[I> 9681 ⇒ SITS164 t. last p. work box?
In t. forgoing pp. there seem to be no page refs. to t. work of 1981! Sum. gap
[in work on TM. — work on Sinclair computer?

1.30.82 Ts

.01 In work on Try. Seq. for Arith, alg: I ran into a sort of bottleneck involving learning to Evaluate functions. I could simply assume that t. Thy was adequately solvd & continue — Then come back to it (later if it still seemd vital).

.07 ((The actual form of t. problem, hvr., suggested that this particular bottleneck was a rather General sort of problem.

.10 An abstract descrn. of t. TM Seq prob: [P_i^0]

We Give T.M. a set of strings (≡ Problems) as input. It Then tries to find a Seq. of symbols S^0 that is a soln. to this set of probs. Eventually 1 or more (S_0^0, S_2^0 ..) are found.

We then Give T M t. now set of problems [P_i^2] That require a soln. that is a seq. of symbols S^1. T.M. searches for solns that are a min. distance from S_0, S_1, ... i.e. S^1 has a small complexity (given S^0...). These complexitys also take cc into account somehow. H (S^1/S^0) is minimal (w.r.t. considerations of cc.)

.17 S_i^1 solves both t. old [P_i^0] probs & t. new [P_i^1] probs.

H (S^1/S^0) ≡ min|r| ∋ $U(r, S^{0*}) = S^1$; S^{0*} is, in chaitin's defn, the shortest code for S^0; In my formulation S^{0*} will be t. shortest code(s) we've yet found for S^0. Also, my entropys are sums of terms involving various ways to code for [S_i^1], that I've found thus far.

.27

Anyway .10 — .27 would look like a complete soln. to t. problem. If an approxn. to a Kusp machine is used, then we should — w. L such, we should, eventually, get something like optimum solns.!

Well, O.K., so I tried to write a Try. Seq. w. that idea in mind. What went wrong?

Perhaps a not very unusual thing: That what I That was a reasonable soln. to a certain induction problem turned out seemd not to be a v.g. soln. — i.e. not better (& perhaps worse) than what looked like a

.36 clearly not-so good soln. The bottleneck (see .01–.07) was that t. older, not so good soln. was: "Do a substitution 10 (say) times." The [$S_{\alpha,10} \equiv S_{\alpha,10}$] soln that seemd much better was "Do a substitution until its no longer possl." t. (Recursive Soln.).

I vaguely remember t. diffy being this: That $S_{\alpha,10}$ was obtaind after $S_{\alpha,9}$, ect... In general $S_{\alpha,N}$ was obtaind after $S_{\alpha,N-1}$ was obtaind. The additional brain of $S_{\alpha,N}$ over that of $S_{\alpha,N-1}$ was rather small, and was less than t.

b cost nec'y to go from $S_{\alpha,N-1}$ to S_β. Hvr., to go from $S_{\alpha,1}$ to $S_{\alpha,N}$ took more b cost than to go from $S_{\alpha,1}$ to S_β .... if N was large enuf.

—— So, by giving T.M. a typ. seq. for which t. successive solns. $S_{\alpha,1}$, $S_{\alpha,2}$ ... $S_{\alpha,N}$ were acceptable, T.M. was prevented from discovering t. better soln., S_β.

.06 Q: Is there some genl. way to prevent ~~this~~ this terrible thing from happening ... so that S_β could be derrd?

Perhaps this difty is a necy evil assoc. w. giving T.M. too small steps in its typ. seq. —— that by using small steps, one can lead t. T.M. into any kind of soln. one wants — w. small likelyhood of TM ever looking for a better soln.

One possl. way to deal w. this: After TM has obtnd $S_{\alpha,10}$, say: ~~to go back to t. site~~ for T.M. to try to solve t. present problem (that has $S_{\alpha,11}$ as soln.), by denying all of its training from $S_{\alpha,1}$ to $S_{\alpha,10}$, & just use $S_{\alpha,1}$ .... would amount to a kind of "Back tracking".

When should T.M. try such Back tracking?

It would seem that in general this difty would arise, & one would occasionally want to see if one could get a better soln. by ~~[illegible]~~ denying part of one's typ. seq. & starting out from (a certain degree of) "scratch".

"On another level" This may simply teach us that we should'nt make t. c.j.s. betw. examples too small!!

.27 I guess my real problem here is: ① Is [redacted] my typ. seq. difty derbd from 1.36 to 2.06 a Real problem that has to occur occasionally, or ~~is~~ [redacted]

.29 ② Is there some way of dealing with this typ. seq. so that neither it nor difftys like it, ever arise?

Say it's a real problem that occasionally arises! It arises as a consequence of t. Typ. Seq idea & t. idea of wanting each ~~soln~~ soln. trial to add (at least) incrimental complexity (1.17. — .27) to t. previous valid soln.

In truth we want a soln. that has min complexity wrt. t. entire previous corpus — . . indep. of previous solns. of parts of t. corpus.

Say s^1 s^2 s^3 s^4 are solns. to successively longer sequences of t. corpus. Now, for trials of s^5, we would normally start w. s^4 & try to minimally modify it. This would

.017 tend to get us some kind of soln. at min cc. → Hvr, if we start out w. soln. S^3 (which in a sense, is "smaller" than S^4) & try to get S^5 as a min jump from it, our soln. will take more cc, but will, in general, be capable of being a better soln. If we have ~~time~~, cc left, we can even go back to S^2, etc.

Note that all of this (.017 ff) is backtracking. → see .25 & .32 for another way of looking at this "backtracking"

Another view of the diffy is more like 2.29: Here we look at the seq. of solns $S_{\alpha,1}$, $S_{\alpha,2}$, $S_{\alpha,3}$ .... $S_{\alpha,10}$ & we induce $S_{\alpha,11}$ from that sequence: But it might also be possl. to induce the recursive soln., S_β by looking at this seq. of solns (— The idea of "coding is Recoding" at a hyper level).

It would ~~seem~~ seem that in looking for a "less el." code one ~~could some~~ should also take advantage of all of the info in $S_{\alpha,10}$, $S_{\alpha,9}$, $S_{\alpha,8}$ .... $S_{\alpha,1}$ — not simply discard the more recent info. Also, the ~~sequence~~ earlier sequences of solns. should be used, not just one of the relatively recent ones.

In general, one's strategy for search will depend on how much cc one has available — but perhaps there is a genl. strategy that automatically .25 finds the low cc solns. (first of not nearly hypc), then gets hyper cc solns of progressively larger pc, later.

The strategy of trying for minimal incremental cost onto previous solns. is extremly el. I should perhaps find a better/more general way to look at this, so less el. solns. are automatically possl.

.30 One approach is to see what humans (I) do in a typ. seq. like the forgg. to avoid simply "sticking to the track" & [illegible]

.32 → A possl. approach to .25: The reason the student uses $H(S^N/S^{N-1})$ for search, is that using $H(S^N)$ would be too [redacted] expensive: i.e. the solns. obtained have excessive cost. By using $H(S^N/S^{N-k})$, approaches $H(S^N)$ as $k \to N$, so one could use $H(S^N/S^{N-1})$ for search; then after a soln. is found, use $H(S^N/S^{N-2})$ .... down to a $H(S^N/S^{N-k})$ for which the soln. takes as much cc as one can afford.

2.1.82 TS

Well: 3.32 isn't bad for a perhaps "workable" soln. I still don't feel very happy about it. Try t. "human" approach of 3.30.

Just how does a very brite human go about avoiding t. "stick to t. road" trap in learning?

.05 → Well, a very brite human would note simple regularitys in t. sequence of solns.; $S_{\alpha,0}$, $S_{\alpha,1}$ .... $S_{\alpha,10}$ ....

Also, it would be noted that repeated applicn. of an operator, yeilding t. same thing, is unnecy & time consuming — so one would naturally want to stop this applicn. as soon as possl. ... i.e. when t result became invariant under t. operator.

There are, then, 2 impt. things going on: ① — 2.27 : say that occasionally one will run into this diffy using a tng. seq. w. too small cj's.

② The possy. of using reasoning like .05 to get much better learning.

---

①Note that .05 doesn't simply look at t. sequence of solns, $[S_{\alpha,i}]$, but it must actually watch t. solns. being computed, to realize that much time is being wasted. I guess this mite be t. big difference betw. a pure "induction" machine that codes & recodes w.o. noticing how things are done & how much time is spent on various tasks: as opposed: to a "practical induction machine" (perhaps employing Kusp m's) that is as much interested in t. mechanics & cost of soln. as it is in their pc's. $H(x,y) = H_x(y) + H(x)$. $H(y,x) = H_y(x) + H(y)$.

I don't think this is t. same as TM_2 (but w. a Kusp machine & assoc. methodology maybe t. idea of a TM_2 isn't so distinctive from a TM_1!). 25 32

This idea of "watching a soln. in progress" & improving it is something I do well in practical computations w. real computers. Can I mechanize this ... Can I genz. it? ~35 pp 7 d 5pp/d.

The particular tricks to improving this seq. of $[S_{\alpha,i}]$ solns. seems very impt. & common: Essentially, .5 m baud = 1 byte/m [illegible] ~ 27 T states

.30 ① extrapolating t. seq. $S_{\alpha,0}$, $S_{\alpha,1}$ ... $S_{\alpha,10}$ .... by simpler ~ Bern seq. means ② recognizing t. time wasted in repeating an operation that leaves t. arg. invariant ③ possibly looking for stop rules in t. situation of .30.

Mite it not be worth while to put them in Ad-hoc?

— well, yes, If it were clear that there was a mechanism within t. system that would have found these things eventually anyway — ... w. a suitable tng. seqs.

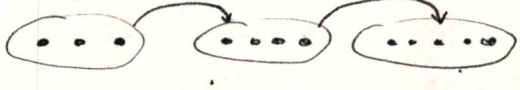
2.2.82 TS

I think both mechanisms [the backtracking, & Less El. Method; & th. Figuring out more efficient ways to do eg. gpm] should be somehow included in TM. — These are both impt techniques to improve prob. solving & TM. must be capable of using both.

W. th. Less El. backtracking: I could simply build this in — but I feel that I don't understand th. genl. prob. well enuf .... that there may be a better, more genl. way to do this.

One way is to Try th. probs of th. trg. seq. in diffrnt. orders. One can think of th. trg seq. itself as an unordered list of problems. Then th. ordering of these probs gn. by th. trg. seq. is some kind of aux info — that could or could not be used ... as desired. Normally, for quick solns, one uses th. info in th. form of $H(s^N, s^{N-1})$ & srch — but for better solns, perhaps at hyper cc, one should use other means as well.

Th. "backtracking" method can also be looked at, in this case, as simply one way of $\uparrow C(s)$, by grouping all "similar" problems together, & having th. "conceptual jumps" occur only betw. sets of probs of th. "same type" — (of course one must be able to recognize probs as being "of the same type". — which is a serious problem).



Another common & very impt. method of improving "solns." is to look at th. set of solns. to a set of probs. & try to express this set of solns. more compactly .... in terms of some new & efficient concepts. This is th. "coding & recoding" idea .... but we could also include cc considerations

How fast is my TRY GET ppsn? How long does it take to go from th. start pt. to th. pt. where it starts printing 2 char?

150 bytes
~ 70 insts at 10 T each = 700 T =
$\frac{700}{2.7 \text{ MHz}}$ = 400 μs.

3.3 ms/char.
80 ~ 120 p/min.

1 liter:
1 g = 4 w.
1 cal ≈ 4 w sec.
80 c = 320 w sec.

1 liter =
320 kw sec.
$\frac{320000 \text{ w sec}}{3600 \text{ sec/hr}}$ = 90 w. hr/liter

= .09 × 6¢
~ .5¢/liter

325 mw.
6V
2W

$\frac{4}{7} = \frac{6}{?}$

600 ma
2V =
1.2W.

$\frac{6V}{.2} = 8$

2.9.82 T.S.

Seems to be 2 diffrnt problems involved in that work on learning to evaluate alge Expressns.

.03 1) By making CTS small, i asking TM to try to find a soln.

→ $H(S^{N+1}/S^N)$ { S^N is t. soln. up to & including t. N^{th} problem } is min —

using Lsrch —— we get non solns at acceptable (low) cc,

but may ~~[redacted]~~ can be very El.

{ 2) We ask then, what is a less El. way to do t. search, if we have (some more) cc available? If the problems come in sets that are sort of similar, (say ~~[redacted]~~ a_1, a_2, a_3 .. are t. numbers of t. first problems in t. first, 2nd 3rd.. sets of probs.)

— then have TM search for t. soln. S_{ar} using minimal

.19 $H(S^{ar}/S^{ar-1})$ & Lsrch.

3) Assuming t. problem of .03 – .19 is real & that it does sometimes occur & that t. Eval " training seq. that I have is an example of it! There is another poss. approach to improving T.M.!

.23 That TM should watch t. various solns. being computed & will see that there is ~~[redacted]~~ an impt. simplifn. that can be made! i.e. that when an operation leaves an operand invariant one can save time by not repeatedly applying that operator. I.e. if $\alpha^{N+1}(X) = \alpha^N(X)$ then (α operator, X string.)

$\alpha^M(X) = \alpha^N(X)$ for $M \geq N$ & we can save cc. w. this substn.

recursp. Machinery.

Another way to deal w. this is for TM to look for stop-rules that tell when to stop t. reiteration of an operation or seq. of

.30 operators.

A general approach to .23 # –.30! That when I work probs. or write complex solns. to probs, I watch myself work them & find impt. simplifns. That sometimes actually solve t. probs very quickly. Can I get TM to do this as a matter of course?

2.9.82 T.S.

One difficulty w. the small cjs. "el." soln. is that this T.S. may be specifically pointing toward this particular soln. I had previously thot that any soln. would be accessible if its CJS was < the least threshold being used! Not so! If there is another soln. w. smaller Least (no matter how bad it is) this smaller Least soln. will be found first. → But other larger Lc soln. will eventually be found, w. a total Lc < t. threshold.

