which must $\to 1$ as $n \to \infty$

Q. Can I find a R so its normzn. constant c_∞ for $\bar{R}'$ is <u>arbly large</u>?

.25 ~~I think so~~ Yes! : Consider t. following UMC (from which R is to be obtained).
T. machine has no output until at least m bits have come in. If these first m bits are not all 0, t. machine stays in an infinite non-printing loop forever.
If t. first m bits are all 0, then it reacts to subsequent bits like any desired UMC. Here, $c_n > 2^m$; we can make m as large as we like.
Call t. R assoc. w. .25, R'. say t. R assoc w. this umc. is R: Then $R' = 2^{-m}\cdot R$. R' is "optimal" — as is R.
since $R \leq \bar{R}'$ for <u>every</u> $x(n)$, $R' < 2^{-m}\bar{R}'$ for <u>every</u> $x(n)$

that t. measure <u>w.r.t. R'</u> of all strings for which $\bar{R}' > 10R'$ is [redacted] $< \frac{1}{10}$. (Tho m can be, 10, say).
This is primarily due to t. fact that R' is <u>small</u>. Its <u>total</u> measure is $< 2^{-m}$ — so not only is $\bar{R}' > 10R'$ "unlikely" w.r.t. R' — <u>anything</u> is [redacted] unlikely w.r.t. R'.

So it is clear that R can be arbly smaller than $\bar{R}$ for <u>all</u> $x(n)$ . . . <u>not</u> just one!

2·19·79 Lev.

.01 : 73.40 Yet, t. Q. of how $\bar{R}$ difrs from R for conditional probys hasn't been resolved : 73.09 -.20 is a step in this direction. Can I get an estimate on t. ratio of condl. probys of $\bar{R}$ & $\bar{R}'$? I think I provd that $\bar{R}$ & $\bar{R}'$ were not identical.

Hvr. $\sum_{i=1}^{2^n} \left(\bar{R}(x_i(n)) - \bar{R}'(x_i(n)) \right) = 0$, and for large n, t. condl. probys of $\bar{R}'$ are very close to those of R.

T. ratio of R to $\bar{R}'$ is bounded by ∞. Since t. ratio of $\bar{R}$ to R is unbndd, t. ratio of $\bar{R}$ to $\bar{R}'$ is also unbndd.

It is possl. that $\bar{R}'$ is not a "Measure function" in t. usual sense.

I think a "measure funct." is a measure on t. space of all infinite strings. M(x(n)) is t. total measure of all strings w. prefix, x(n). A normalzd (like $\bar{\bar{R}}$) measure may be one in which t. total measure of all infl. strings is 1. — I'm not sure that's all, hvr. If it were so, then one would need only 1 constant to change an unnormzd into a normzd measure.

The "R" measure can be regarded as a simple normzd measure, but w. 3 symbols : 0, 1, & U (U has t. roles of Willis' U's).

It may be that t. condl. proby for $\bar{R}$ → those for R as $\frac{1}{n}$:

.20 i.e. {t. proby of a sequence stopping (or not continuing) is ≃ $\frac{1}{n}$, if it has not stoppd for n bits.} ← conjecture.

→ More on R v.s. P_M' → 90.08

2.19.79 Lev: 75

72.30

.01: (spec): So: Genl. Conclusions!

.02 1) T. conjecture of 56.02 is probly wrong: even if we change it a bit.

— as $\limsup_x \left(\left(\limsup_A\right)\right)$ or $\limsup_A\left(\left(\limsup_x\right)\right)$. → [T. discn. terminating on 70.25 ≈ shows this]

.05 2). If we restrict t. conjecture so that t. statement is for all searches where t. choice for t. next trial is a funct of only cc & |q| of that trial — Then it may conceivably be true for "large x". [≈ 41.03 – .40 ; .28 – .37 in particular]

.10 3) If t. restrictions of .05 do not hold, & each choice of trial depends on cc of trial, q of trial & t. history of success & failure of trials for that problem, then t. conjecture is probly not true. — That there are probably much better methods using "hill climbing". [72.17 – .19]
See OSTM 53.10 for possl. way to overcome this objection.

19 4) If we further relax t. constraints of .10, we can include arby strategies.

In 3) ↑ I ~~expected~~ wanted each trial to have t. max expected G. "in view of t. stated info available." This would make "experiments" impossl. If we allow arby strategies with a G like that for a RTM h=1 (like for RTM $h \gg 1$), then we may be able to get something like optimum. [72.20–.21 ; .25–.30]

23 → 5) Hvr., see 89.25 for a new approach that may make t. conjecture possl.

.24

h=1 RTM

$h \gg 1$ RTM

.25 : 72.37 On t. possible sub-optimality of R for cc/R search:

Say we had a $R' = 2^{-m} R$ (like 73.25 ff). If we used R' for search, T. total expected cc = $2^{q+m} \times$ (cc + test of soln), which is 2^m times as much as using R. Hvr., in fact, if we use R' for search, t. search would have exactly as many steps as t. R search & each step would take t. same time as that of t. R search — so total cc would be < $2^q \times$ (cc + test of soln.)

So 2^q (cc soln.) is, in general an upper bnd on Σ cc,

.35 & in some cases, can be a very poor approxn. Using P_M (rather than R) for search (as 72.35) may yield only an illusory ↑ in speed, even tho P_M could be >> R. At t. present time, I really don't know.

2.19.79 Lev:

Sort of summary of 71.01 – .90 (on t. general non-optimality of this ~~lexical~~ use of tag. seqns) and 72.01 – .90; $\boxed{75.01 - .24}$ = summary of (on t. non-optimality of $\frac{cc}{pc}$ search). 56 pp 31 d.

72.01 – .40 – 75.01 – .24 give some ways in which t. search Algm* mite be much improved. Such improvements could also overcome some deficiencies of t. tag. seq. system of 71.01 – .40 — i.e. a more general Algm* could look at previous trials for t. same Sci, ect. Hvr., a serious deficy of t. present/tag. seq. method is that all info re past Sci's is obtainable only thru t. Algm's — (t. details, t. raw symbolseqs) of t. Sci's are not retained.

.10 {Such "details" could be impt. if one is looking at a possl. defn. ~~...~~ in a particular Sci, & t. SSZ. ~~seem~~ is too small to cover t. "overhead" of that definition — & one wants to use t. raw (or nearly ~~...~~ raw) data from earlier Sci's to see if t. defn. can be used.

In general, various abss will be thot of late in t. corpus & t. raw data of t. earlier Sci's would be useful in confirming or denying ~~...~~

.18 their feasibilitys.

So: pp 71, 72 & 75 really do summarize impt. deficys of t. system & suggest, to some extent, how these deficys can be overcome.

(Hvr. .10 – .18 is a new kind of deficy that is not treated therein).

Anyway, t. deficys do not seem to be very bad! Probably a ~~...~~ quite good demonstration system could be constructed, inspite of these deficys. Of course overcoming th. deficys would yield an even better system.

.28 ((Of t. deficys: that of 71.01 – .40 (t. tag. seq. system) is most serious & most impt. to understand, so I can overcome these deficys w/o ganz. t. system.) → 77.01

2·20·79 **Lev:**

.01: (spec 71.40 76.28): A criticism & Genzn. of t. Exp. Seq. approach to TM. developement.

T. presently contemplated system has this large corpus to be coded. Doing t. coding directly would take too long.... so we use a particularly simple form of corpus: i.e. one that is divided into a seq. of Sc_i's, w. t. understanding that much info is contained in t. Sc_i "boundary markers" — & that t. entire corpus can be coded fairly well using $\frac{cc}{pc}$ search on t. individual Sc_i's, one after t. other, with suitable updating of t. apriori of q_{i+1} after Sc_i has been coded.

What are t. approxs. & assumptions being used here?

.14 1) That t. coding of each Sc_i is to be of t. form: $U(q_i, Sc_i)$ = code for Sc_i or proby of Sc_i.

— This assumes that t. stochastic generator, $U(q_i, \cdot)$ formed t. corpus. This is not t. most genl. form of a code for ~~[illegible]~~ Sc_i (But I'm not sure of this), but it is a very common form for v.g. codes & is proby t. most common method used by humans.

.21 2) That the effective corpus for t. $i+1^{th}$ Sc is $q_1, q_2 \ldots q_i, Sc_i$: (rather than $Sc_1, Sc_2 \ldots Sc_i$) i.e. t. regys to be used in coding Sc_{i+1} are all completely contained in t. q_i sequence.

.23 3) As a (perhaps) corrolory of 1) & 2) & perhaps 4): That the influence upon t. q_{i+1} search will be ~~[illegible]~~ entirely in t. form of t. extrapoln. of t. seq. $(q_1, q_2 \ldots, q_i)$.

.26 4) $\frac{cc}{pc}$ search will be used, using 2^{-q_i} as an approxn to pc_i:

By $\frac{cc}{pc}$ search we mean that $\frac{cc}{pc}$ ≈ rigorously determines t. order of trials. Any deviation from that order is for convenience & does not cause $\gtrsim$ a factor of 2 or so in expected Σ cc of search.

Proby Random Search (≡ RS) could be included .. RS could take advantage of inexpensive components having random "Bugs" in them: Also ∥ random search by cpu chips that have difrnt random construction errors in them.

Use of ∣ Pgming ~~[illegible]~~;
1) to make sure some ~~[illegible]~~ spending time on really impt. probs.
2) A difrnt "language" for t. same old prob. is always v.g.

.01 : 77.40 : How to Genlz. from t. constraind problem of 77.01 — .40

.02 a) Relax constraints on search methods (77.26 ≡ (4)) :
3 kinds of less constraind search are possl. Trials are based on knowledge of degree of success of previous trials, as well as t. apriori pd on q_{i+1} as obtaind from t. ($q_1 \cdots q_i$) sequence by Algm*. This dependence is of ≈ 3 kinds:

.08 a) Each trial is to obtain t. most likely q_{i+1} that will fit into t. CB.

b) short range experimental q_{i+1}'s are possl. — say to determine predictive info

c) Arbly. large scale experiments are possl.

.12 for Algm* we have some Algm. that can do this sort of thing [this is true of a) b) & c); but c) is by far, most difft] T. problem of designing Algm*'s of this complexity (this is a "sequencial / opten" problem — t. most complex type of sci. problem that I've much thot about) will be initially done by me: later by TM after it had experience working such problems. At an intermediate stage I might cooperate w. TM to work on such problems.

Here (.02 ff) I am saying essentially 75.01-.24 & 72.01-.30.

b) Relaxing constraints 77.21, .23; (2) & (3): T. coding for Sc_{i+1} is still $U(q_{i+1};)$ but ($q_1 \cdots q_i$) is not t. only info. available.

1) Possy. of T.M. dividing its own Tng seqs. from R.W. or from very large "Library"

.23 — Say ① t. entire past corpus is available ($Sc_1, Sc_2 \cdots Sc_i$)

.24 even more: ② in addition to ①, the paths of computations for t. earlier coding are available.

Just how ① & ② could be used to help find a v.p. q_{i+1}, is not yet clear to me. Hvr., it is clear, that ① .23 is a much more normal view of t. problem, & ② .24 is a natural addition to t. available info.

Note that t. availability of ① & then, even more, ②, will complicate t. simple P/cc search, & even further complicate t. more genl. searches of .08 – .12.

Semi-semi final Genzn: That extrapoln. need not be of t. form $U(q_{i+1}, \cdot)$ — but can be any genl. code.

Semi-Final Genzn: T. corpus is not neatly divided into Sc_i's: T. search method may try to divide it (tentatively) into Sc_i's as on particular coding technique. ∑ Various/devices for tentatively dividing up t. corpus for coding can be devised. Certainly humans normally do this — we know which parts of t. envt. are natural continuations of other parts. 79.01

2·20·79 : Lev!

.01 : 78.40 : "Final" ← ✓ (Not by a longshot!!). Genzn: TM is able to devise its own tng. seqns by access to RW. ~~w~~ 2/o a Large library. Here we must have aux. ~~many~~ goals for t. T.M., so t. devising of these Tng seqns is oriented toward these goals. Humans normally do this tng seq. construction for themselves by reading books of ↑ complexity directed toward understanding a very complex subject. Also, areas of experience ~~s~~ in RW can be selected so as to lead to understanding of a certain complex area. This technq. of Tng seq. construction is usually not diflt to do for Humans. — whether ~~they~~ do it ~~[illegible]~~ very well is unclear, but they do do it more or less "adequately."

.15

.16 : 71.35 One characteristic of t. present, restricted method is t. use of partial R.B. of t. sci's during coding of t. corpus. Are there any ways that I can generalize this kind of partial R.B., so I can find new ways to code very complex corpi that would be otherwise "trans computable". ?

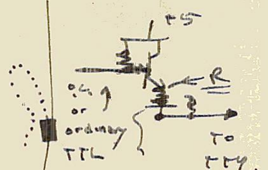
.20

4:45 plugs in
~~p~~ 1 plug out

16k Bytes
10 bytes/sec
1.4 k sec
$\frac{1400}{60} = 23$ minu
$\frac{1400''}{12} \cong 100'$!

~5 k" of tape on spool.
~ 19 k" of tape in Fairfield

+5



diode
patterns
"poke" into ROM, to get serial output.
So use ROM addn "AND" write, to give output gates.
Use 2 output addresses to set, reset output F.F. or 2 bits on DataBus.

2.21.79 Lav — 80

To discuss:

1) Conjecture of 56.02: NP prob. $A(x,y)=0$, given x.

.02 a) to find q $\ni$ $y = U(q,x)$ is a soln.

use $\ell(q) + \lg(T_U + T_{test})$ search.

.05 b) Same as a) but $y = U(q, x, A(\cdot,\cdot))$. We try to find q that works for $\underline{\underline{all}}$ A, x.

$[A_i, q_{A_i}]$: problem A_i, soln. q_{A_i}

Discn: 67.01 is counter example for a) (.02): Table entries: $[A_i, q_{A_i}]$

q_1 is $\ell(q) + \lg T$ search

q_2 " Table plus $\ell(q) + \lg(t)$ search.

$T_{q_1} \approx 2^{\ell(q_A)} \left(T_{\text{to generate } y} + T_{\text{to test } y} \right)$

shortest algm to generate y from x: q_A

If A is in t. table: $T_{q_2} \approx \left(T_{\text{to generate } y} + \underset{\uparrow}{c} \right)$

Time needed to find A in finite table.

$\frac{T_{q_1}}{T_{q_2}} \approx 2^{\ell(q_A)}$ which can be $\cancel{}$ $>> 1$.

Also $\limsup_x \frac{T_{q_1}}{T_{q_2}} \approx 2^{\ell(q_A)}$.

for b) (.05) { 68.29 – 70.25 discusses this.

Say we want to know if $\limsup_A \limsup_x \frac{T_{q_1}}{T_{q_2}}$ is ≈ 1.

Let q_1 be as before; Let q_2 be as before, but t. table has one entry:

$A(\cdot,\cdot,\cdot)$ $(\equiv A(\alpha, x, y))$ α is t. index of t. problem.

table: $\left[A(\alpha, x, y)\, q(\alpha, x) \right]$ or $A(\cdot,\cdot,\cdot)$; $q(\cdot,\cdot)$

E.g. NP problem is $\alpha \cdot y = x$

to find y, given x and α.

Restrict defn of $A_\alpha \equiv A(\alpha,\cdot,\cdot)$

$T_{q_1} \approx 2^{\ell(q_A) + \ell(\alpha)} \left(\underbrace{T_{\text{to compute } y \text{ from } x}}_{\substack{\equiv \text{ time to compute} \\ \alpha \div x \ \approx \ell(x)^2}} + T_{\text{to test}} \right)$

indip of α

$\frac{2^k}{k} \to \sim k^2$ b.t.

$T_{q_2} \approx T_{\text{to comput } y \text{ from } x} + c$

$\sim \ell(x)^2$

Time to use table; indip. of $\ell(x)$.

So $\frac{T_{q_1}}{T_{q_2}} \approx 2^{\ell(q_A) + \ell(\alpha)}$

$\limsup_X \frac{T_{q_1}}{T_{q_2}} \approx 2^{\ell(q_A) + \ell(\alpha)}$

$\left(\limsup_A \left(\limsup_X \frac{T_{q_1}}{T_{q_2}} \right) \right) = \infty$

2·21·79 Lev:

To discuss

2) Thm: If ~~Rem~~ M is any measure function infinite strings (not necly Normzd)

and N is any normalized measure on inft. strings.

~~Then, the measure~~ $M(x(n))$ is t. measure of all strings for which ~~xxx~~ t. n bit segment, $x(n)$ is the prefix.

Then / the measure wrt M of all strings x such that / for any n, $M(x(n)) < \frac{1}{k} N(x(n))$

is $< \frac{1}{k}$

If R is t. usual optimal measure, then $R' = \frac{R}{1000}$ is also optimal.

For all strings, x $R' < \frac{1}{1000} \bar{R}$ yet, the measure wrt R' of all strings for which $R' < \frac{1}{1000} \bar{R}$ is $\frac{1}{1000}$

~~This suggests~~ Note that M is used to measure how much smaller M is than N.

If M were very small it would give a very small "probability" for $M < \frac{1}{k} N$.

A more reasonable measure to use would be $\bar{R}$. R can be arbitrarily smaller than $\bar{R}$ and still be "optimal".