.06 impt. The only way to get a lower total bc. soln. would be to backtrack to a previous point in the TS. at which the total bc. is lower & then use ~~incremental~~ bc for Lsrch.

Some other ways: ① change order of problems in latter part of TS.; ② do ① but remember order toward problem order given by "teacher"; ③ ①, but somehow use some or all of the solns. obtained in the small cjs. soln. to reduce cc of finding solns.

.20 Another approach: Note that the small CJS solns are adequate for the time being. Continue to use them until a prob. arises in which they are not adequate & the less cc soln. is needed. .... This might require much greater cc, hwr.

Also, this may not work: we would like the teacher to be able to get TM to acquire a certain set of abss. so it could be in a position to learn some more complex combinations of these abss.

.20 ff either doesn't do this right, or does it badly.

[illegible] .06 seems to be an impt. point.

2.13.82 TS

.01 On t. small CJS v.s. less el. soln. :

We can order t. possl. solns. obtainable on t. basis of cc, pc & Lc.

T.M. can optimize "expected yield" — in either a fixed cc envt. or an envt. w. a "softer" cc edge. The planning for a search should use probablistic info.

Another way would be like t. chess pgm: To look for all solns. w. Lc. of < 1 units, Then after these are found, look for solns. of Lc. < 2; Then Lc < 4, ... Lc < 2^n.

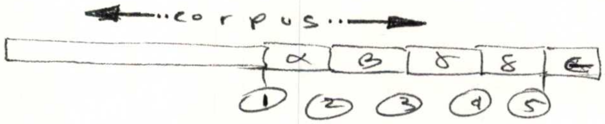
Trouble is: Hy Lc solns. can be of 2 forms: (1) as augmentations of previous soln. (2) as augmentations of far backtrackd solns.

.16

1) Alternate psychologys: — T. lack of team would suggest non-feasibility of present psychology(s).
2) little/social Mayaesa: Good for study of language in a reasonable sized lab.
3) add a maybe .5 volts to comp. vid. signal to get rid of top line staggering.

[2.23.82] I haven't been able to get much out of .01 yet, but 7.06 seems v.g. & ≈ true. Since I am interested in a low bc for t. entire corpus, & this incremental coding, using minz. of conditional entropy, is certainly incapable of giving better bc's for t. entire corpus, I must do something like backtrack:

.23



So I can do incremental coding of ε, from (5), & get some kind of soln.

Next try incremental coding from t. code usd up to (4), of δε. Try to get a better bc for t. entire corpus.

Next try inc coding from t. code usd up to (3), of γδε try to get a better bc for t. entire corpus...

.30 Next back to (2), etc...

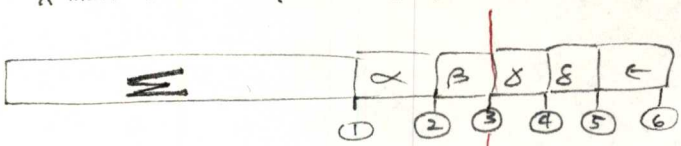
I vaguely remember my thinking that I had made some headway in this particular TS prob: I guess 7.06 was it. — However, not bad!

T. genl. idea is, I think, that I must have a less el. soln. : T. easiest way, t. way that uses most directly, t. regys found in t. past, is .23 — .30.

.35 There is some Q about whether to move back from (5) to (4) to (3) ... i.e. in such small steps. It may be o.k. if t. ≈ pc jump betw. these points is > a factor of 2. (→ see 10.20 for a dozen. of ≈ this)

A more exact analysis of 8.23 — .30.

.02



Say we have a code for Σ, C_Σ [redacted] so $M(C_\Sigma) = \Sigma$.

Then we try to code α, given C_Σ. So [redacted] $M(C_\alpha, C_\Sigma) = \alpha$.
↑ a new, 2 sym. machine.

I think that $H(\Sigma\alpha) = H(\Sigma) + H(\alpha/\Sigma)$ t. defns. of H are those of 81 TS 131.01ff.
$H(\alpha/\Sigma)$ = H of α, given Σ.

or $H(\Sigma, \alpha) = H(\Sigma) + H(\alpha/\Sigma)$. — This hvr., is true only if we have [redacted] $cc = \infty$ to compute t. H's. for finite CB, T, we may have

$H_+(\Sigma, \alpha) \geq H_+(\Sigma) + H_+(\alpha/\Sigma)$ — Tho I'm not sure ... t. inequality may be t. other way!

---

simplifying t. idea of .02 a bit by cutting it off at ③!

Say we have some codes for $\Sigma\alpha$; These are $C_{\Sigma\alpha}$.

Then, if we get codes for β [redacted] given $\Sigma_{\Sigma\alpha}$, these are $C_{\beta, \Sigma\alpha}$,

Th. combination of these 2 codes (sets) is a code (set) for $\Sigma\alpha\beta$.

Say $C_{\Sigma\alpha}$ does not include all short codes for $\Sigma\alpha$. Then [redacted] even if we got all codes for β given $C_{\Sigma\alpha}$, we wouldn't end up w. t. best coding of $\Sigma\alpha\beta$ — i.e. "all" t. codes for $\Sigma\alpha\beta$.

So, if we want to improve on t. code for $\Sigma\alpha\beta$ [redacted] from

{ [redacted] we can't retain just t. old codes for $\Sigma\alpha$. One way to get more, would be to go back to t. codes for Σ ; try to get codes for $\alpha\beta$ [redacted], knowing t. codes for Σ.

[redacted] t. upper bound for pc. of t. corpus $\Sigma\alpha\beta$; is hyar if we [redacted] code Σ, [redacted] w. a finite cc, then code $\alpha\beta$ knowing that code for Σ. > : As opposed to < starting w. t. same code for Σ, coding α [redacted] knowing that Σ code, then coding β knowing that [redacted] $\Sigma\alpha$ code. > — unless / in t. 2nd case we consider all possl. codes for α knowing this / finite cc code for Σ.

— T. moral being that a less El. soln is potentially better! ☺

~~Usually breaking a corpus into 2 parts that are more = in size is less El. than~~

The elism of Σ coding Σ, α, β as opposed to Σ, $\alpha\beta$. $\Sigma\leq\{\}$

—

Oh. now: what about a search strategy for 9.03? maybe that's what 8.01–.16 was about!

Consider e.g. figure of 9.02: Consider the follg. searches:

.10 5) code for $\in$, knowing the codes for $\Sigma\alpha\beta\delta$

4) $\delta\in$ " " " $\Sigma\alpha\beta\gamma$

3) $\gamma\delta\in$ " " : $\Sigma\alpha\beta$

2) $\beta\gamma\delta\in$ " : " $\Sigma\alpha$

1) $\alpha\beta\gamma\delta\in$ " " " Σ

I would expect 5) to be shortest, 4) ~~most~~ longer, 3) longer, etc.

"longer" in the sense of LC. Now, just how much LC to expend on each of these searches?

Clearly, the low ~~num~~ order searches are less elis are potentially better.

.18 We could just first use as much LC as nec, to finish 5), then continue w. remaining LC on 4), etc., using as much LC as one had to get as far out as possl. toward 1). The Q is, then, should we include all of the steps? ~~Ask~~ Skiping some will save LC, if we are successful. If each search takes ~ twice the LC of the previous one, probly there's no point in skipping (e.g. 8.035)

.20

In the search of .18 – .20, we assume that the first soln/found for each problem is accepted & then one goes on to the next. A more advanced technique would be one that would search until a soln was found ~~for~~ that obtained a ⟨code for the entire corpus⟩ that was better than the ⟨~~code~~ entire corpus code⟩ implied by the earlier best previous soln.

Another search method would go thru .18 – .20, finding 1 code for each search, then go thru .18 – .20 finding 2 codes, the 3 solns, etc.

I would think that if a single soln has been found for 1), that usually one would next look for a second soln. to 1), & a third, etc., if there were any cc left.

2·26·82 TS: N.B. 25.01 ff seems to be a much better analysis of the pc problem.

T. order of search in 10.18–.20 should be investigated in some simple cases —

e.g. the Eval problem w. t. 2 kinds of subs. I do have estimates of —

t. costs of the various subs of ①, ② ··· ⑤ [81TS 117.19 – 119.14; 121.10–.23 126.30, 127.17] } 81TS 147.21–.28 Note

T. results of 81TS → 12.01

300 baud
200 T
40 sec!
× 125
400 k by
3.2 M bi
3.200 k
3pp
10k sec
/50 min
= 2⅔ hrs
or so Real

400 k
15
=
= 30 sec.
×5 =
/30 T.

.10

cc: ① using ~~stop~~ loop (≡ stop rule)

n_i = no. of subsns. needed in i^{th} example

m = total no. of examples.

A ≡ cc of 1 substitution.

B ≡ cc of observing whether loop ends or not. [usually A >> B]

N ≡ $\max(n_i)$ = largest n_i overall i.

cc of loop =

$$\text{cc using loop} = (A+B)\sum_{i=1}^{m} n_i$$

② cc not using loop = $A \cdot m \cdot N$.

$$\frac{\text{cc not using loop}}{\text{cc using loop}} = \left(\frac{A}{A+B}\right)\frac{N}{n_i}$$

1 Let L = 0
2 Let N = 0
5 Print "sp"; L, N
3 Let L = L + 1
4 N = N + 1/L
6 Go to 3

pc's: ① using ~~stop~~ loop (≡ stop rule)

c ≡ pc of 1 xfmn. Eval_1; which is 1 substitution.

ℓ ≡ pc. of observing ~~using the rule~~ whether the loop should end, & ending it

pc using the loop is ~~probably~~ $\approx N$

loop = $c \cdot \ell$

② not using the loop:

$$pc \approx c\,\frac{1}{(N+1+\epsilon)N} \quad : \epsilon \ll 1 \text{ so } \approx \frac{c}{N^2}$$

no loop

[81TS 127.01

~~Feynman~~ For more exact formulation see 81TS 118.17–.32

I think it is $pc = \frac{c}{N(N+1)}\,K^c\,e^{\gamma c}$

$= \frac{c}{N(N+1)}\,(1.78N)^c$

[$\gamma = .5772157$ ≡ Euler's const.
$e^{\gamma} \approx 1.781$
$= 1.7810725\ldots$]

$$\frac{pc \text{ not using loop}}{pc \text{ using loop}} \approx \frac{1}{\ell \cdot N^2}$$

$$\frac{Lc \text{ not using loop}}{Lc \text{ using loop}} \approx \frac{A}{A+B}\,\frac{N^3 \cdot \ell}{n_i} \qquad Lc \equiv \frac{cc}{pc}:$$

$$\approx \left(\frac{A}{A+B}\right)^{\approx 1}\frac{N}{n_i}\cdot\frac{N^2\cdot\ell}{\text{maximize}}$$

A strictly dominated by m or N

$1 + \frac{1}{2} + \frac{1}{3}$
1 1.5 1.833
0 .7 1.098

$\gamma + \frac{1}{n} - \ln\left(\frac{n}{n-1}\right)$
$= \frac{1}{n} + \ln(1-\gamma)$

2·26·82 TS

.01 : 11.10 : From 11.10 : consider a picture of 9.02 & 4. defns of 10.10!

loop pc code:	Non loop cp
5) [illegible]	
≃ [illegible] $\left(\frac{c}{4.5}\right)$ c.l	$\frac{c}{5.5}$
4) $\left(\frac{c}{3.4}\right)$ (c.l)	$\frac{c}{5.5}$
3) $\frac{c}{3.3}$ · c.l	$\frac{c}{5.6}$
2) $\frac{c}{1.2}$ c.l	$\frac{c}{5.6}$
1) c.l	$\frac{c}{5.5}$

It would seem obvious to a thinking observer that as we go from 5) toward 1), that 4. non-loop pc stays constant & that the loop pc ↑ .... watching what happens in detail should make this even more obvious ... but I don't know if that sort of observation & reaction is necessary for this T.S.

The dimension Yo-Yo in War and Peace.

The attack Yo Yo

The defensive Yo Yo

Avocational Yo Yo ...

Professional Yo Yo. Supersonic Yo Yo.

i.e. the loop operator makes 4. code whose pc is $\frac{c}{n(n+1)}$ unnecy ... for any n

[redacted] is this extra "c" necy [redacted] in all these codes) — probly not: see .30

Several notable things about this "soln." to 4. problem:

① 4. presence of cc in 4. pc except for code 1). I'm not sure but the c's are necy

② 5) is 4. lowest pc code. T. pc ↑ as we go from 5) to 1).

So for code 5) (which is 4. first code we do) the loop code is least encouraging. — Things slowly begin to look better as we move toward code 1).

[redacted] Even tho 4. loopy coding method seems very poor & wasteful of cc, if it is workable (even at excessive cc), it will be a (perhaps) good start for a serious TS.

More details on 4. codings of .01 :

.30 When 5) is coded : after coding ΣαβγδS (see 10.10 & 9.02), 4. symbol for Eval$_1$ is rather likely, ... it has very hy pc. — so instead of p.c. $\frac{c}{4.5}$ · c.l for 4. code pc., it's more like $\frac{c}{5.6}$ · l.

Also, however, since 4. symbol Eval$_1$ has pattern such hy pc, 4. symbol for 4. loop recogn. must have its pc ↓ — so 4. final pc should be < $\frac{c}{5.6}$ · l.

I will have to do a more detailed analysis of 4. coding (& perhaps search), to see whether 4. system would really work & get estimates of 4. cc's of solns.

2.28.82 : TS:

On "Minimal inductive capacity": Perhaps there is a minimal inductive system that can be mapped into any (arbitrary) inductive system, using a suitable TS. This would correspond to a "Universal" inductive system, like a UMC — or, more like a Kusp, since cc would have to be considered.