3) A common method of soln. of NP probs:

a) look up problem in table; use soln. if prob. is in table.

b) search for ~~"solns" to~~ probs that are "close" to t. unsolved prob. ~~... close problems in table~~ Look for solns that are "close" to those of t. "close" problems.

2.22.79 Lev:

Re: Yesterday's discn:

1) Consider any problem that is "NP complete" ("Univl. N.P. prob.)

Using L's soln., say we find a minimal $[\,l(P_0) + \lg T_0\,]$ — ⇒ $y = U(P_0, x)$ is a soln. to t. problem.

(we can call this Min value, t. "L-complexity" of t. soln w.r.t. t. reference [redacted] umc

This P_0 & $\lg T_0$ will both be dependent on t. "x" in t. univl. N.P. prob.

This problem need only be shown for Univl. NP probs — in which case it is true for all NP probs.

.10 One Q is: how do $l(P_0)$ & $\lg T_0$ grow w. $l(x)$?

.11 Is $l(P_0)$ bdd? ; Is $\lg T_0 \approx$ bnd? (i.e. very slowly growing w. $l(x)$.

For Times, T_0 must $\to \infty$ as $l(x) \to \infty$, since t. time taken to read x by t. $A(x, y)$ testing algm & y computing algm. is $\propto$ ~~\$~~ $l(x)$.

SN

L says one can have machines that have bdd time to read x.

I think this is ~~an~~ perhaps not ~~an~~ practical, because c.c of reading x is an ↑ funct of ~~[illegible]~~ $l(x)$.

Also, this is/a perhap criticism of ~~a~~ Kols' parallel network machine: It may have interesting properties, but perhaps parts of it that should ~~take~~ cost computing, do not.

back ref.

L feels that t. answers to .10 ~~■~~, .11 are likely to be surprizing.

2) Re: The $\bar{R}$ v.s. R problem: L says there are 2 not necly related Q's:

a) [redacted] : How much is $\bar{R} > R$?

b) In general is $\bar{R}$ or R a better concept?

My own impressn. is that b) need not be answered in genl., & in each

.30 application one must decide which is to be applied.

T. discn. he had was Re: a)!

Actually, this is not of ~~direct practical~~ import: In any real case, we can compute R & $\bar{R}$ & compare. It is of theo. interest, hvr. Also note that for small C.B.'s R is smaller & ~~\$~~ so ~~$\bar{R}$ becomes that much $\gg R$~~

.32 Say we chose a certain umc, U; d is its characterizn. function:

d ~~is~~ a binary string. Let ~~[illegible]~~ x^N be t. binary string representing t. integer N; Then, if for input x^N, U eventually stops, the N^{th} bit of d is 1.

d is clearly incomputable. It is also "Random".

Also he said: $d = \sum_{n=1}^{\infty} p(n)$ in which $p(n)$ is any semi-computable distribn on integers: $p(n)$ is a distribn. on a set of finite objects ∴ Kraft inequality holds

I don't see this! computable proby distribns. are also semi comp. If $p(n)$ is computable, d is, also: which is contrary to previous data!

.01 Anyway he said if $\lg \bar{R}(x) - \lg R(x) = \alpha$ — I think this is supposed to be true for large α.

.02 then $I(x;d) \gtrsim \alpha$; R, $\bar{R}$ & I are all computed wrt t. same reference machine.

$I(x;d)$ is the information about d contained in x — (See Levin for exact defn).

.01 – .02 would seem to have to be wrong. Suppose $R \to \frac{R}{k}$; then from .01 $\alpha \to \alpha - \lg k$; Hvr., from .02 $I(x;d)$ should $\downarrow$ by $\lg k$! (If my concept of $I(x;d)$ is correct.) ; On t. other hand if α & I both $\uparrow$ or $\downarrow$ by same amt., t. result is trivial : like (67.19 – .32)!

Anyway, L says that $I(x;d)$ being large means that x is very much like d (e.g. say x & d have t. prefix $x(n)$ in common, then $I(x;d)$ is $\sim \lg n$. I think) — so large α means that there are "very few" (in some sense) x's of this sort.

$C_\alpha(x)$ = smallest p $\ni$ $P_\alpha + T_\alpha \le \alpha$.

$C_T(x)$ = smallest p $\ni$ $T \le \alpha$ — diff to find! — it may not exist in any way but very long search — But for 82.10 – .11!

3) Again on t. $\bar{R}$ v.s. R problem.

.17 L. says $\underbrace{R(x/d) \gtrsim \bar{R}(x) > R(x)}$

This means $R(x/d) > 2^{-b}\,\bar{R}(x)$, where b is t. length of some short pgm. L. says Umc can be selected so $b = 1$.

.20 so $R(x/d) > \frac{1}{2}\bar{R}(x)$ for that Umc.

I guess from he says that $R(x/d)$ is "usually" not much $> R(x)$ — if it is, only if x is a lot like d. "proby wrt a uniform proby distribn." Perhaps t. "proby" of x having B bits of info about d is $\sim 2^{-B}$?

[We know that $\bar{R}(x)$ is only $> R(x)$ for at least on x : this could be $x = d$ — for which $R(x/d) = $ something > 0 and $R(x) = 0$. (even to x is a infinite, non-recursive string!

[But $\bar{R}(x) \neq 0$ only if x is recursive, (note that d is not recursive)]

T. idea of .20 is that if d is known, $\bar{R}$ is "a computable proby measure" & so $R(x/d) > 2^{-b}\bar{R}(x)$ holds for all x.

.30 More exactly, using d, $R(x,\cdot)$ can simulate $\bar{R}(x)$ with, for a certain Umc, 1 extra bit of x/tn. code.

If .17 – .30 is correct (& it seems not unreasonable); Then there is a certain Set of umcs that has t. instructions for using d to help simulate $\bar{R}$. This Umc has to have t. defn. of t. umc that created R, built into it! Perhaps part of itself could be that umc. So if U_0 is t. ref. machine for R, & $\bar{R}$;

then $U_0(p, \Lambda, \Lambda)$ gives t. distribn. for R (using uniform distribn. for p)

$U_0(p, d, \alpha)$ simulates $\bar{R}$; α is some sort of fixd pgm that tells $U_0(p, d, \cdot)$ how to use d to simulate $\bar{R}$.

T. argt. of 83.17–.40 seems closest to being reasonable of all L's argts in this area. Hvr., I must go over it & see just what t. "constraints" are ....
— Just how restricted t. reference Umc is.

[That $\overline{R}$ is not much > R implies that very few inputs to t. Umc. fail to converge.
Hvr., it still may turn out that for reasonable (CB's), very many inputs do not converge.
I think there was this sort of result by Doyle on how very short codes took very much time to converge.

Also note 82.30–.32 re: t. practical imptc. of t. forgg. results (even if they are true!) or lack of imptce.

Ew Muffs on 825p 9P rite or may hurt a bit.
925p still tolerable

Another interesting constraint on R's we should consider.

$\forall x$, we will not consider an R, if there is another $\tilde{R}$ ($\neq R$) $\ni$ $\tilde{R}(x) \geq R(x)$. This elims. t. possy of $R \to \frac{R}{k}$ being legit., & is one way to keep R as large as poss!.
R's satisfying this constraint are, of course, closest to $\overline{R}$.

T. Machine / for $R(x/d)$: Call it $U_R(p,d)$: U_R looks at p's first input bit:
if it is 0, it acts like $U_0(op)$ — a Umc. with an assoc. proby. distribn $\frac{1}{2}R(x)$
[redacted] if t. first bit is 1, it simulates $\frac{1}{2}\tilde{R}(x)$: a machine that would be assoc. w. measure. So its final proby distribn is $\frac{1}{2}(R(x)+\tilde{R}(x))$.
[This "proof" of L's would seem to work for any R, even if $R \to kR$; which is clearly impossl.! Try to see why it [redacted] wouldn't work for $\frac{R}{k}$.]

Re: $U_R(p,s)$: for [redacted] all values of s, (other than d), $U_R(p,s)$ is semicomputable. For s=d, it is computable, since it converges for all inputs, in finite times.

One simple way that $U_R(p,d)$ could work, that would produce a normed output measure (tho not necly $\overline{R}$). Any such ● normed measure would be, on the average, at least k times as large as $\frac{R}{k}$. (?)

Another $U_1(p)$ of interest:
for first bit 0, it acts like $U_0(op)$ & gives \$cpm $\frac{1}{2}R(x\,\&\,n)$ — it waits for t. next input bit:
" " 1 its output ≡ its input. " " cpm 2^{-n} for all x's
that begin w. 1; It gives $\frac{1}{2}2^{-n}$ for [redacted] x's that begin w. 0. This output measure is at least "$\frac{3}{4}$ normed".