In particular, I'm thinking of my present system, (learning / working) re. "Eval" problem.

My idea is perhaps that this TM is an operator! Its input is a string, its output is a code for that string wrt some UMC.

- 10 Say TM ≡ machine M_0 ; $M_0(x) = p$ $\quad M_U(p) = x$

we want to modify M_0 in various ways to fit its efficacy.

$M_0(\cdot) = M_U(\alpha, \cdot)$. α is the descr. of M_0 wrt M_U.

This looks like a TM_1, TM_2 problem — but in the prob. described, the TM_2 is a optimum. prob., & the TM_1 is a min. coding problem —

One (Not so great) way to deal w. this:

Say $\alpha_1, \alpha_2, \alpha_3 \ldots$ are a bunch of values that lead to not bad Scores for M_0. We then want to extrapolate this set. for trials, & this is the sort of thing M_0 can do. It would do this using the (maybe) best α_i value thus far.

I do remember an old trick there! Say $G(\alpha_i) = G_i$ (corr)

so we have $G_1 = 3$, $G_2 = 7$, $G_3 = 6$; & $9 > \text{all } G_i$ thus far:

.29 → so we want TM to find a α → it is most likely to have a $G = 9$.

Good! More exactly (& may be easier): we want a α w. an expected G of as large as poss!

A direct induction problem.

In trying to solve .29 for $G=9$, we might as well try it for $G = G_{\text{general}}$ — which means we want to find a useful (low cc) relation between G & α that is reliable — i.e. so we can predict a G from a α & conversely, invert the operator w. acceptable cc to obtain a (bunch of, usually) α from a given G.

Not bad!

.37 → The usual way to do this: look at the bunch of good α_i's, try to find a min complexity model/relating/pair G to their structures, α, so that $G(\alpha)$ is not too difficult to invert.

It would seem that .37 could be done by the machine, M_0, of .10 →

imc.
250 counts in 4 ms.

Prof Cheeseman will know Stat dept.

47×10^{-12}
$\times 10^{6}$
47 μsec

Tues:
E.40 MIT
4 PM.

$\pi = 4$ ms
my π 4 yrs!

$\pi = 47$ μs

low w.

actually, they needed not be all or even mostly good α_i's

3.1.82 : TS

Th. [redacted] technique of 1.37 is o.k., but it will be useful only if TM has had enuf experience w. probs of that sort.... otherwise its unlikely that he would make an TM-ish "breakthrough".

If I am to bring TM to t. pt. where it can usefully work on t. self improvement problem, I will have to have taught it various techniques related to self-improvement. It might be easier to just "wire in" those techniques, so TM gets to the self-improvement threshold as soon as possl.

Hwr., t. advantage of having TM learn these techniques, is that it's more likely that TM will then have v.g. access to them & will know how to combine them & their parts to form other new, useful to bss. Otherwise we have t. prob. of "Representation of knowledge".

3.8.82

.18 1) In t. β^{12} v.s. loop problem: More generally, the β^{12} soln. is t. result of using t. best Lcost soln. in view of t. β'' soln. We want (at least) t. best Lcost soln. in view of β'', β'^{0}, β^{9} ... β^{0}. — also in view of seven previous soln's. T. method of effectively search ce [redacted] betw. these various efforts is unclear.

2) We want a system of working on TM (& TS's) in which any debugs or other criticisms of t. TS. can be readily translated into a correction (or modifn.) of t. TS.

3.10.82

.30: In line w. t. idea of "Min inductive copy" of 13.01: That I might devise a not-so-good inductive system that would be able to learn certain TS's: Then I would gradually improve this system to work more & more difficult probs.

→ One way: assign = appnd to [redacted] $\beta', ;, ;, \beta^{12}$. Then in searching, the Lcost pnd of t. final (desired) soln. is mult. by just 12. One searches over these strings in Lcost order. Since t. integer "12" can be >> 12, & if there are many examples, this can end up being a very hy Lcost. It would be well to find a better method. → 18.11 spec

3.10.82 TS

.01 → T. thing I don't like: That I had this idea that TM could learn a concept in a tng. seq. of cjs = α, by a search whose Lcost (≅ cc) was ~ α. The β^{12} v.s. loop business seems to contradict this!

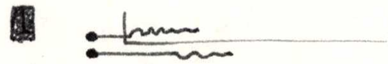
8 6
12 9

Th. idea of .01 is true, hvr. in one sense: That if t. proper conditional probys (or aquist) are gn., & t. proper incremental corpus or problem set(s) (are) given, then t. cc for Lsrch will be t. expected α.

.09

.10 [3.11.82] A possl. approach:

16 × 12
9



Consider fig of 9.02: first, in view of Σ, we code α. The lowest Lcost code will be β. But we continue, since we have much more cc available, & we find many alternative codes. We find $\underline{\beta}^2$, $\underline{\beta}^3$ ... & eventually, if we have enuf cc, we find $\underline{\beta}^{12}$ & t. loop code. (The loop code has, say more pc than $\underline{\beta}^{12}$ — but both are << t. pc of β — which does fit.)

20 40 58

"Def" [Note: we have underlined β in $\underline{\beta}$, $\underline{\beta}^2$, $\underline{\beta}^3$ ... $\underline{\beta}^{12}$ to distinguish from t. β of 9.02]

[3.12.82] This idea of continuing Lsrch for hyr Lcost codes is not unreasonable: it is usd to get better probys, but also was considerd as a good way to avoid (usually) backtracking. It's this latter that I'm involvd in now. → 81TS: 92.30, 94.01, 96.38, 127.23

— Perhaps see re-(?) Back tracking

21 lines
42 chars

200 sc
20 pf
= 40 mc
100 e
30
= 9 W

One impt. Q. about retaining many non-Minimal solns: Storage costs. One way to face this problem: Cost of extra storage V.s. cost of not using t. storage & getting into backtrack trouble later. Hvr, it may be that if there are many solns., this set of solns can be collectively coded compactly.

M?
Q?
W

345P
Stew Boiling
Var 50

.32

353P
Var 60

In t. approach of .10: An alternate approach that seems close to it: In coding t. fig of 9.02: first code Σ. Then code α in view of t. code(s) of Σ

" " αβ " " " " " " Σ

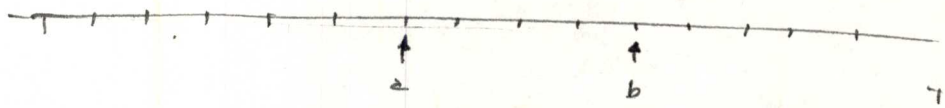
αβγ " " " " " " Σ etc.

21 ×
42 =
882

32×16 =
512

If we continue to code αβ.... ε etc, eventually, t. loop code will be of lower Lc (& hyr pc) than $\underline{\beta}^{12}$ or say.

.01 So t. goal problem seems to be: we are given this try seq. of probs. To decide how much LC to use at each point, & what to use as conditional probty at each pt.



← t. line represents t. T.S.

↑ t. marked cont. line are t. ends of probs.

E.G. At point b, we try to code t. section from a to b, using t. known code(s) of up-to-a., & we use LC limit L_0. **T. Q is:** at pt. "b", how far back should "a" be & what value of L_0 should be used?

Well, I guess that we push a back as far as possl., .18 w. t. constraint that a soln. be obtainable w. available cc. → 18.17

Another approach: Consider "Levin's conjecture" that Lsrch is ≃ t. best way (within a small factor (like 2 or 10)) to search for solns. to problems of all kinds. So t. Q is then, what are t. typ. segos. that humans use ≃ Lsrch on? We do know, to some extent, t. solns. that humans obtain, so perhaps we can figure out just how t. Lsrch was done: perhaps how L_0 & (b−a) ← (& how far back to go) were obtained.... Tho, of course, this last may not be a meaningful Q.

.25 → Note that the points at which candidate pts. at which to place "a", are limited in no. At most pts. in t. corpus, t. code "does not change much" — whatever that means! — I guess it means that if p_1 & p_2 are 2 pts at which t. "code hasn't changed much" in view of t. code at p_1 or at p_2 would be .35 pretty much t. same. → 17.25

In t. B^{12} vis. loop soln: Tl difficulty is not that B^{12} is inadequate; B^{12} works fine / for t. present problem. T. only trouble is that I want TM to learn t. loop soln. because it will be (or its parts) will be useful in learning more complex things. In. o. t. concepts of this loop soln., these more complex things would be of excessively hy LC. to learn.

8024 ~ 1K

say 512 × 256 dots = 128K dots.

At 4 bits/dot: 16 shades of Gray.

at 9 bits/dot: 3 bits for each of 3 colors: so 2^9 = 512 colors possbl.

64K dots:

.7 × 512 ×	180K
.7 × 256	360

A radically difrnt. approach ! : Since β'^{12} works O.K. for t. probs. up to now, it will probably not be too bad for t. more advanced probs. in which the loop soln. would be normally expected to be used. — so, could I temporarily accept β'^{12} & go on?

There are at least 2 approaches to "t. BUG" ! (1) Try to find good ways to divide up t. corpus for less e/. soln(s). (2) Observe t. β, β^2 ... β'^{12} seq. of solns. Observe them in action & notice that t. loop soln. would be better. (This last looks like a TM₂ approach).

Another approach: That TM normally tries to code for <t. entire corpus as a unit>. Since this is usually impossl., it backs down as little as it can to less e/. approaches. This is ≈ t. idea of 16.01–.18 Also not too dis'tant is t. idea of 15.32–.40 ... which tries to keep things as non-e/. as possl.

Looking at the β'^{12} soln: it's clear that it's an increment to t. soln. of Σ (see 9.02). .... So perhaps it is natural to try for solns. coded as $\langle \alpha\ \beta\ \gamma\ \delta\ \epsilon$ in view of t. code(s) of $\Sigma \rangle$.

32 × 24 × 8
6 × 32 × 32
= 16 K by tos

.25 Reconsider 15.32–.40: We use Σ as our "in view of"
.26 & continued t. corpus since, until t. cc of soln. is too large — i.e. we don't find a soln. in t. available cc. We then move t. adjcnt t. "In view of" sub-corpus, forward. Well, how far to move it forward? This seems a little simpler than t. idea of (16.01–.18, 16.25–.35.) Perhaps use t. idea of 16.25–.35 to decide on a trial increment of Σ.

In t. present case (= t. "Eval" problem), say Σ included stuff before "Eval" type probs. were gen. Then α was t. first "Eval" problem. Well, say we used .25–.26 to obtain d. loop soln, as well as t. β, β^2 ... β'^{12} solns. Say we continue w. more Eval probs. T. loop soln. is adequate, but we also have β'^{13}, β'^{14}, etc., which are of lower pc. in hyper Lc. Then we give a new type of problem. We use "In view of t. soln. of Σ" & find no soln. in available cc.

.38 One possl. "soln.": To automatically move t. "in view of" adjc to as far as we had coded before "t. cc of soln. got too large". → 18.04 spec

3.13.82 TS

.01 T. forgg. stuff sort of assumes t. corpus is a simple sequential object to be coded. In fact, it is not. T. problems are ordered but only for heuristic reasons.

.03 T. prob. for TM is to devise a min. complexity operator that relates t. I/O pairs gen. in t. TS.

.04 : 17.40! Re: 17.38! while I'm not overjoyed w. it, it looks like t. best thing this far & may, w. possl. refinement, more or less work.

— we code as far as we can go w. t. gen. cc., then we use that point's code as t. conditional prob for each t. subsequent code trials.

.11 : 14.40 14.18 is perhaps correct! At any point in our coding, we want a code "in view of" all of t. previous partial codes. As we noted there, t. allocation of cc to these various searches is a big problem

.17 : 16.18 There is something like a monotonicity condition in 16.01 — .18 : w. a fixed "b" point, & a very long "previous corpus", as one moves "a" back, the cc of soln. almost always does not ↓. This makes it possl. to find an "a" point that is ≃ as far back as can be, given t. cc available. This "a" .25 point is obtained by log search. Say "a" can be any point from α to β

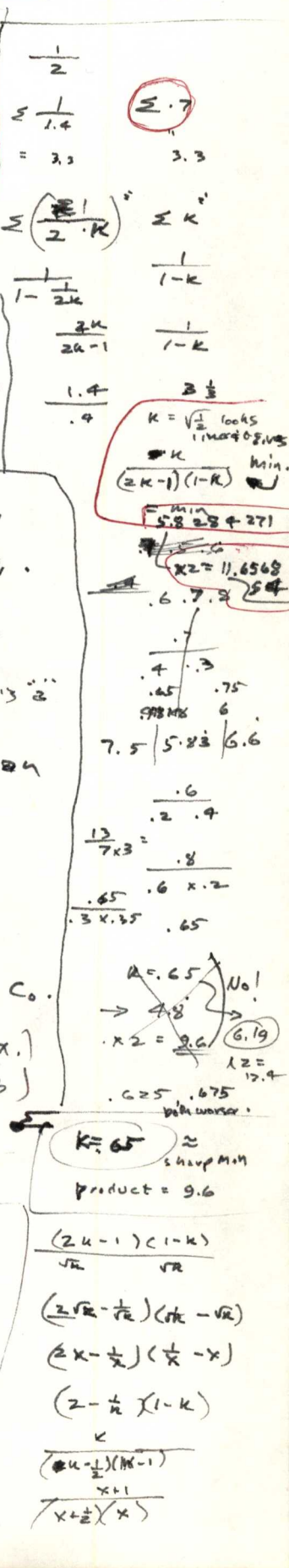
α ——— n₁ ——— β

We first try a mid point using a certain cc threshold = C_0.
If a code is found, we try n_2 = (½ way betw n_1 & α.)
" " " is not found " " n_2 = (" " " " " β)

.30 etc.

This monotonicity cond. makes it possl. to get estimates of L costs as a function of position of "a". We can also get L cost as a function of "a" position for t. first 2 or 3 or n valid codings of t. corpus

.36 3.14.82 Also, in 17.38, 18.04 - 08! Say TM is gn. a big Block of problems (ordered). He could then try (t. whole set as a unit. If unsuccessful (in cc < C_0) try ½ of probs, ... using a log search like .17 — .30 to determine how far it can code & stay under t. cc threshold. → 19.01 spec



3·14·82

.01: 18.40 Actually, 18.36 may be unnecy; it may be as effective to simply start w. ε. first problem & continue coding until one's cc runs out. Here I assume that we ~~code~~ code a gn. prob. until 1 soln (or 2 or 3 or some fixed no. of solns) is found — then go onto ε. next problem.

There are 2 similar, tho difrnt. approaches: ① that of 17.30 ff (gn. a Σ, code as far out from that pt. w.r.t. ~~doing~~ Σ's codes, as one can w. ε. available cc). .09 ② 16.01 Given a problem b, push a as far back as one can, given ε. available cc. (assuming 1, 2, 3 or some fixed no, n of solns to each prob.). So in ① we work forwards, in ② we work backwards.

The details of ① are clearer than those of ②. In ② I expect to use ε. log search of 18.25, somehow.

.19 A difficulty w. ①: The jumps would seem to not be very "natural chunks". As a result, suppose a gn. problem needs a bss. that would normally be derived in ε. previous 7 problems — so this problem would be solvable, if there was a "chunk" that included it & ε. previous 7 problems. Hvr., in ①, the "chunks" may well have boundaries in ε. middle of those 7's & so ε. proper a bss. would'nt be found.

In ②, hvr, in solving a problem, we always include in ε. "chunk" for that problem, ε. previous n problems (n being as large as is allowed by ε. cc limit) — so usually, ε. 7 would be included.

A modifn. of ① that makes it very close (if not identical) to ②!

.30

.31

Σ : α : β : γ : δ : ε :
1 2 3 4 5 6

Using ε. code up to 1 as conditional probs, we code as far as we can into ε. future w. gn. cc (standard ①-type approach).

Then, given ε. code for $\Sigma\alpha$ we: then gn code for $\Sigma\alpha\beta$, etc.

This Modifn. of ① would seem to overcome ε. difficulty of .19, but at ε. apparent expense of large cc to do ε. entire corpus.

.37 i.e. if c_0 is ε. cc limit being used, then it takes $\sim c_0$ of cc to solve each problem in ε. corpus. → 20.25 spec.

3.14.82 05

On coding w.r.t. a gen. sub-corpus (= say $\mathcal{E}$). Say $\mathcal{E}$ has several known codes. Is there a way to do a joint/search for contin. codes?

Yes: say ~~one of the codes~~ there are 2 known codes for $\mathcal{E}$. One code X_0 has pc p_0, & one X_1 has pc, & $p_0 > p_1$ (say). Using X_0 to start, one adds on bits in pc order until a usable code is found, an impossl. code is found or Lc threshold is exceeded.

However in adding bits to X_0, one eventually comes to a pt. where & total pc is $< p_1$; At this point one begins adding bits to X_1.

It may be possl. to devise a tree/search based on > 1 initial codes, just as one does a simple tree search based on a single initial code. The details must be worked out.

Well, actually, its rather trivial & not very interesting!

Say one is at the stage of Lc threshold = L_0. One tries all continns of X_0 w. Lc $\leq L_0$; then one tries all continns of X_1, w. Lc $\leq L_0$. Then all " " X_1 " " " $2L_0$; " " " " " " X_1 " " $2L_0$

" " " " " " $2^2 L_0$ . . . . . X_1 " " $2^2 L_0$, & etc.

Similar technique w. < 2 initial codes

.25 | 19.40 More detail on 19.30 – .40: consider the β, β^2, $\cdots$ β^{12}; loop codes.

Def A temporary defn. If Σ is the "Basis" of a code for α then that means this code for α is w.r.t. all the known codes for $\mathcal{E}$.

Consider, as in fig. of 19.31, using the method of coding of 19.30!

using basis $\mathcal{E}$, we code α & get $\underline{\beta}$; with still w. Basis $\mathcal{E}$, we get code $\underline{\beta}$ or $\underline{\beta^2}$ for $\alpha\beta$; then maybe $\cdots \underline{\beta^3}$ for $\alpha\beta\gamma\delta\cdots$ etc. for a long enuf contin. we finally get the loop code as well.

Next, using $\Sigma\alpha$ as basis, our initial code is $C_{\Sigma\alpha}\underline{\beta}$ ($C_{\Sigma\alpha}$ is the set of known codes for $\mathcal{E}$). Then the code for β w. basis $\Sigma\alpha$ may be $\underline{\beta^2}$. The codes are the same as the first time thru, but they (except for the loop code) have lower pc's. The total pc's of these codes, which are the pc of $C_{\Sigma\alpha}\beta$ times $\sum$ (the pc of β)^k are about the same, or a little smaller

.01 than the first time thru — which gave pc's of (1. pc of C_E) times (1. pc of β)$^{k+1}$. We end up w. $\approx$ same pc for the loop soln. ≈ 70834

But anyway, at each point, (say 5 of 19.31), we end up w. a code for δ w. basis $\Sigma\alpha\beta\gamma$ plus
" " " $\gamma\delta$ " " $\Sigma\alpha\beta$ "
" " " $\beta\gamma\delta$ " " $\Sigma\alpha$ [redacted] "
" " " $\alpha\beta\gamma\delta$ " " Σ.

Which looks about the same as (2) of 16.01 (& 19.09).

This is the same as 16.01 – .18 but placing "a" at all possl. intermediate pts., using some fixed value of L_0.

While in the most genl. case, the remark of 19.37 is true; i.e. it takes a very large amt. of cc per problem. However, in the case of the "Equal" problem, in fact,

Very little additional cc. is needed as one passes from Σ to $\Sigma\alpha$ to $\Sigma\alpha\beta$ .....

At Pt. 2 of 19.31 / [redacted] the solns. are β, β^2 ... β^{12}, loopsoln. (& several others.... since there is only 1 example there may be many ah solns. that will be discarded as more examples are given.).

At pt. 3, the solns. are β^2 .... β^{12}, loopsoln., perhaps β^{13} & fewer ah. solns. the cc needed to find β^{13} after β^{12} has been found, is small. β^{13} loopsoln

At pt. 4 / ∴ etc. One has the alternatives of spending the same amt. of cc at each problem, i.e. by coding fairly far into the future, or using lots of cc at pt. 1 & less cc at successive pts. .... if one could recognize that the situations were pretty much the same. One basis for such recognition would be the fact that most of the β solns. are retained, & the loop soln. is always retained. However, the β^i for lowest i may be discarded since (which is, in a sense, [redacted] a ah. soln.) & other ah. solns. are discarded, so it's not so clear how one could do this recognition.

.29 → 22.01

"Next day" [SN] In human TS's: using very small cjs. can lead to the non-aquisition of certain more complex concepts. So a teacher may occasionally give a prob. that has a very larger cjs. This will be beyond the [redacted] available cc of certain students, but for the ones that can make it, there will be rewards of impt. new concepts.

3·14·82 TS

.01 : 21.29 : Another (to some extent) approach ! (similar to 16.01 [illegible]) : Normally, TM moves from problem to problem, with "a" at about the previous problem or a few problems back (depends on how much cc is available). Occ., every once in a while, TM has time for "Meditation", at which time he moves "a" as far back as the available cc will allow. This "moving back" motion, can be done by jumps (as in the logarithmic search of 18.25)

.11 .12 In moving "a" backwards, it might be possible to move it ≠ ≈, problem [illegible] at a time, since the cc is small for small values of "b−a" (This may be true int. moreover (i.e. true than just the problem of β^{12} v.s. the loop soln.)

.14 Occ., the pc is, I think, exponential in "b−a", so if k is the pc of the loop soln. ([illegible] say (a the pc of β^{12} is ≈ k also), then the pc of the β^n solns will be ≈ $k^{\frac{n}{12}}$, and the total cc of

If each trial [illegible] is of cc. = c_0 then the total cc will be

$$\sum \frac{cc}{pc} = c_0 \sum_{n=1}^{12} k^{-\frac{n}{12}}$$

If $k^{\frac{1}{12}} = \frac{1}{2}$, then this ≈ $c_0 \cdot 2^{13}$ which is only 2 times > $c_0 2^{12}$ which is the least of the loop soln. ($k^{\frac{12}{12}} = k = 2^{12}$)

So, in general, maybe just stepping backwards 1 problem at a time (or 2 probs at a time) may not be bad. The number of problems [illegible] per step is ⇒ the pc of soln. is by a factor of > 2. So if the probs [illegible] don't ↑ in difficulty too much we will take many at a time.

.30 **In fact, we can** servo the no. of probs/jump. Take the pc of 2 solns of successive jumps. From their ratios one can find the no. of probs to jump to get this ratio ≈ 2. One makes the jump, gets the new pc. & recomputes the size of the next jump etc. — discussed in 29.17

.35

A human teacher w. human pupil will make sure the student has acquired a certain concept before going on to more complex probs. a/o probs. requiring the use of that concept. In the case of TM. I don't know just how to do this. ∴ It could be done by

7·27·81 to 3·19·82 7 ± months on this one problem (more like 2½ to 3 mo) 1 mo to (7·27·81 / 9·6·81) 1 mo 10 d (1·30·82 to 3·14·82) 1½ mo? so 3 mo

giving probs. ~~require~~ That seem (to t. teacher) to require that concept, & seeing how TM (or t. human student) does w. them. In t. case of TM., if t. desired concept was not acquired, T. teacher could tell TM to spend more cc on a particular set of (past) problems, or, t. teacher could examine t. T.S. to see why TM didn't acquire t. concept from it.

T. (most attractive thus far) approach of 22.01 seems to be a kind of backtracking. ~~[illegible]~~ T. system seems like it ought to work. A perhaps major Q is: Could it be made significantly more efficient? Well, in t. case of $\beta \dots \beta^{12}$ & t. loop soln., T. analysis of 22.12-.35 22.42-.35 suggests that only a loss by a factor of 2 over t. cost of t. loop soln. would be involved: & using t. servoing method of 22.30-.35, perhaps an even smaller factor! A Q. is, of course, as to whether t. approach of 22.01-.11 (likely) will work in t. more genl. case / & whether t. quant. analysis of 22.12-.35 is more generally applicable (less likely).

For t. time being t. Q is: is 22.01 ff an adequate soln. to t. dilly of $15^{.01-.09}$ -? i.e. will TM. find all concepts of c_j's $\sim \alpha$, ~~[illegible]~~ if its able to use a cc of $\sim \alpha$? Well, this is true to t. extent that t. quant. analysis of 22.12-.30 is true — otherwise, t. cc needed could be much larger -

— Hvr, in a genl way, t. removal of 22.14 is probably usually true: i.e. pc is exponential in b-z & to t. extent this is true, then, using t. servoing of 22.35-.35, t. pc of ~~doing~~ t. ~~wh~~ backtracking will not be much > t. pc of t. concept ultimately sought for.

Actually, I think t. analysis of 22.12-.30 is way off for Thm $\{\beta^{12}$, loop^{12} soln.$\}$ > problem ..... well, seems to be off by a factor ~ ln 12 (maybe much worse than this, hmm.) — which is not too bad! The reasoning! Say t. corpus is

$\Sigma \quad \alpha_1, \alpha_2 \dots \dots \alpha_N$.

We will consider only pc's: cc's of solns do not vary much.

(Sec 11.01-.10 for a review of pc's & cc's of various concepts.)

SN A criticism of backtracking in t. present situation: 81 TS 127.27 ← Not easy to understand!!

81TS 127.01 & 130.01 have an analysis of the pr. of the sequence $\beta^{(n)}\Delta$ ... a being a "stop" symbol (to ensure a prefix set). The overlapping exposition is not very clear, however. I guess the idea is, one can assign an arbitrary value to the pr. of Δ (as a function of n) — subject to the constraint that Σ(pr. of all strings) ≤ 1.

T4 analysis assumes there are only 2 symbols in the alphabet: β & Δ.

.10 If the probability that the next symbol is β is $\frac{n+c}{n+1+c}$ (81TS 127.08)

then " " " " " " " Δ is 1 minus this no. Well:

~~If we remember the c factor, then~~ $1 - \frac{n+c}{n+c+1} = \frac{1}{n+c+1}$ ~~Then without the~~

~~the character is retained & c is small, the probability of Δ is $\frac{1}{n+1}$~~

According to 81TS 118.20 this should be $\frac{n+c}{n+1}$; so prob of Δ is $\frac{1-c}{n+1}$ (Note: c is small)

The initial probability of β is c; the probability of $\beta\Delta$ is then $c \cdot \frac{1-c}{2}$ (the t. symbol)

" " " " " " Δ is $1-c$ since the prob of β^2 is $c \cdot \frac{1+c}{2}$

$\frac{1-c}{2} = 1 - \frac{1+c}{1+1}$

The probability of $\beta^2\Delta$ is then $c \cdot \frac{1+c}{2} \cdot \frac{1-c}{3}$

.19

" " $\beta^n\Delta$ " " $c \cdot \frac{(1+c)}{2} \cdot \frac{(2+c)}{3} \cdots \frac{(n-1+c)}{n} \cdot \frac{(1-c)}{n+1}$

.20

prob of Δ = 1 − prob of β in that position

$= \frac{n+c}{n+1}$

Note that this prob is $\neq c$

.21 Q&A: Given the operator $\beta^5\Delta$; what is the condl. prob of it contain β^3? — or, more exactly, what's the prob. of the operator $\beta^{5+3}\Delta$? Exactly what does this Q mean?

.23 The diffy arises because the strings of interest form a prefix set.