.30 I would guess that t. weak pt. in t. argt. is that if $R(x/d)$ is really within $\frac{1}{2}$ of being normed, then $R(x/d)$ must be able to be arbly > $R(x)$ for all x.
L. says $R(x/d)$ can't be much > $R(x)$, except for a relatively small set of x's, that are informationally close to d. This may be false; because $\equiv R(x/\Lambda)$
while $R(x/s)$ is ordinarily not much > $R(x)$, this may not hold for t. special case of s=d, since d has very particular info about t. structure of R.

L. mentioned something about $U(p,s)$ being a 2 input Umc;
If so, then that would mean that t. 2nd inputs form a prefix set: i.e. they are descriptions of machines to be simulated via t. first input: hvr. this would seem to rule out t. infinite string, d, as legal 2nd input

2.24.79 Levin:

Another thing he mentiond often in these recent discussn:
He wanted me to chose t. umc first, then he would show that it had t. desired properties. He felt that we would probably agree on what a "reasonable umc" was (& this is probly true) .... But then I got t. idea that perhaps $\overline{R} \approx R$ mite be true only for these "Reasonable" umcs: —→ [redacted] we haven't really been able to characterize them.
Also, I'm not at all sure that t. "natural" umcs are necly also t- ones for which $\overline{R} \approx R$. Here [redacted] by "natural" I mean t. ones that have short defns for things that we think are "natural".

It is not clear to me whether L. feels that $\overline{R} \approx R$ is true for all R or for just R's that are reasonable a priori. Clearly, $\overline{R} \approx R$ can't be true for all R's — (because of $R \to \frac{R}{R}$).

So: Genl. conclusions so far:

1) I'm not so certain about t. argt. of 83.30; (84.30 seems most strong against it)
Also 85.04

2) T. Q of whether $\overline{R}$ is usualy $\approx R$ for most umc's of interest. This is an interesting theo. Q. It amounts to asking $\approx$ what fraction of inputs do not converge? If most do converge (i.e. $R \approx \overline{R}$), then this may mean that certain approxns. we make w. finite CB's tend to be ~~error in which we can~~ → we can estimate upper bds of error.

3) Even if $R \approx \overline{R}$ for $CB = \infty$, we are much more interested in t. finite C.B. case — & it is likely that there's a lot of difference ... since computation time for short codes may ↑ very rapidly w. shortness of code & w. n.

4) As to whether R or $\overline{R}$ is "more unpt" — this is not a good Q: in any particular case we can calculate t. one that is most appropriate — also we can calculate if R_T is very close to $\overline{R}_T$ — for finite CB. — which is t. only directly practical situation.

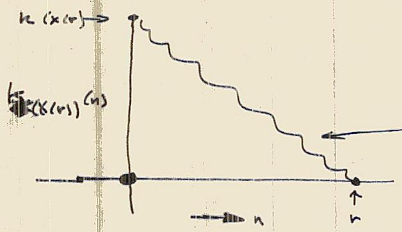
5) This may be relevant in making estimates of error in situations like calculating proby of malf. of Nuc. reactor: in which we know we have not traced out all of t. possl. failure modes.

6) T. Q. of whether t. cond. probys of $\overline{R}$ approach those of R as $n \to \infty$ [see 86.01 ff for discn.]

The main thing he wanted to show me was some ideas of Kolmog:

$K_{x(r)}(n)$ is the smallest no. of bits needed to define a $\le 2^n$ member set of strings to which $x(r)$ belongs.; $K_{x(r)}(0) = K(x(r)) \equiv$ t. K cost of $x(r)$.

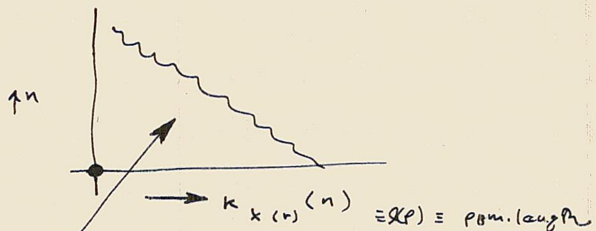
$K_{x(r)}(n)$ is t. cost of specifying $x(r)$ to within n bits. $K_{x(r)}(r) = 0$



If $x(r)$ is Random, then this curve is a st. line "at 45°".

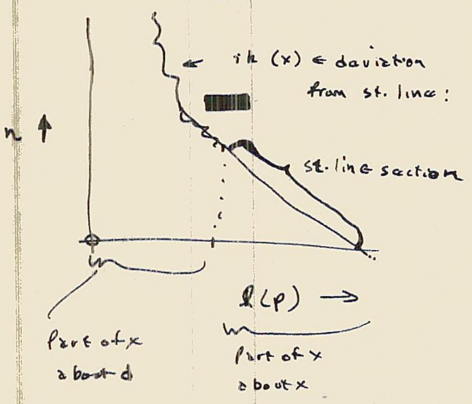
Usually it is drawn:

In this case, t. curve is strictly monotone ↓.



L. has shown that this curve can have any monotone ↓ shape — i.e. gn. any monotone ↓ funct, one can find some $x(r) \ni$ that funct is its $K_{x(r)}(n)$.

Since this st. line, 45° behavior is characteristic of Random sequs., K asked if one could use this curve shape to characterize randomness. L. says one can't: that the only x's whose curves deviate much from a st. line are those that are informationally close to the "d" of t. ref. Umc. "d" is t. "characterizn. funct of t. umc". (see 82.32 for defn. of "characterzn. funct").



ik(x) cannot go below t. st. line.
(I guess this would amount to > complete info).
The section where ik(x) deviates from a st. line corresps to t. part of x that is about "d"

$ik(x) \lesssim I(x;d) + \sim \lg k(x)$

$k_n(x) = K(x) - n$ (≡ K cost.) or $< I(x;d)$

L. feels that this $K_{x(r)}(n)$ is not so interesting for CB = ∞, but if one considers cc limits, it does become an impt. concept.

2.27.79: Lev:

L. said he talked to some guy who found a way to tell if a finite TM was in a loop — using same memory (3 tapes I, O, work), but exponential time!
Hvr: he said it could be done in ~ loop time if twice original memory were used. Q: Under what conds. could we save anything using this method to reject certain trial TM codes (for induction or NP probs)?

L. says that Kol. & Barzdin showed that if one has N^2 neurons, then the volume of wires needed to connect them w. one another is $\propto N^3$. So: the "Grey matter" (cortex) is $\propto$ Area; "White matter" (connexns) is $\propto$ volume.

2.2.87 Lev

In Rev 7.12 — .40 a serious problem has arrisen: i.e. t. more detaild dcrn. — see ibid 7.30 for bibliog. review.

of t. TM I've been contemplating: One one hand, there seems to be a somewhat clear dcrn. of a TM for "pure induction" [Rev 1.01 – 4.40; Lev 53.01 – 54.40,] also detaild discu. & criticism/of this model & suggs for improvement: [71.01 – .40; 76.01 – 79.40 || 72.01 – .40, 75.01 – .24]

There is some discu. of t. Genzn to a mixd NP, Gray NP, Induction Machine: [43.28 – 45.40; 48.01 – 50.40; 52.01 – .40] — 90.33 is I think, a final integration of Induction & NP.

Hvr., at present, I'm not even sure about t. mechanics of t. pure induction Machn... & even less, its genzn. to NP & GNP. → T. Model of Rev 8.01 – .40 may be adequate.

O.K. so start on "pure induction" machn: Consider t. mechn. of 53.01 – .20, w. method 53.27 ff. for extrapolating apripd of q_{i+1}. This model is n't so much t. way humans do induction. [Hvr. t. criticism, a genzn. of

.18 this is relevant.] More directly, the 53.01 – .40 method, at best, uses t. previous

.19 ($q_1 \cdot q_2 \cdots q_i$) sequence to get proby distribn. for q_{i+1} [see 45.18 – .40 for discn].

.20 It does not "look at" t. sc, sc_i & upon t. basis of a few low cc observations,

.21 decide on a modified apripd for q_{i+1}. Upon t. past ($q_1 \cdots q_i$) info, trial q_{i+1}'s are made in ascending $2^{(q_{i+1})}$. T. compn. order: One can have, then, a rbly good algm (Algm*) for computing t. proby distribn. of q_{i+1} upon t. basis of ($q_1 \cdots q_i$). This idea is another factor pointing up of t. sub optimality of $\frac{cc}{pc}$ search & seems to be in addition to t. ideas about this on 72.01 – .40; 75.01 – .24

.25 A perhaps fairly non-al. version of what Algm* should do: It looks at all previous info available: $q_1 \cdots q_i$, the pc's obtaind by each, t. associated sc_j's worked on, & t. path of trial compns. for each pc. computation. From this, a seq of compns. on sc_{i+1} is done, that has a hy expected $\frac{\Delta pc}{\Delta cc}$, for reasonable Δcc. This seq. of compns. can involve compns. that are not trials of $Algm_j$'s (i.e. for pc_{i+1} evaln) — Any such non-terminal compns. can be regarded as "experiments" & are oriented toward obtaining info — they are like those of a $RTM_{h>>1}$. The "looking at sc_{i+1} w. low cc observations" in .20 – .21

.35 would I think, be coverd by such a technique:

Note that t. "≈ Non-al" device of .25 – .35 is included in t. olde $h>>1$

.37 device of 72.21 – .30

Well, say .25 – .37 is an adequate approach to a Pure induction Machn.:

T. Q then is: what is a good path of development ~~toward such~~ culminating in such a device? Would it be possl. to start w. t. device of Rev 1.01 – 4.40; Lev 53.01 – 54.40?