Well, consider the machine, M(α, p): Its first output is a string that's a member of a prefix set. It is the "past m bits" part of the input. p is the pgm. & it, too is part of a prefix set, & the output forms a prefix set.

The defn. of the condl prob would have to be in the spirit of 81TM 132.27ff

My present impression: That the answers to .21 – .23 are that the prob of $\beta^{5+3}\Delta$ obtained in this way is the same as the prob of the $\beta^{5+3}\Delta$ obtained directly (e.g. viz .20). That the prob of the "contains β^3" is simply $\frac{\text{prob of } \beta^{5+3}\Delta}{\text{prob of } \beta^5\Delta}$.

Another important refinement of the result of .20 would take into consideration the ~~fact~~ ±ε probs of symbols other than β & Δ. Say there are m−1 symbols other than Δ. Initially they all have prob $\frac{1}{m}$. If the first symbol is Δ, that's the end. If it's not Δ, say it's β, then the distribution for the 2nd symbol changes.

.01 From I & II, the way we do this: write the symbols $\beta_1, \beta_2 \cdots \beta_{m-1}, \Delta$;

Then the prob of any symbol is the rel. freq. that it has had up to that pt., including

I think this is $\equiv$ or $\simeq$ "Laplace's Rule".

I'm almost sure this kind of rule is automatically normed.

.04 For $m = 2$ we should have 24.20! $c = \frac{1}{2}$: $\beta\Delta$;

prob of initial β is $\frac{1}{2}$; of $\beta\Delta$ is $\frac{1}{2} \cdot \frac{1}{3}$

of $\beta^2\Delta$ is $\frac{1}{3 \cdot 2}$ $\Delta\beta\beta$; $\beta\beta\Delta$: $\frac{1}{3} \cdot \frac{2}{3} \cdot \frac{1}{4} = \frac{1}{12}$

24.19 gives $\frac{1}{2} \quad \frac{1}{2} \cdot \frac{1}{3} \quad \frac{1}{3} \cdot \frac{1}{4} = \frac{1}{2} \cdot \frac{1}{2 \cdot 2} \cdot \frac{1}{2 \cdot 3} = \frac{1}{16}$) which suggests that 24.19 is wrong.

$\beta^n\Delta$ should be from .04 $\Delta\beta$; $\beta \cdots \beta\Delta$ $\quad \frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$

$\frac{1}{2} \frac{2}{3} \cdots\cdots \frac{n}{n+1} \frac{1}{n+2} = \frac{1}{(n+1)(n+2)}$ | checks for $n \geq 2$.

t. prob. $\frac{n+c}{n+1+c}$ [81TS.24] $\left(= \frac{2\frac{1}{2}}{3\frac{1}{2}} = \frac{1}{\quad} \right.$

gives $\frac{1}{2} \cdot \frac{1\frac{1}{2}}{2\frac{1}{2}} \cdot \frac{1}{3\frac{1}{2}} = \frac{1}{2} \frac{3}{5} \cdot \frac{2}{7} = \frac{3}{35}$.

---

for $m = m$; $\beta_1 \cdots \beta_{m-1} \Delta$;

.20 prob of $\beta^n\Delta$) $= \frac{1}{m} \frac{2}{m+1} \cdots \frac{n}{m+n-1} \cdot \frac{1}{m+n} = \frac{n!\,(m-1)!}{(m+n)!} = \frac{(m-1)!}{(n+1)(n+2)\cdots(n+m)}$ $\quad \approx \frac{k}{n^m}$

for $m = 2$ prob $\beta^n\Delta = \frac{n! \cdot 1}{(n+2)!} = \frac{1}{(n+1)(n+2)}$ so, it checks.

See 83TS2.01 for mpf. corrected

for $m = 3$ $P(\beta^n\Delta) = \frac{n!\,2}{n+3!} = \frac{2}{(n+1)(n+2)(n+3)}$

I feel that t. model of .01 is better than that of 24.10 ($\equiv$ 81TS 127.08).

— Namely that t. 24.10 model was a vague remembrance of Laplace's rule, &

I think .01 is closer to it & is more reasonable: also it extends readily to > 1 t symbol.

I really don't remember exactly how I got the 24.10 model

< 81TS 118.17 ff is more detailed on how was obtained — but not totally clear, anyway!

Anyway! from .20, it looks like, for reasonable values of m (say m ≤ 10)

that t. prob of $\beta^n\Delta$ could be rather low [& drops rapidly w. n], & I think t. prob of

t. loop soln. isn't nearly as low, so it should beat t. β^n soln. quickly.

However t. problem of β^n v.s. loop is a more general problem that

I'd have to face eventually. Also, I still have to work out

a better quant. analysis ref. 22.12 -.30

Next day Re: t. quant. analysis of 22.12 -.30! My present impression is that by using .01 ff (& .20 in particular), t. cost of backtracking (even including all of t. problems), is very close to t. least of "t. loop soln." (within a factor of, say 1.1! & certainly < 2).

The reasoning! Say we have β^n & we backtrack 1. / p.c. = $\frac{n}{m+n-1}$ (penultimate factor of $\frac{1}{2} \cdot 20$)

conditional

backtracking 2: p.c. = $\frac{n-1}{m+n-2} \cdot \frac{n}{m+n-1}$.

The respective costs are $\frac{m+n-1}{n}$ $\left(\frac{m+n-1}{n} \frac{m+n-2}{n-1} \right)$ | $= \frac{m-1}{n} + 1$, $\left(\frac{m-1}{n-1} + 1 \right)\left(\frac{m-1}{n} + 1 \right)$

.01 The Lwst of [redacted] backtracking & probs is $\left(\frac{m-1}{n-2}+1\right)\left(\frac{m-1}{n-1}+1\right)\left(\frac{m-1}{n}+1\right)$ ← Not exactly, t. Backtracking costs) s. Σ of 3 terms; This beats t. largest.

$$\left[\quad a + a\cdot b + a\cdot b\cdot c + a\cdot b\cdot c\cdot d = (((d+1)\cdot c+1)b+1)a \quad\right]$$

The total Lc is $\sum_{n=1}^{N}\prod_{i=1}^{n}\left(\frac{m-1}{i}+1\right)$ T. Q is: How does this compare w. $\prod_{i=1}^{N}\left(\frac{m-1}{i}+1\right)$? - T. last term.

~~No~~ : Also ~~Not exactly~~ ~~Also~~ Actually, There are 3 params of interest.

$M \equiv m-1$, n & j

$$\sum_{k=0}^{j}\prod_{i=0}^{k}\left(\frac{M}{n-i}+1\right) \quad \Big| \quad = \quad \Pi$$

If $n \le M$ Then all factors are ≥ 2, & all terms are $\ge$ previous term by more than a factor of 2.

In general, all factors are $\le \frac{M}{n}+1$ & t. sum of all t. terms is

$$\ge \sum_{k=0}^{j}\left(\frac{M}{n}+1\right)^k = \frac{\left(\frac{M}{n}+1\right)^{k+1}-1}{\frac{M}{n}+1-1} \cdot \Big| \text{ of more interest, each term is } > \text{t. previous one by a factor} \ge \left(\frac{M}{n}+1\right).$$

consider t. series:

$$S = \sum_{i=1}^{n}(a_i)^i \quad ; \quad a_i \text{ is } > 1 \text{, } a \text{ is a } \uparrow \text{ funct of } i \text{.}$$

We want to know limits betw. S & $(a_n)^n$, t. largest term of S.

prob'ty $p^n\Delta = \frac{1}{m}\cdot\frac{2}{m+1}\cdots\frac{n}{m+n-1}\Big|\frac{1}{m+n}$

T. last k terms are $\frac{n-k+1}{n-1+m-k-1}\cdots\frac{n}{m+n-1}$

$$= \frac{(n-k-1)!}{(m+n-1)!}\cdot\frac{(m+n-1)!}{(n-1)!}\cdot\frac{n!}{(m+n-k-3)!}$$

see .01 for reasons why

Worst situation: small M, large n, small j — The small j may not be so bad, since there are only j terms, which limits t. amt. of wastage.

Apparently, large j is usually bad. — but small j can also be bad See 27·11-·18

So consider $M=2$; then $M \ge 3$.

Actually I could compute for various values of n; The various values of j would not impose additional computational burden.

for $\frac{M}{n} << 1$ & $n >> 1$ & j small,

.38 $\text{Sum} \approx \sum_{k=1}^{j}\left(1+\frac{M}{n}\right)^k = \frac{\left(1+\frac{M}{n}\right)^{j+1}-1}{\frac{M}{n}}$: T. largest term is $\left(1+\frac{M}{n}\right)^j \approx j$

Example False : lower limits should be $k=0$

T. ratio of sum to largest term is $\frac{n}{M} \cdot \left(1 + \frac{M}{n} - \frac{1}{(1+\frac{M}{n})^j}\right)$

This is an unexpected! function of j as $j \to \infty$ it $\to \frac{n}{M} + 1 \simeq \frac{n}{M}$ (

However, the approxn. could a.

of $j \ll n$ is important. so j can't $\to \infty$

for $j = \frac{n}{2}$ this becomes $\left(1 - \frac{M}{n} - e^{-\frac{M}{2}}\right)$

For small j, it becomes $1 + \frac{M}{n} - \left(1 - j\frac{M}{n}\right) = (j+1)\frac{M}{n}$

i.e. product is $\frac{n}{M} \cdot (j+1)\frac{M}{n} \simeq j+1$. | small j means $\frac{jM}{n} \ll 1$

so: $\frac{M}{n} \ll 1$, $j \ll n$; [if $\frac{jM}{n} \gg 1$; ratio = $\frac{n}{M}$; may be compositi since $j \ll n$; $M \ll n$; $\frac{n}{M} \gg 1$ may be o.k.]

Ratio = $\frac{n}{M}$; If $\frac{jM}{n} \ll 1$; ratio = $j+1$. ← linear

< 1 may be o.k.

if $\frac{jM}{n} = 1$; ratio = $\frac{n}{M}\left(1 + \frac{M}{n} - \frac{1}{e}\right)$ which is $.63\frac{n}{M}$

Then $\frac{M}{n} \ll 1$, $j \ll n$ implies $\frac{jM}{n} \ll 1$

which implies ratio = $j+1$

NO! $\frac{n}{M} \gg 1$; $M = j = \sqrt{n}$; Then $j \ll n$ $\frac{Mj}{n} = 1$.

Hence J is t. no. of problems backtracked.

T. rounds. are essentially that all terms are about =.

→ T. answ. should be J, not j+1 :

Back to t. error of 26.38:

$$\text{Sum} = \left(\left(1 + \frac{M}{n}\right)^{j+1} - 1\right)\frac{n}{M} - 1$$

ratio of sum to last term:

$$\frac{\left(\left(1+\frac{M}{n}\right)^{j+1} - 1\right)\frac{n}{M} - 1}{\left(1+\frac{M}{n}\right)^{j+1}} = \frac{1}{\left(1+\frac{M}{n}\right)^{j+1}}$$

$$= \frac{n}{M}\left(1 + \frac{M}{n}\right) - \frac{n}{M}\left(1+\frac{M}{n}\right)^{-j-1} - \left(1+\frac{M}{n}\right)^{-j-1}$$

$$= \frac{n}{M}\left(1 + \frac{M}{n}\right) - \left(1 + \frac{n}{M}\right)\left(1 + \frac{M}{n}\right)^{-j-1}$$

for $\frac{M}{n} \ll 1$, $\frac{JM}{n} \ll$, t. result is J, as expected.

$$= 1 - \frac{1}{1 + \frac{jM}{n}}$$

$$\simeq 1 - \left(1 - \frac{jM}{n}\right)$$

$$\simeq \frac{jM}{n}$$

$$\sum_{k=1}^{\infty} \left(\frac{1}{2}\right)^k = ?$$

$$\frac{1}{1 - \frac{1}{2}} = 2$$

Another task! That the things being computed are absolute probs.

As such, $(pc)^{-1}$ of < 1000 is usually not much of a threat!

E.g. in t. decrp. example, t. ~~result of j inverse~~ $\simeq (pc)^{-1}$ of $\simeq j$ for t. sum is of 1 for t. largest term, means that I would had $\simeq pc = 1$ for each problem & could just solve it immediately!

- clearly no threat!

Lets look at a situation in which the largest term is ~ 1000 ; look at worst case : how much larger than 1000 can the sum be ?

($M \equiv m-1$) .

from 25·20 : $(m+n)$ · pc of $\beta^n D$ is $\frac{n!\,(m-1)!}{(m+n-1)!} = \frac{n!\,M!}{(n+M)!}$

for $M=1$, its $\frac{n!}{(n+1)!} = \frac{1}{n+1}$

" $M=2$ " $\frac{n!\,2}{(n+2)!} = \frac{1\cdot 2}{(n+1)(n+2)}$ | for $M=3$: its $\frac{1\cdot 2\cdot 3}{(n+1)(n+2)(n+3)}$

T. Let's list the $M=1, 2, 3$ and $n+1$; $\frac{(n+1)(n+2)}{1\ 2}$; $\frac{(n+1)(n+2)(n+3)}{1\ 2\ 3}$ . . .

$(1+1)^X = \sum_{i=0}^{X} \frac{X!}{i!\,(X-i)!}$ | $(1+1)^3 = \frac{1\cdot 2\cdot 3}{1\cdot 2\cdot 3} + \frac{3\cdot 2\cdot 3}{1\cdot 2} + \frac{1\cdot 2\cdot 3}{2\cdot 1} + \frac{1\cdot 2\cdot 3}{1\cdot 2\cdot 3}$

these are the least factors of β^n.

$= 1 + 3 + 3 + 1 = 8$.

.18 A possible way to get a **Bad Result:** largest term = 1000 :

j is very large; $m >> j$; $m << n$.

So all of the terms are of the values $\left(1+\frac{m}{n}\right)^k$; $\left(1+\frac{70}{10000}\right)^k$

$(k = 1 \mid J)$. Say $j = 1000$ $(1000)^{\frac{1}{1000}} = 1.007$, so $\frac{m}{n} = .007$

$(1.007)^{1000} \approx 1000$.

$\sum_{i=0}^{1000} (1+.007)^i = \frac{(1+.007)^{1001}-1}{.007} \approx -1$

$\approx \frac{1000}{.007} - 1$; so ratio of entire sum to largest term is $(.007)^{-1} = 140$ so complete backtracking takes 140 times as much effort as largest term.

.25

How common is this Bad Result? Well, say one has 1000 almost certain choices (probs all ~ .993). Have $m = 70$ symbols (70 possible choices) with $n = 10{,}000$; So I did a sep. of 10,000 choices since I going to backtrack only 1000. ($m = 70$, $n = 18{,}000$ need not nearly be that large; $n = 2000$, $m = 14$ might be O.K.).

In general, the result of .18 is that if L is the $(pc)^{-1}$ of the largest term $(1+L^{-j})^{-1}$ is the desired ratio : $(L^{-j}-1)^{-1}$

for $L = 1000$, $j = 1000$ we have $L = 1000$: $j = 1000$ $L^{\frac{1}{1000}} = 1.007$

$(.007)^{-1} \approx 140$; for $L = 10^6$; ratio = 70 only.

It's my impression that this may be the only way to get a "Bad Ratio" for backtracking.

for $L = 100$; $J = 100$; Ratio = 2.1 .

3·16·82 TS

Superficially, it would seem that the situation of 28.18 would be unlikely! i.e. we make a step of 1000 or 100 choices of arbitrarily by pc., Then we go back & try to backtrack. Probs like .95 or .99 in a long seq. are pretty much like normal logical deduction, & usually one doesn't question that sort of thing.

No: Not so unlikely

It may be that the situation of 28.18 is easy to detect & one can then jump back in >> 1 step to previous "in view of"'s

Def Perhaps even more generally, one can detect when the "backtrack ratio" is relatively ≥ (the cc ratio between backtracking cost using all intermediate probs. steps, & search cost of the last prob. in the backtrack.)

Or devise methods of backtracking that would automatically skip many probs if pc ratios were small: The

.17 T. "servo" idea of 22.30 would do just that! — However, it shouldn't be normally used, because I suspect that only in situations in which the pc's of prob solns are very crossed (, would the servo idea be of value.

A possl. situation in which 28.18 occurs: TM has devised a v.g. heuristic algm that works very well, & uses it on a very large bunch of problems, w. success. But the algm. doesn't always work in general. (Tho it has always worked for probs gen. in the past — so, say the pc. of $1 - .007 = \frac{993}{1000}$ could be due to a sss of 1000 w. strip offs for the "other side".

.25 3·17·82 More General Q than "the Bug": The "soln" of the bug suggested by 22.01 and by the approx. quant. analysis of 25.01 – 28.25, is that if one has the cc available, backtrack as far as one can w. that available cc. However there remains the Q: How many solns. should one obtain for the problem that is maximally backtracked? One has, say, the alternative of searching for 1 soln. to a problem "Backtracked 15 probs"; or get 10 solns to a prob. backtracked only 8 probs back.

In general, one has a chance of getting a hyper pc soln. to a problem by either (1) using a hyper LC limit for that problem or (2) Being less El., backtracking a bit & using the hyper available LC on the backtracked probs(s).

Consider 3 params in search of a particular problem: "L" "P" "B"

① L cost # limit ; ② Pc obtained ③ # How far back in backtracking that search was component

One impt. pt. For a given L, and accepting first solns found for a particular problem, P is a non↓ funct of B. This if true, is because t. search .01 soln. becomes less el. as B ↑, so t. soln. obtained cannot get worse. .05

T. exact meaning of .01 - .05: That C (C is a code) exists for B = B_0; Pcon for B = B_1, with $B_1 > B_0$, t. same soln. C also exists; w. about t. same Pc; t. P is! is its cc much diff? Well, we can actually figure out what t. cc should be, since we have already gone thru all t. calcns. necy.!

Here α & β are sequences of problems.

Σ | 1 | α | 2 | β | 3 |

SN My impressn: That Pc is usually more impt. in Lc than cc is.

3.19.82

Σ Pc1 Pc2 ... Pcn
α_1 α_2 ... α_n
cc1 cc2 ccn.

Ingoing forward (→) from Σ we get conditional Pc's of Pc1, Pc2 ... Pcn, & respective incremental cc's of cc1 ... ccn. So, there exists a coder from α_1 to α_n .25 w. Lc. of $LC = \frac{\sum_1^n cc_i}{\prod_1^n Pc_i}$.

TC. Lcost for α_n was = $\frac{cc_n}{Pc_n}$.

T. total Lc if finding t. el. codes was $\Sigma\left(\frac{cc_i}{Pc_i}\right)$

If all $\left(\frac{Pc_i}{cc_i}\text{'s}\right)$ were t. same, then Σ t. Lc for this would be $\frac{n \cdot cc}{(Pc)^n} = n \cdot \frac{cc}{Pc} \cdot \frac{1}{(Pc)^{n-1}} = \left[\frac{cc}{Pc}\right] \cdot \frac{n}{(Pc)^{n-1}}$

can be quite large.

$\frac{cc}{Pc}$ is t. Lc of a single el. soln.

$\left[\frac{n\,cc}{Pc}\right]$ = Lc of entire set of el. solns.

This would be $\left[\frac{n\,cc}{Pc}\right] \times \frac{1}{(Pc)^{n-1}}$

Actually, in fact, we know the pc's & the cc's, so we know this upper limit of 30.25 on the LC of this backtrack soln. We can obtain such upper limits for any distances back of backtracks that we wish.

However, contrasting those upper limits w. the results of greater depth (i.e. 2nd or 3rd or hyer order solns) at zero, 1 or or any particular distance back, is a problem we don't know how to solve.

There is, hrr, an upper bnd. on the Δpc that we could get for deeper solns" of α_n : i.e. $\prod_{i=1}^{n-1} pc_i$: we would get this pc. if we found a soln. for α_n w. $pc_n = 1$.

Similarly, by backtracking a distance of k problems, the max we could do to improve the pc, would be by a factor $f = \prod_{i=n-k}^{n} pc_i$! (even w. $cc = \infty$) & of course via 30.25 we know we would obtain at least 1 soln. w. LC = $\sum_{i=n-k}^{n} cc_i / f$.

3.21.82 The general goal: Gn. a certain cc to expand, to find a strategy giving the greatest expected value of Δpc.

N.B. that going further back in some sense "covers" all lesser backtracks! e.g. Backing up 2 units will cover all solns. found at Backup 1 unit. if one uses a larger amt of CB.

.9A 12V
.6A 5V

A possl. search method: Back up 1 problem. Use CB_0 for LSearch = $\dfrac{cc_{-2,-1} + cc_{-1,0}}{pc_{-2,-1} \cdot pc_{-1,0}}$. If nothing better is found,

Use $CB_1 = 2 \times CB_0$, & Backup to *, until $\dfrac{\sum_{i=k}^{0} cc_{-i,-i+1}}{\prod_{i=k}^{0} pc_{-i,-i+1}} \geq CB_1 = 2CB_0$

When ever we are unsuccessful in finding a new better soln, we double our CB & Backup until the est. LC obtained from the old el. soln. was $\leq$ that CB.

If we find anything good, then we continue Backtracking as before, but all the CB values are changed, since the pc & cc for that particular point is changed. (My mind is razed on this point!)

.35 → Another, More specific goal: That if the cjs of a certain abs = α; & we use do the LSearch w. a CB of $\geq \alpha$, we are certain that we will decr. that abs.

3·22·82 T.S.

31.35 seems like a good idea! However, it may well be that it is satisfied by the normal initial search method of assigning the "above-threshold" CB to each problem. The difficulty seems to arise with the β^{12} vs. loop soln because the CB of the loop soln must be defined differently, before we will accept it. No! — If we used the above threshold CB for each problem, we would, indeed, determine the loop soln. I think the trouble may arise because we don't want to use that large a CB on all problems — it seems very wasteful! — usually as we "solve along" our TS, we find solns that are "adequate" — in the sense of large pc. — & we are satisfied with them — so we don't go to larger CB's in our searches. However, occasionally we do have the CC available, & so we want to go back & review the past & try to find a better soln (or set of solns) for the past set of problems.

.20 (SN) On the other hand, maybe the occasional Backtracking thing isn't any better [better] in saving cc! ~~If we back track N problems,~~ say our loop soln required the Lc of A to find, if the loop soln were applied to each problem. Now, If we backtrack N, it will require a Lc of $\tilde{\sim}$ NA to find the loop soln. This is because it takes N times as long to check the loop as a "trial soln" — since there are N problems, rather than just 1. — So, superficially, it would seem to take about the same cc if we applied the CB of A to every prob. that was solved!

Maybe .20 is not exactly true! In theory, for every new trial operator, one should try it on all past problems. In fact, what one does is find a particularly untractable subset of past problems to try candidate solns. on, since they are able to discard bad trials most rapidly.

$x \to 2x$ all x
$x \to 3x+1$ odd x
$2x+1 \to 6x+4 \to 3x+2$

Will t. foll. work? ([redacted] It looks like a not particularly economic. soln.)

Say, for every problem one uses CB = A, & one returns all of the solns. [redacted] — discarding only those that fail for t. new prob.

.04 T. way this is done: Say before t. present problem, one has solns $\alpha_{1,2,3}$; w. pc's $p_{1,2,3}$. One adds on to those solns. & one tries them in pc order — so if p_1 is greater than p_2 by a factor of 2^5 [redacted], we will probably try about 2^5 [redacted] continuations of α_1 before we form trials by [redacted] continuing α_2. The usual Lerch Algm. is used. → 3 4.22

.10 My impressn. is that this would work, but it would be expensive on 2 counts: (1) storing all those solns. (2) using CB = A on every problem. (The 32.20–.40 suggests that normally, one has to use CB=A on all of t. problems, anyway ... if its a Operator Induction problem — if true, then expense #2 may be illusory).

.21 The idea of backtracking, hvr, is using CB = A on that backtrack has t. advantage that one has a larger no. of problems to try t. soln. on, so one ends up not storing nearly as many — since many more of them are discarded by this larger "sub-corpus". Also, one doesn't have to test each trial on all of t. new probs, since usually trial solns. fail quite early in t. set of probs.

So, t. idea may be something like this:

.25 

I guess A is t. cc available for this backtrack.

Starting at α_0, one has coded t. initial corpus, Σ, up to α_0. One then incrementally codes problems 1 by 1 up to α_1, accepting the first or second soln. that occurs. Using $\Sigma\alpha_0$ as reference, when one have solved enuf problems so that $\frac{\sum cc_i}{\prod p_{c_i}} = A$, we

.36 backtrack to α_0 & use CB = A to code t. entire seq. from α_0 to α_1.

.37 At α_1 we continue as we did at α_0, but using t. "beginning" $\Sigma, \alpha_0 \to \alpha_1$ instead of $\Sigma\alpha_0$; This takes us up to α_2, etc.

So; $(\alpha_0$ to $\alpha_1)$, $(\alpha_1$ to $\alpha_2)$, $(\alpha_2$ to $\alpha_3)$ ... are then new "Macro problems" in t. corpus. They are each solved individually, once

after t. other. Each problem is delimited in such a way that all t.
.02 macro probs are of about "equal diffy" in some sense.
.03 An objection to t. idea of 33.21 ff: That t. points $\alpha_{0,1,2,3}\ldots$ are obtand in this way, to make t. "macro probs" of ≃ Leost, but these α_i dividing pts. are not nec'ly ~~obtain~~ good ways to divide up t.
.06 corpus. They have no meaning w.r.t. t. problems themselves.

((Actually, in Human prob solving, the division pts α_i, are obtand largely by considering t. probs themselves.

A way to avoid needing large storage, avoid "arby" pts. for α_i, but use much more cc:

Σ α_0 1 2 3 ... α_1 1' 2' 3'

start as .21 & go to .36, ~~But then we incrementally code 1 micro problem~~

Next, we take the first problem after α_0 and using t. soln. of $\alpha_0 - \alpha_1$, we code as far as we can into t. future, until $\frac{\Sigma\, cc_i}{\prod p_{ci}} = A$.

Using this new soln., we go back to t. next problem & code into t. future as before. [This is rather vague, I'm not sure it all means anything.]

7083A ~~6~~ lite Bn 10:08A.

.22 : 33.10: Say one starts w. t. code up to Σ & one codes 1 extra problem using a larger $cB \neq A$. One retains all solns found in memory.

[In working t. next problem, all solns. thus far are first tried on it — overnight in order of pc. This will probly distroy a few solns. Hvr., this checking of old solns takes little cc. (relative to A).

Next we do an Lsrch on solns. that are close to old solns. 33.04 – .10 is t. rite spirit, but is not literally correct. Close means small conditional
.30 entropy relative to t. old solns.

Who, with a single poetic phrase changed the jingle jangle jungle of American poetry into a garden of ineffable ~~unspeakable~~ delight.

Say one has $\alpha_1, \ldots 14$ are problems following Σ. First One codes the solns
.32 Σ ① $\alpha_1\ \alpha_2\ \alpha_3\ \ldots\ \alpha_{12}$ | ② $\alpha_{13}\ \alpha_{14}$ incrementally as $\beta^1, \beta^2 \ldots \beta^{12}$

Then we decide to backtrack & we code $\alpha_1 \ldots \alpha_{12}$ as "loop".