Wed.
2.28.79; Lawn:

Modrine 400 mg. } 3 per day
Started Mon.
Bishko said it was for Arthritis.

1 — Note, that at t. begining, I ~~will~~ be TM_2 -- So Algm* can be arbly. complex.

Start out w. t. TM. of 53.01 – 54.40; using 88.18 – .19. Next add t. "trim" of 88.20 – .21: Here we "factor" t. Algm* operation into 2 parts: (a): algm β to look at SC$_{i+1}$ & Modify t. apripd of possl. Algm$_{i+1}$'s to try. (b) T. trial of t. selected Algm$_{i+1}$'s, using $\frac{cc}{pc}$ search.

I can really use any Algm* I like (.01) — T. main constraint is that I put it in good factored form, so that later, when $TM_2 = TM_1$ is "effected", t. machine can "understand" Algm* & improve it. So I can go as far as I like toward 88.25 – .35 in Algm*.

As I construct t. Trg. Seq. I will realize what features I need in Algm*. To start off, t. seq. will be simple & t. same Algmj will be adequate for all Sci's. Next, we will have several difrnt types of Sci's, — so that difrnt Algmj's are needed for each. ∃ ~~Algm~~ Algm* will then have to find out how to tell which Algmj to use w. which Sci.

At first, Algm* can try all of t. known-to-be-useful Algmj's on all of t. Sci's: By noting ~~then~~ which work w. which, t. set of all Sci's is partitioned into subsets. Algm* then must try to ~~find~~ devr easy ~~...~~ obs. ~~recognize when~~ to tell which subset a new Sci is in ... a usual "pattern recog" problem. — we want t. obs to be low cc & hy pc.
— so perhaps use $\frac{cc}{pc}$ search for it!

---

25 : 75.23 Re: Levin's idea that $\frac{cc}{pc}$ search may be optimum (in some sense) for finding induction codes. One way this ~~may~~ be true:

30 Say we have a 2 input Umc; $U(x, L^*)$. Here x is t. corpus to be coded, and L^* is t. "shortest code" of all of t. "Library ~~...~~ into," $\to L$, one wants to use to help code x. [Shades of Chaitin]

[Prob'ly I meant $U(p, L^*)$ we want $p \ni U(p, L^*) = $ ext. of x.]

[■ L^* = shortest $|p|$: $U(p, \Lambda) = L$.]

33 If L is to contain heuristics, hvr, L^* should not be t. shortest code for L, but the best obtainable within a $\frac{cc}{pc}$ C.B. that is "comparable" to ~~...~~ t. C.B. that one is using for this search. T. meaning of "comparable" will have to be expanded, hvr. [see 90.04 – .07 for a brief comment ~~...~~]

We want L^* to be such that t. $\frac{pc}{cc}$ of x ● w.r.t. L^* is ~~adequately~~ hy.

[Note: if L^* is "shortest code" it will be random — i.e. have no reggys!]

36 If x_i is a subcorpus, Then we code x_i via x_i, L_i^* (.30).
T. next "library" becomes $\approx L_{i+1}^* = L^* \frown x_i^*$.
mm. Hvr. .36 suggests a hole in this ~~...~~ approach. With .36, we are simply coding t. code of t. corpus sequencially in incremental batches. x_{i-1}, x_i, x_{i+1}, x_{i+2} ... etc.

38 Well! maybe that's not so bad! As befor, we code each x_i by first dering a cpm, then obtaining t. pc of x_i wrt that cpm. so t. squence of "good codes" $q_1 \frown q_2 \frown q_3 \cdots \frown q_i$

becomes t. corpus for predicting q_{i+1} — which is ≈ one of t. earliest approaches!

.03 Hvr., 88.01 ff & Rov 7.12–.40 in particular have refs to serious non-optimalities of this early approach

.04 It is possl. that in 89.33 use of t. shortest code (C.B. ≡ ∞) mite not be so bad, because cc comming in as $\log_2$ Acc. Hvr., certainly not always so! — i.e. cc can be exponential in (or greater) argument length

.07 for many functions.

.08:74.20

Re: Pm' v.s. R for seqs. in which one does not know whether t. seq. will continue. E.G. take a horse race. There is some probty that t. "3rd race" will not occure tomorrow. — But determining this probty should not be done by examining t. previous seq. of races. T. info. relevant in this case would be mainly that of weather, politics, probty of power failure, ect… & estimating t. probty of this race not occuring is best done w. info not in t. race data.

In general, this is probably often true, for most seqs in which there is t. probty of non-continu. to consider. ⟶ 93.01

.16

.17 TNG. SEQS: T. main problem in T.S. design for both induction & ≈NP (deterministic) problems, is to reduce "q" (of $2^q T$) to reasonable levels. I dont see any way to reduce "T": i.e. conceptually, I think of reducing T as changing t. nature of t. soln. — while I conceive of it's being possl. to recode any soln. so its dern. can be changed — yet retain t. same "soln". (so q = 1 (or perhaps even 0) is possl.).

This division into q & T may be really artificial — i.e. perhaps there is a way to look at it so w. t. "same soln", one can change T. — well, ordinarily, when one changes t. dern. of a gn. soln., T changes also.

.26 Also, consider a certain defined operation: like sorting a list in order of size. There are many ways to do this — some have much less cc than others, yet all have t. same (I, O) Relationship.

.28 So, in view of .26 it would seem that in $2^q T$ one could find ways to reduce both q & T & still retain what one would regard as "t. same soln". ▶91.33

.30:70.34 ⟶ That if one's Refrence Trunc. contains "All of t. info that one brings to t. problem" then (perhaps by defn.! ☺) t. $\frac{cc}{pc}$ search is ≈ t. best way ≈ or not far from "Best".
This view brings us back to t. discu. leading to 70.34!

.33:70.30 ⟶ Re: 70.30 on "1 shot learning" from a worked example of a gn. problem type. This is statistical info. TM is learning by induction. T. "laws of math" are just another set of v.g. rules that TM has derived. that give useful regularities in t. usual universe: That "N.P. problems" that have been correctly worked are merely part of some statistical data on t. universe. In this sense, then, N.P. probs are a proper subset of t. "prediction probs that TM normally works, & t. search techniques to find solns. are identical to that used for any other induction problem!

3.13.79 Lev........

.01 T. idea of 90.33 – .40 seems very impt. It really does seem to unify t. induction & "N.P." problem types. In a perhaps realistic sense, this is as it should be, since a human doesn't really know anything that is not a statistical estimate based on his previously observed time series. As such, any ~~soln~~ soln. to a problem that he may propose, is nothing more than a statistical estimate based on his previous experience — and this estimate includes as statistical random variables, t. meanings of all words, phrases, sentences and indeed t. statement of t. problem itself.

Another way to do "NP" probs w. an induction machine! One can assume a hypothetical freqs ∋ TM knows str. ~~ease~~ hy probty. that t. only acceptable seq. following: "X, A(·,·)" is a y ∋ A(x,y) = 0. / for solns.

.13 [redacted]

T. foregg. stuff is mindful of some other recent ideas I had on searching within gn. CB's. There, I had genzd. t. idea so that it included not only searches for good induction codes, but searches for (I,O) functions that would [redacted] maxz t. expected value of some Gorc. Try to find this stuff ... it was certainly within t. last few months, & it may well be that w. t. present approach I can devise a more direct soln. to t. more general problem of behavior optimization so t. concept of "Utilities" need not be specifically introduced, but would (perhaps) be invented by th. machine as a reasonable approxn. — — a pos oft of approached & par. cc.

I think one of t. ideas there, was that in searching for a good behavior mode, one would not include ~~exclude~~ certain kinds of behavior in t. search, & these exclusions acted like a C.B. — similiar to t. cc computation Bnd. Perhaps one could devise a method of ordering search ~~ocean~~ by some function of pc, cc & G (= Gorc). — Or, perhaps one would simply use a $\frac{cc}{pc}$ search (as we did in looking for cpm's) & chose our mode of action on t. basis of pc of cpm × pc of corpus wrt cpm × (G of that action) or whatever is relevant.