So we now have 2 codes from ① to ②: The $\beta^1 \ldots \beta^{12}$ code & t. loop code which has > pc than. In coding α_{13} from point ②, we will continue adding (or modifying) both codes, but will put more cc on t. loop code because it has more pc: (see t. discus. of 33.04 – .10 & 34.22 – .30)

As we continue coding α_{13}, α_{14} ... we use small incremental ("ed.") codes. Finally, we get to α_{17}, say; we have [redacted] cc available for meditation (= Backtracking). If we have enough cc. available, we go back to (1) & use the loop soln. & proceed from it. If we don't have that much, we just go back to (2), or if we have conditional codes contingent on B^*, B^2, ∴ B^{12} (A hope we will unless the pc of loop is >> that of $B' \neg\neg B^{12}$), we can backtrack to some point between (1) & (2) (Fig 34.32).

.10 In general, Backtracking when one has several ll codes of the past, is a "standard procedure," based on 33.04-.10 & 34.22-.30

The .10 seems not too bad, it may still suffer from the points to which one backtracks are not much determined by the strs. of the abss. being learned. (This is the diffy of 34.03-.06) — Nvr., this latest soln. seems better in this respect than the methods of 33.25 ff.

.19 In fact, it is bad, for this reason! Consider fig. 34.32!

Say one had incrementally coded thru α_8, then backtracked at that pt. because cc was then available. Say Σ was problems in learning simple arith. notation like 3, 4, +, ect.

.25 So, we backtrack "some distance into Σ" & try to code thru α_8, using CB = A. The "A" is enuf to decr the "loop soln." by itself, A isn't enuf if several problems of Σ also have to be solved. So we haven't decrd the loop soln. yet!

Next, after this backtracking, we continue from α_8 on, incrementally. When we backtrack an amount (CB = A) again, we may or may not discover loop soln.: My guess is that we would probly decr it — if the backtrack went as far back as α_8. I guess one must assume that the periodic "meditations"

.30 decrease freq. enuf & have at least enuf cc allowed in them, to enable

.31 overlap of the "Backtracked sequences".

A kind of model for this search task: One has a bunch of n problems & one is allowed a mean cc per problem, i.e. a certain total cc for the set. How best to distribute this cc? Presumably, the total cc. must be large enuf so the overlapping of .30-.31 is poss'l.

.01 → Viewed this way; & also using the idea of 35.10 for back tracking ll codes; it may be that the idea of 33.21 — 34.02 is not so bad. I'll have to go thru it carefully to see if the objection of 34.03 holds & more specifically what flaw of analysis of 35.19 — .31 will destroy it.

The method is like 33.21 — .36, but at .37 instead, we save the ll codes & backtrack code incrimentally like 33.04 — .10; When we backtrack, we do not process ll codes as 35.10; 34.22 — .30 The idea is of new trials being epistemologically "close" to old solns. is impt., but I don't know it I want to use that more exact approach just now.

Another idea for a trick: In general, the problem of having to save too many codes can be dealt w.: By coding to only small CB, the first time thru a corpus. Then we recode that portion of the entire corpus to greater depth.

.20 Not exactly: Consider $\Sigma \alpha_1$. We code α_1 wrt Σ to a depth so that we only end up w. ≤ 100 codes to store. Then, we code $\alpha_1\alpha_2$ wrt Σ to greater depth, but we still only again retaining only ≤ 100 codes. Hrr., we can do this to greater depth ($\subset \Sigma$ CB) because the large ssz (α_1 & α_2) rejects more alt. codes. Then we code $\alpha_1\alpha_2\alpha_3$ wrt Σ to a depth to obtain ≤ 100 consist codes again, etc.

100 k
20
5k
3k
2k

.19 ff isn't exactly it, either! I think the idea is this, that if one has a coding algm that is characterized by a certain set of params, that ends up w. too many codes to store, Then we only use that set of params of the algm. When we have enough ssz to ↓ the storage needed to acceptable levels. This ↑ssz is normally obtained by "waiting for it to come in".

In fact 00 Can B1 B2 M1 M2

→ Well, looking at 35.19 — .31 this ll coding may fix it:

Using some structure as $\Sigma_1 \subset \Sigma_2$ $\alpha_1 \ldots \alpha_8 \frown \alpha_9 \ldots \alpha_{17}$

≡ ll some is true enough to Σ" of 35.25

Well actually!: 35.19 — .31 didn't show that the raw ≡ no ll codes contained methods of 33.21 — 34.02 was nearly bad! .... I really need a more detailed analysis!

.01 Consider t. 2 code sequences $\beta^1\ \beta^2\ \beta^3 \ldots \beta^{12}$ ① : ② loop

The sequencer ① is a way to store a lot of codes i partial codes . . .

"partial" codes being codes that work up to a certain pt. in t. corpus.

t. seq. ② also does this.

The codes of ① are particularly useful in back tracking, because they make it possl. to go back an ≈ graduated amt. of pc. — i also know how much cc i LC is involved.

So, a kind of approach to summarize (?) !

At each point in t. coding process, one has in storage, a bunch of code sequences i fragments for t. corpus, that, taken together, code it up to a certain pt. Those sequs i fragments include ll codings. Info about pc's i cc's of t. various parts is also saved.

examples of things that might be retained are t. ① i ② of .01

Having such a mass of info in memy, one is given a certain amt. of cc, (≡ cB) i one is asked to take t. corpus i this stored info i use as good as possl. ee pc for t. corpus, using ≈ that specific cc (cB).

This is essentially t. problem of How should I best Backtrack?

If t. cc (cB) is very small, one does usually single best conditional readings of each problem, based on t. code up to t. previous than t. previous prob.

If t. cc (cB) is much larger, then more like true back tracking is done.

Suppose we have t. 2 code sequs. of .01 !

$\beta^1\ \beta^2 \ldots \beta^{12}$ are successive solns. to larger precas of t. corpus. They are also successive modifns of each previous soln.

loop is a soln. like β^{12} that fits as far as β^{12} does . . . . but needs no modifn. to fit much further along in t. corpus.

so! say ■ to pc of loop is somewhat > that of β^{12}

If loop didn't solve t. next problem, then we would first try to modify loop to work. When t. modifn's pc ≤ (pc loop) − (pc β^{12}) as ■ smallas entire pc of β^{12} including pc's of all modifns from β^1, β^2 . . . .

then we start modifying β^{12} for trials.

—— At what pt. does we start modifying β^{11} for possl. trials?

—— One possl. diffy: That possl. modifns of β^{11} were already tried when "loop" was found by t. " " backtracking.

an early demo of computer response to Human language

7 kw

210
70
−140 OP

7.5 kw →
−150 OP

−50 →
4 kw

3·26·82 IS

Hvr., that's o.k.; we don't work on modifns of β'' unless we plan to use a larger cB (or equivalent cB) than was usd befor.

$\Sigma_1 \cdot \Sigma_2 - \beta^1\ \beta^2\ \beta^3 \ldots\ldots \beta^{12}$.

Loop

Very impt, very good idea: see 40.23 for comments

.07 A possibly good way to study this genl. problem: Make 1 (or several) models of t. search space, telling how much cc & pc are needed for various obss. along t. corpus. Then, given a certain amt. of statistical info about t. corpus (cc knowing its true structure) what is t. best search strategy? e.g. gn. a certain amt. of cc., how to get max expected Δ pc from it in search.

E.g. say, in fact, ~~we have coded the~~ there is a long sequence of problems. (say >20 probs) each w. incremental LC of 2; incremental pc = x.707 → so pc = .001 for 20 probs. Hvr., if one starts after t. 12th problem, a soln. is obtainable at pc = 1/30; LC = 1000

.20 { NB. These exact figures may not be good examples: T. stoch grammar of ~ 25.01 ff may give more reasonable values ... }

SN Also in many searches of this sort, testing ~~putting~~ candidates in utility/probability order is optimum

ΔLC = 2

each prob. has cc = 1.414, pc = .707 } for incremental solns. 20 problems | total pc = 1/1000; " cc = 1.4 × 20 = 28

for corpus of anysequence of 12 problems, ∃ a loop soln. of pc = 1/30, cc = 30k; LC ≈ k.

.26 What one knows apriori about t. search space: well, say one knows t. each of 20 probs have ~~LC of 2~~ incremental pc = .7; cc 1.4, ΔLC = 2.

Better have each incremental pc = 1/2, cc = 1, 20 probs: max But note .20

a ~~natural~~ loop soln. for any ~~subsequence~~ subsequence has LC = 1000; pc = 1/1k so cc = 1.

incl. soln for 10 probs has pc = 1/1k; cc = 10 so LC = 10k.

What one knows apri: as .26 ① that t. 20 probs have incremental pcs = 1/2 & cc = 1, because this is inexpensive to determine, & is a useful thing to have around — since t. solns. are fairly useful, (tho not as good as loop, say).

② That there exists a soln. of LC >> 1 that has pc > 1/1k

.35 for t. entire seq. of 20 probs (or any subseq of t. 20 probs). 39.22

SN Perhaps this $LC = \frac{cc}{pc}$ is not such a good upper bnd for search: since so many trials drop out well before t. max allowable cc is spent on them, yet they contribute perhaps almost all of t. cost of srch. This seems to be not relevant in t. most genl case hvr.

See 39.01—.21 for discn. & conclusn.

For searches in which $\Sigma pc = 1$ (or ≤ 1) (e.g. Operator induction [rather than sequential induction]),

we have $\frac{cc_i}{pc_i} < A$, say so $cc_i < pc_i \times A$ $\therefore$ $\Sigma cc_i < A \underbrace{\Sigma pc_i}_{\leq 1}$.

Now, the inequality $\frac{cc_i}{pc_i} \lesssim A$ occurs usually only if the trial doesn't converge in the required CC.

However, in doing an L search, we can keep track of the actual cc expended in doing an L search of $CB = L_0$: perhaps there is an L_1 constraint.

The ratio may depend very much on what kind of search it is: [In some searches,

type 1 all trials converge & take about the same time for each. [In such a case,

if pc is normalized, the search time is $\lesssim$ [cc for 1 trial / pc of successful trial (the highest pc].

In this case A, the CB used, determines the lowest pc that we can reach.

type 2 If most trials — or [most] trials of type — don't converge in required CB, then the total search cc is $\lesssim$ [cc for successful trial / pc " " "].

.18 type 3 If most type trials converge in $\ll$ C.B. because of observed failure, total search cc $\ll$ [cc for successful trial / pc " " "]

The moral: What I want now is something that's very general, so → 41.23

.21 the probs. will not in general be of type 3: so search time $\lesssim$ [cc for successful trial / pc of successful trial].

.22 : 38.35 : I think a more realistic part of the idea of the search space might be that e.g. all probs in the T.S. can be solved w. C.B. < A, since the T.S. was constructed w.

.25 this constraint, the value of A is not actually given to TM, however.

An additional possible constraint in writing the T.S.: That it is also possible to solve the TS using a total cc < B. This means that the average cc per problem has an upper bound. TM is not given B, but may have some idea of B/no. of probs.

The constraint of .22 – .25 isn't entirely clear. The sub-corpus

.34 size is not stated. [In this discussion, by "SSZ", I mean the no. of probs that we have coded, past the initial starting pt.

.35 A possible search algorithm for the "B" v.s. loop "problems" is different:

Start by using CB = A on a single problem. But wait until ~ 4 or 5 probs have come in so that one can reject A.H. solns, & not have to store too much in memory (once we've seen all solns. found). Before SSZ of 4 or 5 is obtained, use a small SSZ for search. After CB = A has been used on 1 problem, wait until SSZ gets much larger. This will

Not Bad! At least Glorious!

3.27.82 TS

.01 further ↓ no. of solns. in Memy. Eventually, w. large enuf SSZ, no solns. will work. Usually this will occure suddenly: i.e. for SSZ = 20, say (≡ 20 probs into the "future" off. prob. one spent CB=A on), one has 10 solns., then for SSZ = 21 no solns. work. This 21st prob. then means a new kind of problem has been gen.; i we will try a new search w. CB = A, using the 10 solns of the 20th prob. "to be modified" for trials.

.09

One funny thing about 39.35 ff is that it sure doesn't look like the way Humans do it! (.12-.22 seems more human-ish — or can be pushed in the direction of being more like human problem-solving) This soln. seems almost "too good" i.e. it also performs the "bunching" of problems into subsequs of probs. needing ~ solns. (.29-.35 seems even more human-ish)

.12 39.35 ff can be streamlined a bit by looking for a set of the first 10 solns for SSZ = 1 (or fewer solns. if one hits CB = A first!); then retain for SSZ = 2 ↑ CB until 10 solns is retained (or until CB ≥ A. Then for SSZ = 3 ↑ CB " 10 " " " " or until CB ≥ A, etc. Continue doing this w. greater SSZ, until no solns. are retained, i CB = A. Then SSZ ← SSZ-1 : the resultant solns. are then to be modified for trials for the next problem, loop to for the next batch of problems. ("streamlined" means it uses less memy (i.e. less cc).

.22

.23 Anyway, whether 39.35 ff is correct or not, the approach (≡ methodology) of 38.07 - 39.34 looks very good. If I ever get into trouble in this area, be sure to use that approach. Putting the problem into that form really must clarify the whole thing tremendously!

.29 A shorter modcln. of .12 - .22: instead of retaining finding & retaining 10 solns. at each point; just retain 1. Then as soon as no soln. is obtained because of excess SSZ, step back 1 problem & use CB = A & retain all solns. Then modify this set of all solns. as trials for the next problem .... etc.

.35 43.08

The method does seem economical: It doesn't seem to take much more cc than is absly necy! Also .12-.24 & .29-.35 save on amt. of memy needed for storing codes, and make it closer to what a human might do: (.29-.35 does this, more).

3.28.82 TS

.01: further Diftys w. t. method: It would seem to work only if t. problems are properly bunched up in types: like t. kind of t.s. that a Human teacher would write for a human student.

As I see it, it works only for MTM problems (essentially non-problistic problems) —— One outgrowth of this: if there are one or more errors in t. T.S., then it wouldn't work.