Perhaps first "maximize" G so t. sum of 2 actions = sum of G's: Then search in order of $\frac{G \cdot pc}{cc}$: or in way to get max

I also think that I considerd what happend to heuristics when these constraints were invoked. I may have written a review of this work, since I felt it was very impt. at that time.

Perhaps look at "RS" notes or a file closely related to "RS."

In General, it may well be that I should review many of my previous "important ideas & concepts" in view of $\frac{cc}{pc}$ search.

Makes Newton's & Einstein's ~~...~~ work in formulating general laws for t. universe. look like parochial exercises.

Makes Ein's attempt at a Unified field Theory look like a parochial exercise!

Resin = \$10/gallon.

10 lbs/gall?

\$1/lb.

1 ft³ ~ 63 lbs (P=1)

$\frac{1}{16}" \times 1' = \frac{63}{12 \times 16} = \frac{1}{3}$ lb.

P=1 ∤ 33¢ ~~at~~ for \$1/lb.

.33 : 90.28 One impt. possy.: To get a working TM model that isn't so bad, it is only necy. to make t. Tng seq. & t. Algm* ∋ t. T. i q of ~~...~~ 90.17 ff are small enuf to be findable. Alg* could, perhaps, involve only rather simple regy detection — such as Bernoulli freqs, & simple definitions & perhaps that's all. Of course w. a better Alg* TM would be better & need less carefully designed tng. seqs., & would get better pc for a gn. cc.

So: Try to integrate all of this into a v.g. review

.01 Related to V.H.M.: Is there any bound we can make on how much smaller $2^{-l_0(x(n))}$ is

.02 than $P'_M(x(n))$ or $R(x(n))$? — in particular, is a ~~constant factor~~ adequate? $l_0^{x(n)}$ is t. length of t. shortest code for $x(n)$:

Say t. corpus was created by some cpm, $P_0(x(n))$. Then if $l_0'(x(n)) \equiv IP(-\lg P_0(x(n)))$, [ ] ≡ int. part of

there is a positive constant $k \ni l_0 < l_0' + k$.

$l_0'(x(n)) \equiv IP(-\lg P_0(x(n)))$.

Also $-\lg P'_M < -\lg P_0(x(n)) + k$

If .01 - .02 is false, then for any b, (no matter how large), there exists a $x(n) \ni P'_M(x(n)) > b\, 2^{-l_0(x(n))}$: This means that there are always $x(n)$'s $\ni$ t. codes other than t. shortest one, have arbly large relative total wt.

If P_0 created $x(n)$, then for large n, "almost always" t. shortest code for $x(n)$ is about of length $k + l_0'(x(n))$. → then there always exists a code for $x(n)$ of length $= k + l_0'(x(n))$.

Whether P_0 created $x(n)$ or not, there always exists a code for $x(n)$ of length $k + l_0'(x(n))$. [P_0 can be any cpm]

If P_0 created $x(n)$ then for large n w.p. ~ one: $-\lg P'_M(x(n))$ is not much $< -\lg P_0(x(n))$.

i.e. is probly $\approx -\lg P_0(x(n)) + k$ & is $\lessapprox -\lg P_0(x(n)) + k$.

there exists a code for $x(n)$ of length $IP(-\lg P_0) + k + 1$

So $l_0 \lessapprox IP(-\lg P_0^{x(n)}) + k + 1$ (length of shortest pgm.)

.20 This → if $-\lg P'_M \geq -\lg P_0(x(n))$

seems to prove about as much as can be proved.

Then $l_0 \lessapprox IP(-\lg P'_M) + k + 1$

This says that $2^{-l_0(x(n))}$ is within a bnd const. factor of $P'_M(x(n))$.

$2^{-l_0} \geq P'_M(x(n))\, 2^{-k-1}$

I.P. $(-\lg P_0) + 1$ is t. length of $\approx c'$ (t. $\approx$ willis of w.f.o.r machine assoc. w. P_0) shortest pgm. for $x(n)$.

.25 Is $P_0(x(n)) \geq P'_M(x(n))$ "almost always true" for very large n? Perhaps for ~~certain~~ ergodic or other special cases. Or even knowing $P_0(x(n)) \geq P'_M(x(n)) \cdot 2^{-b}$ was true for some fixed ~~b~~ would be adequate.

This is t. critical, weakest pt. in t. argt.

I guess .25 would be true: otherwise one would gain more w. P'_M as a betting strategy than w. P_0 (if t. seq. bet on were truely generated by P_0).

Hvr., I'm really not so sure of this!

Look at, more rigorously, t. statement of .25: The probly that it is true as a funct of n, say, or $-\lg P_0(x(n))$ or $-\lg P'_M(x(n))$. etc.

→ .25 is not true as stated! P_0 can be very small (even zero) for certain set of seqs of measure > 0 (wrt t. uniform measure)

— but P'_M will always be signifly > 0 for any such set.

→ Hvr, .25 mite be $\approx$ true for all sets that have measure > 0 wrt P_0 — which is empt.

Hvr. if VHM $\approx P'_M$, then how come the summation over all codes as in Maxm gives signifly difrnt. varc. than just using t. single "best fit" code? T. Max m. soln. is obtainable by assuming uniform a priori. for t. coefs (& perhaps for σ^2 also). Even t. "square" $\frac{\sigma^2}{N}$ soln. is much difrnt from t. $\frac{\sigma^2}{N}$ soln.

2.71828 1828 .71828 18301 .81828

3·21·79 Lav:

.01 : 90.16 On **R V.S. P_M**: I think this is t. most impt. argt.: In almost all, if not all, applicns. of proby, one is using it to make decisions. If utilities are usd, (& perhaps in most other situations) one is only concernd w. t. Ratio of 2 probys (like $\frac{R(x_i)}{R(x_j)}$): ~~This quantity~~ when R_T is usd to approx. R, this ratio is not monotonic in T. This monotonicity — which narrows t. value of R somewhat — is usually not of importance in real problems, since ratios of R are needed.

I doubt if any measure exists that will give optimum values for these ratios. That no optimum normed measure exists, suggests that it is impossl., & that use of R gives one false confidence in its accuracy.

On just how t. TM works:

2 possys to consider:

.20 1) 89.25 – 90.03: simple sequencial coding, but backtrack only to t. most recent Sc_i boundary. Hvr., note t. impt. modifn. of 89.38: t. Sc_i's are very lg., & we code

.22 them by derivg a cpm for each. L^* is t. seq codes for theses cpms: $q_1\ q_2\ q_3 \dots$

2) The Alg_i approach: Each Alg_i has as input, a) t. Sc_i to be coded b) t. aux info available for t. coding such as t. set of previously usd Alg_m's & t. definitions they used.

T. output of Alg_m is an induction code (or equivt.) for Sc_i (& perhaps a distribn on t. contn. of Sc_i — either as a function or a Mt. Carlo simulation or whatever is desired.)

Methodologically: ⓐ Only induction probs need be considered: see 90.33 — 91.13

ⓑ A way to derv. just what is needed for t. tng. seq. & TM construction:

Work backwards: Say we have a problem Q_i w. soln., X_i. **Find** a way for a human to derv, X_i. Take t. concepts usd by t. human; Definitions of these concepts then become goals for t. tng. sequ. up to problem Q_i. From these goals we work backwards, as befor to find goals for t. tng seq. up to that pt., etc... untill we are able to end up w. a reasonable no. of concepts that seem reasonablely "primitive". (tho they don't have to be "intuitively primitive" to humans! They just have to be a "basis" set of concepts, so that practically any complex concepts can be derived by

.36 minimal tng seqs. from them.

If [c_i] is a set of primitive concepts ∋ it can generate ~~proportional cc~~, each hy level concept that is accessable to human H_j, with a cc that seems to be proportional to t. cc usd by H_j for those hy level concepts: then t. set [c_i] is an important kind of partial dern of H_j

3.23.79 Lav

My impressn. of t. relatively critical ~~ideas~~ toward TM in t. forgo. pp.

1) Use of $\frac{pc}{cc}$ search having known upper bnd. for total search time of $2^q \cdot T_q$.

2) Breaking up corpus into subcorpi (Sci's) so that (a) smaller Sci's have reasonable search costs (b) by putting them in tng seq. order, q's & T_q's for each Sci can be much reduced by using concepts, defns, srtns learnd in previous Sci's.

.10 3) Make Sci's large, but ~~use~~ code each by ~~leaving~~ a cpm dcrn followd by t. pc of Sci wrt that cpm. Since t. cpm's have prefix code dcrns,
.12 t. search method of 1) ($\frac{pc}{cc}$ search) is appropriate. → 98.01 ← impl ← objection to this idea

4) Only induction probs need be considerd. NP probs & any other well defnd math probs can be viewd as special ~~kinds~~ kinds of induction problems. (See 90.33 – 91.13 for ~~expl~~ discn.) It will be useful to include well defnd math probs (≡ BW probs) in t. tng seq., hvr, their soln methods will
.20 be ≡ that for induction probs.

.21 5) The $Algm_i$ idea: It may be possl. to construct a TM based on .01 – .20 only, but a much ~~[illegible]~~ improvd device uses t. $Algm_i$, Alg^* idea.
Here, for Sci, we try to find a ~~[illegible]~~ algorithm, Alg_i. Alg_i's args are ~~[illegible]~~ Sci; plus any previous info we want to include; like defns & srtns & concepts usd in previous subcorpi & t. previous set of $Algm_j$ (perhaps). Its outputs are (a) A short code for Sci or set of codes or a proby assignment
.26 to Sci (b) A Monte Carlo prodn for t. contnn of Sci, or a distrbn. funct for such contnns or equivt.

The big idea here is that Sci is not a "sequential" input to $Algm_i$ — i.e. $Algm_i$ can look at t. entire Sci before trying to devise ~~[illegible]~~ code for it.

∴ $Algm_i$ is not a process for t. Sci input.

Also, note that $Algm_i$ has 2 kinds of inputs: (a) (a1) Sci & (a2) info usable to reduce t. pc of Sci's codes: This is info like ~~previously used~~ srtns, defns & concepts usd in previous Sc codes.) (b) Info that helps $Algm_i$ ~~find~~ find good codes for Sci. This is info on good search techniques obtaind by experience of previous $Algm_j$. T. problem is to find short p ∋ U(p, (a2)) → (a1) Sci

.36 6) The $TM_1 = TM_2$ idea: TM_1 is a machine that does T.S. extrapoln. T. idea is that since TM_1's problem is a univl. model for all induction, TM_1 can solve t. problem of improving itself. Th. (b) output of .26 is to implement this. $Algm^*$ was that of a t. device that creates new $Algm_i$'s: t. (b) output of Alg_i could be made a trial for $Algm^*$ improvement.

T. Big change since: See Rev 8.03 ff: we use $Algm_j$ — which is $Algm_j = TM_1$ at time j. If these same 5.0 as desrbd here, but $Algm^*$'s job is to improve $Algm_j$.

t. search algm. at time j. $Algm^* \equiv TM_2$.

3 23 79 Lav:

.01 7) Discussn. of inadequacies of t. forgg. ideas & suggested improvements (including a fairly non-cl. system): See Rev. Lav. 7.30–.40 for biblig on this; Lav 77.14 – 79.40 Has much impt. matterial, but t. rest of t. stuff refrenced
.04 is also of import.

.06 8) The idea that if pc's are defnd wrt some finite CB., Then t. pgms will include heuristics. (Here heuristics are pgm techniqes designed to obtain results using less cc). (pc's defnd for CB do not include heuristics).

At first Glance, There would seem to be 2 kinds of heuristics: — See 96.04 for defn. of Ⓑ.

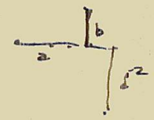
a) Production heuristics: we want to find shortp→ U(p, Ⓑ) → Sci.

The features of a pgm, p, That allow it to satisfy ↗ & have small $2^{\ell(p)} \cdot T_p$ are heurs of This sort. [defns & strings that p can refer to are heurs of This sort. Sci is not observed for These kinds of heurs.]

b) Search heuristics: Techniqes of chosing P so That p's are found satisfying ↗ & having small $\ell(p)$ in as small a total time as possl. [Note: This can include redesign of hardware]

.20 One possl. way to clz. This: for Algm: (Sci, Ⓑ) ⇒ p we want $2^{\ell(p)} \cdot T_p$ to be as small as possl. So This is a Gorc for Algm's which Algm ($\equiv TM_2$) will try to optimize. ∧ T_p is t. actual time needed to find t. finally accepted pgm, p. The features of Algm: That move it in This direction are "Search heuristic".

.25 Not exactly; we want $\frac{cc}{pc}$ to be min: i.e. mmze $\frac{\Sigma T_p}{pc \text{ of } Sci}$ ← total cc to get all that total pc. ← total that was obtand by Algm: as cc of Tp.



See Rev 3.01–.40 for more discn. of Heuristics.

$a^2+b^2 =$
c^2+d^2
$=(b+d)^2$
b^2+d^2+2bd
$b^2+a^2 =$
c^2+d^2

7/23/79 Lev:

Discn of 94.01 – 95.40: I think 94.21 (Algm_i idea) & 94.36 (TM₁ = TM₂ idea) contain impt. indicators as to just what my confusion is: One big idea however, was that Algm_i could look at t. entire Sci before trying to code it.

.04 I think that Algm_i only has 2 kinds of inputs: (a) Sci (apsi) (b) all other inputs (apriori):

we want / shortest $P \ni U(P, (b)) = (a)$ [Sci]; Here (b) can be any real info other than Sci; i.e. (b) must be "apriori" info (obtained before Sci was seen). (b) can be (redundant information-wise for $CB = \infty$), but not necly redundant for finite CB's. i.e. (b) can include t. same data in various forms — each form of which may be particularly appropriate to different kinds of probs.

.15 A big confusion in my own mind is separating Algm_i from Algm* (Algm* is supposed to create new Algm_i's). *

$\sqrt{x + \sqrt{x + \sqrt{x}}} = x$

1 One approach: say Algm_i is ME. In this case, Algm_i is fairly constant — changes very slowly w. i.

$f(x) = \sqrt{a + \sqrt{x}}$

if $f(f(x)) = x$

& f is monotonic ↑

Then on three f(x) = x is a soln, or no soln.

.19 Slide side note: If t. (b) input includes t. total previous corpus, Algm_i can try

.20 finding new regys in t. entire corpus, if this would seem to help coding Sci.

.21 One possl. view of t. Algm_i, Algm* system: ← looks v. G.

(Algm_i is more correct, since in this system, t. Algm isn't necy changed w. each Sci). Anyway Algm_i is at any particular pt. in time, TM's search algm (or whatever method it wants) to find good codes for Sci. see .04 – .15 for more on (b) input

.25 It does have t. 2 inputs types of .04. T. (b) input includes all past info that we see fit to save — paths of calcns., defns, status, concepts, info about Math & statistics & Hdwr., if we like. Also, all of t. previous Raw corpus; $Sc_1, Sc_2 \ldots Sc_{i-1}$ to use

.28 Part of Algm_i's pgm is to determine just what part of this info to use. Algm_i's output is a seq. of pgms for Sc_i which should have hy Σ pc.

At first, (small i), Algm_i will probly be a simple L-search. Later, it will use various hour tricks suggested by Refs of 95.01 – .04.

T. View of .21 does seem reasonable. It is Algm*'s function to look at t. entire corpus & t. paths of compns & all that's going on in general, & from this to try to devise a better Algm_i. Algm* is TM₂ ← See 95.20 – .25 for TM₂'s Genc.

.36 Start out, perhaps w. t. search method of 94.01 – .20

So write a brief, detaild review of this all, in view of .21 – .36!

Then read t. older stuff on Algm_i, Algm*, & see if t. "new" approach answers all Q's. (unless t. "new" approach is = some of t. old ones!)

.01 : 96.21 – ~~thru~~ .36 does seem like a reasonable model for T.M. & a good statement of what t. Algm$_j$, Algm* system is:

T. corpus is dscrbd in 94.01 – .20^{5} [Hvr. note objections & changes in 98.01ff See Rev 8.10

Algm$_j$ (Sc_i , (b)) → [p_{ij}] is th. current (at time = j) search algm of t. system. Input Sc_i is t. sub corpus: input (b) is any other info. That is, truely apriori to Sc_i, that has no possy. of being spuriously related to Sc_i. See 96.04 & 96.19 – .20 ← impt & 96.25 – .28 for discn. of what (b) may contain. Output is a seq. of pgms for Sc_i. We want

.15 Algm$_j$'s to give max Σ pc for small total cc; max $\frac{\Sigma pc}{\Sigma cc}$

Here t. sums are taken over all outputs/ for gn. input pair up to a gn. time.

Most search Algms. will continue putting out codes indefinitely.

Algm$_j$ can use info.m Sc_i only to order t. trials for finding short p ∋ U(p, (b)) → Sc_i. t. separation of "ordering t-trials" may have other equivt. forms.

It is t. work of Algm* to improve Algm$_j$ wrt. t. Gore of .15.

The initial Algm$_j$ will probly be ≃ t. simple L-search method of 94.01 – .20. Initially, I will be Algm* (≡ TM_2) & improve Algm$_j$ as much as I can. ← [Note inherent "biased search" in me being TM_2 ! see 98.20]

It may be possl. to use a simple Algm* after Algm$_j$ gets good **enuf**. **Then** later, of course, **let** Algm*'s work be done by Algm$_j$.

See Refs of 95.01 – .04 for v.g. suggestions on how to improve this system.

Actually, there is no real hurry to get $TM_1 = TM_2$ — This shouldn't be done until TM_1 has been devising extremizing pgms of t. proper level of diffy.

.01: 94.12: On using large Sci's & coding them by (cpm$_k$ dern) x (dern of Sci wrt cpm$_k$)

The t. cpm's have prefix codes: ① [redacted] They are not enumerable (tho they are countable): so o.u. do a partial enumeration — (i.e. listing in which some entries will not converge).

② T. listing in prost order is really not very relevant to obtaining good bc's for t. [redacted] Sci itself. [redacted] — Certainly t. aprip ordering of t. cpms is not at all constraind in any way by t. nature of Sci itself.

.11 (So: A good Algmj should somehow constrain the cpm search on t. basis of observations on Sci (presumably inexpensive observns!)).

In general, Algmj looks at Sci & must decide what general type of code to use: t. "cpm [redacted] followed by code of Sci wrt that cpm" is only one of many possl. types: it is, of course, particularly common for numerical T.S's. Even in [redacted] this type of coding, t. general nature of t. cpm to try, must be decided upon by Algmj.

This deciding can be done by making various obs. on Sci. Devising such obs. & relating them to search methods to be used is a normal induction problem.

.20 An impt. dif[f]ty [redacted] assoc. w. .11: Any such constraint of t. search biases t. search & biases t. result induction results. By suitably constraining t. search one can get practically any desired (spurious) [redacted] induction result!

.28 Hvr., when I am no longer TM$_2$, presumably T. Algmj's would be selected to get max pc per cc & ∴ would not bias t. results, since this ↓ pc. i.e. [redacted] features of Algmj's would be used that have been found to give hy $\frac{pc}{cc}$, & this would move away from [redacted] biasd results (which have lower $\frac{pc}{cc}$).

So: one big advantage of getting TM on "its own" (w.o. me as TM$_2$) for a long time: is that eventually, it would get rid of t. bias I introduced as TM$_2$

.30

Anyway; So consider Algs as $\boxed{97.01 - .15}$, but t. [redacted] coding methods are not nearly [redacted] cpm$_x$ probly of Sci wrt cpm$_j$) — type. Algmj is, hvr., a complete search algm of some sort. It does look at Sci w. various obs., & decides what kind of search to do. At first, we will probably want some constraints on what kinds of obs. to use on Sci so t. search wouldn't be biasd. Later, we will probly be able to relax such constraints because of .28 — .30

At first, Algj will be rather simple $\frac{pc}{cc}$, L search. I will try to get t. previous Sci's coded so that t. defns, setus, concepts therein are as easy to use in subsequent Sci's as possl.

Firstly: Using a very simple search Algm. of this sort, it would be possl. to design Tng. sequs that it could solve at reasonable cc — but t. "conceptual jumps" involved would be small.

.03

So: Design such a T.M.; & make some tng seqs for it. Then introduce various improvements: like better search ~~algms~~ tricks, etc.

Also see if, w. a minimal Algm. type, it would be possl. to do a $TM_1 = TM_2$ trick after a suitable tng. seq.

Look into just what kinds of seqs. & heuristics such a TM would have ① ease in devrng ② difty in devrng.

Secondly: **Take** a tng. seq. in which a seq. of humanly accessable (cc-wise) solns. ~~would~~ seem to exist. **Look** at what modifns of t. forgg. T.M. (.01–.03) would have to be made so it could work that tng. seq.

W. about t. same heurs, concepts, etc.

Take specific probs & see just how t. TM would have to be augmented to solve them: e.g. ordinary linear regn., Maxm., Various kinds of non-linear Regn; various computational speedups, like Fast Fourier xfm; Perhaps devry of fractions, neg nos., irrationals, complex nos. . . .

30 rps
24
960 bpps.
96 ips.

It would seem that t. TM of 97.01–.05 would be good to start w. — that any problems that arose would involve finding some abs. having certain properties ("B.W." or "Gray") & that all such problems are solvable by $\frac{pc}{cc}$ search, preceded by a tng seq. that is capable of ↑ t. pc a/o ↓ t. cc. — and that's it! i.e. t. tng. seq. must make t. soln. of t. problem more likely a/o make that soln. take less cc — that humans cannot solve such probs unless their $\frac{pc}{cc}$ is in t. acceptable range either.

.01 On t. optimally of L's search method (usig ordering $\frac{cc}{pc}$): Plan for proof: Suppose we have some other search method/ S_2 over some problem domain, & over that domain S_2 yeilds ont. average, shorter search time than S_1. ($\equiv$ L search). Then (conjecture) we can use S_2 to define some new ragys over t. domain of problems that have not been included in S_1, & thereby reduce S_1's/search expected

.15 time to < that of S_2.

If L search is optimum, I think that .01–.15 defines one impt. sense in which it may be optimum. This "definition of t. threm" ~~is an~~ can be an impt. step toward its proof!

In one sense, we know L search is optimal: i.e. it is within a constant factor" of any optimum method. We want to show, hvr., that this "const factor" is not very large.

An approach; perhaps show that there is a mapping betw. search methods & proby assignments, such that: If one had a new search method that was better than S_1 ($\equiv$ L search) the probys it mapped into would have to give a higher pc to t. corpus than those gn. by P_M' — This latter is not impossl., but it means that (I think) S_2 gives a better apriori than P_M', & should be used. I don't think that the new apriori can be much better than P_M', but it may be better by a constant factor — & I guess I want to show that t. const. factor must be small.

.28 Another way of looking at this: If we did get a apriori better than P_M', then this mite imply more aprip info about t.

.30 corpus than was gn. by M (of this P_M').

[.28–.30 is simply another way of saying that P_M' is the Best — that anything better implies better ($\equiv$ more) apri info. — info. from acquirt to better M"..

(1) If we confine ourselves to indep trials (non-sequencial search — the only info available from a trial, to use in t future trials, is that that particular trial can be avoided — i.e. trials w.o. replacement. This is ordinarily not very much better than trials with replacement), perhaps its not hard to show L-search is near optimum.

(2) In t. case of sequencial search (learning from any info generated in previous trials is legal), t. problem seems to be much diffnt. Hvr., it may be possl. to express all seql. search probs. as "indip. trial" problems.

.04 So: T. first trial is t. same as for "indip trials": T. 2nd trial uses ≈ $\frac{cc}{pc}$ ordering of trials but both cc & pc are for t. new trial in terms of 1) apri info 2) t. info (codes, srtus & success or failure) of t. t. first trial.

T. third trial is similarly based on apri info as well as info generated in all previous trials.

"How much time should be spent in t. search on looking for rugys in t. search info thus far"? Well, if ½ of one's cc is usd this way, t. optimality of t. cc of t. entire method should be off by a factor of ≤ 2.

This is something like Chaitin's Complexity (X/Y) which is t. min code for X given t. min code of y. Hvr. here, we have a set of codes (not necly min., but somewhat short) for y, & we are trying to get as "good as possl" (using $\frac{cc}{pc}$ criterion) codes for X w.r.t. those codes for y.

---

(3) T. analysis of "sequencial search" of .04 ff is for a simplified form of t. problem: There is no "planning" allowd. One tries to make a best choice each time; No "experiments" are allowd. A more truly optimum search would have a more general RTM-type goal. (w. h>>1)

.25 Actually, .04 ff as is, is wrong. Say we have an early trial that generates t. corpus w. small pc & large cc; ~~Well - Actually~~ then we can use info about this trial (i.e. t. corpus itself) to generate t. corpus at pc ≅ 1 w. low cc. So I (previously) had t. idea that there Hvr. must be some restrictions on how info out t. past could be usd — & I did formulate this restriction w. some exactness (perhaps in O.S.TM — possible in "Levin").

.35 Hvr., in t. present case, it is impossl. to use such info about t. corpus if L-search is usd, because no earlier trials ever generate t. entire corpus. T. first trial that succeeds in generating t. entire corpus wins — & that's t. end of t. search!

.39 Maybe not: I think L-search can first obtain one code w. $\frac{cc_1}{pc_1} = \alpha$, then continue t. search & obtain another code

w. $\frac{cc_2}{pc_2} = 2\alpha$; & Its possl. that $cc_2 = 4\, cc_1$
& $pc_2 = 2\, pc_2$.

So t. second soln. would be of interest.

or , even $\frac{cc_3}{pc_3} = \alpha$ with $cc_3 = 10\, cc_1$
$pc_3 = 10\, pc_1$

So : restrictions of 2.35 – .39 may still be nec.