Errors in t. T.S. can be del't w. in several ways:

.11 ① If occasional problems occur that don't fit an otherwise v.g. algm., then just write in A.H. solns. of those probs. [For an operator T.M., we have a Σ/ol table for "errors". If there are only a few errors, t. table doesn't cost much in cc or pc or Lc.]

② Looking at t. set of such "Don't fit" problems, is there a set of hy pc xfmns that can xfm them into probs for which t. usual algm(s) fit(s)? These xfmns would be ≈ "error correcting" algms.

[for sequential induction, errors must be handled somewhat difntly. Maybe use a list of "dates" & sub. corpi for these dates]

③ We may include in TM's repertoir of xfmns some hy pc. xfmns that correspond to t. kinds of errors humans make in writing TS's. ∴ ff is ≈ a human way of dealing w. errors in T.S.'s. & is easy to do.

.22 → 42.01

.23: 39.21 A lot more work could be done on making better estimates of t. amt. of cc needed for an Lsrch. 39.18 is perhaps t. most impt. deviation from t. simpler formula — but this can be developed in more detail — when does it occur & how much, just what kinds of problems does it occur in ... Give broad areas of examples.

Also, usually $\Sigma p c_i << 1$ (probly), so t. normzn constant can give a further ↓ in cc needed for t. srch. I don't know how large this constant is. Can it → ∞ for certain kinds of systems? We don't actually need to know t. normzn. const, but we need a lower bnd for it.

200 kwh.
≈ $20
So lite bulbs cost 10% of Elecy
So 67¢ saved on cost of bulb.
83¢ lost in elecy cost for 1k hrs.

.01: 41.22: Re: Non MTM probs (≡ NMTM): There are 2 kinds of problems that I've considered: (1) small corpus: TM tries to find a codes for t. entire corpus. [corpus can't be very large, since pc of corpus can't be too small, because cost of searching ∝ 1/pc]

.04 (2) large corpus: TM tries to find a model or PEM or CPM to fit t. corpus well. Th. Σ pc of these models is ≤ 1.

Gen c is pc of model x (pc of corpus wrt that model).

.10 (3) A ≈ third kind is similar to t. small corpus, but t. search method may be different. T. structure of t. ~~corpus problem~~ trng. seq. is like that of the T.S. for "Eval" (≡ evaln. of Alg. expressns).

The method of search / I'm thinking of is ≈ that of 39.35 - 40.22. T. Q is, can this search method be used for stochastic corpi, & if so, what kinds of stoch. corpi? How can I characterize them?

In t. case of [MTM-type corpi for which] is adequate, I can

.22 characterize t. corpi.

.21 Actually, 39.35 - 40.22's corpus looks much like a NMTM "large corpus", in that we devise a model & then see how it works on a long string of problems. [In both t. construction of t. model is about t. same; usually using stoch grammars, so Σ pc < 1.] T. difference betw. MTM & NMTM corpi, however, is that in MTM, the fit is always "yes-No"; In NMTM, t. fit always occurs, but it's given a pc value — & one tries to find models w. hy pc, that give hy pc's for t. sub-corpus involved. We want models w. max product of these ≥ pc's.

.32 For small corpus/ NMTM case similar to 39.35 - 40.22: we play it similar to [that's for MTM! i.e. we can't start at a] type (3) (≡ .10).

.34 particular point, then code out until we observe a "sudden ↑" in pc per symbol: At this point, we start out anew & try for a new code for symbols following that pt. This will be conditionally codes based on t. code(s) before then.

In both t. NMTM probs of. 42.21 & .32, the choice of CB, i.e. deciding how far to code before changing models (or starting a new conditional code subseq) is unclear. The criterion for "sudden ↑" in 42.34 must be quantified.

.08: 40.35 A (perhaps) modifn. of 39.35 – 40.22. One starts out at a gn. pt. in t. corpus. Then, one tries all codes in ≈LC order & one obtains for each, its cc, its pc & how far it was able to code t. corpus (≡ l). (Def) $(pc)^{\frac{1}{l}}$ is t. pc per symbol & will be of much interest.

.10 SN Re: [struck out] LONG CNMTM Corpi:

My previous idea on this search was that I would try t. models (≡ PEMs) in Lcost order, since, if I tried them in order of ↑ ((pc of them) x (pc of corpus wrt them)), t. Lcost would be far too large, since t. pc of t corpus wrt t. pem is very small. (of t. models alone; ccs)

Hvr., if I use only a specially selected small part of t. corpus, then I can use cc / ((pc of Pem) x (pc of part of t. corpus)) for LS rch & get reasonable cc. After obtaining a list of "promising" Pems, I can try them out on t. full corpus & use them in ‖ for predn.

.25 ⟶ The codes w. best $(pc)^{\frac{1}{l}}$ are then entered in ‖ for future coding. One then has all these codes that go out to difnt. distances (≡ terminal pts). Each terminal pt. is then used as t. start pt. of a new conditional coding of t. corpus. The shortest distances have most pc & so they are "continued on" first. Eventually, certain codes strings will pass by others in t. corpus, so we can compare their pc's, & discard some that are very far behind. (Tho I'm not sure how good that would be! we might well discard a loop soln. v.s. β^{12} in such a situation!).

N.B.

.37 In considering cc, do not count t. cc after t. model has fit t. first few problems. (unless the cc per problem is >> t. mean for t. first few probs). This is because we mainly want to use Lsrch to get rid of non convergent or otherwise very hy cc models. If a model works at all in reasonable cc per problem, then accept it & see how many probs it can work. : see 45.03–.14 for an improvement ← not merely an improvement: see 45.20 a/o clarifn. of this method.

The search Algm of 43.08 – .40 looks v.g. : it looks like a much less a/l (& less A.H.) method than t. 39.35 – 40.22 that inspired it.

A big Q is: can I modify this method to work w. short & long corpi of 42.01 – .22? Actually, I'm not sure just how real, practical NMTM corpi are modeld by t. long & short corpi of 42.01 ff — & just what .10 t. 3rd type of corpus is, is not very clear! I guess t. search (type) (have seen 42.32 – .40 also)

1/2 30 mpg . 3.7 $/mi

[redacted] divides t. corpus up into sequencial chunks & does a short corpus-type search on each chunk, w. t. codes being conditional .13 on t. preceding codes thus far. It's sort of "batch processing".

It would seem that the method of 43.08 – .40 could apply directly to t. small corpus (NMTM), and to t. sequencial (small) chunk corpus of 42.10 – .22, 42.32 – .40, 44.10 – .13 (≡ 3) w.o. modifns except perhaps t. lines 42.34 – .40 may not be directly applicable (ifstall!).

shuttle for comm runs maybe PLA. Fr. from 100mph Bomb.

It may well be that [redacted] I don't have v. g. methods of breaking a NMTM corpus down into "large NMTM sub-corpi" (of 42.04) 922

It may require some kind of hyper order reasoning to decide that certain parts of t. corpus (say in R.W.) are best coded in this manner. [redacted] in particular, e.g. numerical T.S's seem to be best coded in this way.

I can probably write a fairly useful T.S. w.o. getting any part of t. corpus in a form that needs this "large stock corpus" treatment.

Actually, this is a squiff. Update of 43.08 – .40! see 45.01ff

.30 On 43.08 – .40: sequencial NMTM short corpus: One way to implement it! → 45.15

Start à conditionally code along until there is an error. Back up 1 symbol of code à write t. "end" symbol, Δ; then continue w. a conditional code sequence as before until there's another error then loop. In general, t. pc of t. Δ symbol is ~ ∝ $\frac{1}{2}$, where l is t. no of successful code symbols since t. last Δ symbol. Each code sequence, with its "Δ's" has a pc à a cc, à once terminates a trial when its cc exceeds t. threshold CB (≡A) being used.

As described, t. method is completely characterized by t. param, A. → 45.01 spare

.01: 44.40 : Hvr. t. new method described in 44.30–.40 it's not ≃ that of 43.08–.40.

It is a new search method.

.03 A method closer to 45.08–.40! We start a code conditionally "interned" with a symbol (▲) until either we get an error (in which case we backtrack 1 symbol & write ▲) or until we exceed t. least cost CB, A in that subsection of code. — In which case we write ▲. Renew loop ■ to .

We finish a particular code sequence when we've coded t. entire corpus as far as we want to go. — Then we start over w. another code seq.

.14

.15 The trouble w. 44.30–.40 : It's really rather non-el. certainly more so than 43.08–.40 : T. diffy is that t. pc's of t. codes of t. corpus include many t. products of t. pc's of many sequences — so t. pc's found to be very small, i ∴ t. Lc to find t. v.g. codes ends up being very large

.19

.20 → The criticism of .15–.19 (applied to 44.30–.40) also may apply to 45.03–.14: i.e. it is necy. to first select out t. partial codes of hy $(pc)^{\frac{1}{2}}$ (as in 43.08–.40 — specifically 43.25) before continuing them. Otherwise, in .03–.14 we end up tracing thru too many codes — which is excessive cc.

.26 A possl. trim of 43.08–.40! First, using CB = A, trace out all codes as far as they will go. Picking t. code of max $(pc)^{\frac{1}{2}}$, continue that code, trying continns. in order of Lcost, until

.28

.29 t. $(pc')^{\frac{1}{2}}$ of (t. original plus its continuation) is no longer t. max of all known partial codes. At this point, start continuing t. next code of max $(pc)^{\frac{1}{2}}$ & do this until — etc, loop.

.32

.33 A diffy of .26–.32! I really don't know how practical it is: will it really code t. entire corpus in reasonable cc?

Also .28–.29 is not clear: for a continuation of a gen. code, one tries continns in a Lc order — but to what Lc limit? If CB = A, it would seem to be rather expensive to code this way — tho it looks like one would get a v.g. set of codes — perhaps at least as good as that of any coding method

① limited backtrack of say 5 symbols or pc of ≃ $\frac{1}{32}$.

② 44.30 is not ≃ 43.10–.40 — it's a new method!

③ use of $\frac{(cc)^{1-\epsilon}}{(pc)^{1+\epsilon}}$ as stop criterion. to emphasize by pc.

.01 Using CB=A for Search of sub-"short corpi" — (Tho perhaps expensive)

One way to deal w.t. dirty of 45:33 ÷ 40! Say one has spent A on obtaining a bunch of partial codes (using CB=A). Each of those codes/c_i is for length l_i of t. corpus and of mean pc/symbol = $(pc_i)^{\frac{1}{l_i}}$. Say one choses c_{85} because its $pc_{85}^{\frac{1}{85}}$ is low (or "nice"). T. Q is: can I afford to use as much as $\frac{A}{l_{85}}$ as cc/symbol of this sub-corpus? This will give an estimate of t. cc of coding t. entire corpus, i.e. it may be too low.

So what one may do: chose a tolerable value of cc/symbol.

Def .17 cc/corpus symbol $\equiv \alpha$. Find t. partial codes of lengths $\geq \frac{cc}{corpus} = \frac{A}{l} \leq \alpha$; i.e. $l \geq \frac{A}{\alpha}$. ⇒ pc/corpus symbol is max.

Use those partial code. (at least) (& perhaps some other good(codes).)

Essentially, what we find, is: all codes that go to $\frac{A}{\alpha}$ or beyond, that have $Lc \leq A$: we then are interested in those of highest $(pc)^{\frac{1}{l}}$ ≡ main pc/symbol codes. We could then just list them in $(pc)^{\frac{1}{l}}$ order, & continue those of highest order.

We can abbreviate t. normal Search; just try all codes w. Lcost < A — this will save a factor of 2/(maybe) in cc. If this search takes (squiffily) less than cc = A, then try 2A for t. CB, etc. Faster yet! If it take cc = $\frac{A}{\kappa}$ for t. Search, then use κA for t. C.B. — It's likely then that this will take cc ≃ A for t. search & waste ≃ 0 cc (if $\kappa >> 1$).

Another time saving thing! we just look at t. pc of all codes as they "go by" t. $l = \frac{A}{\alpha}$ point in t. corpus. This pc is ≃ a measure of t. $(pc)^{\frac{1}{l}}$ of that code — Tho $(pc)^{\frac{1}{l}}$ may be better because it's involved more w.t. future efficacy of this code.

O.k.: for the sake of argument, what do we now do w. this bunch of partial codes that got past $\ell = \frac{A}{\alpha}$? — How do we continue?

One way; pick the code of max $(pc)^{\frac{1}{\ell}}$ & use it to continue via conditional coding.

SW Suppose that one of our accepted codes is one that stopped at a particular ℓ value because [redacted] to CB, A was exceeded. Then we could easily continue it at (presumably) low cc/per symbol, ... should we use this continuation?

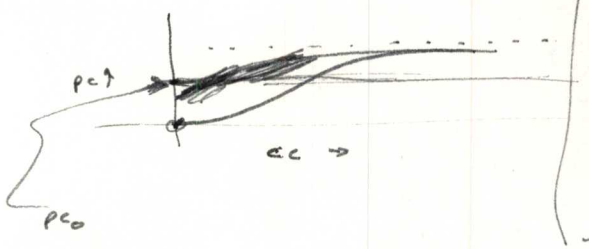
.15 Partial codes that get far past $\ell = \frac{A}{\alpha}$ should be given some preference, because they save us much cc. in coding .... — Enabling us to use more cc (at the $\equiv$ hyper CB's for search).

Also, if we have a partial code w. very hy pc, that doesn't quite code as far as $\frac{A}{\alpha}$, I should think we might want to use it.

.20 .15 – .20 might be that a good cut in the right way! for $\frac{cc}{\ell}$ (cc per symbol coded) = k, if k is large, we get large pc/bit.

So pc/bit = $(pc)^{\frac{1}{\ell}}$ is an $\uparrow$ funct of $\frac{cc}{\ell}$. Say I knew this

.20 funct. (in a ruff, probilistic way!) $pc^{\frac{1}{\ell}} = f\left(\frac{cc}{\ell}\right)$.

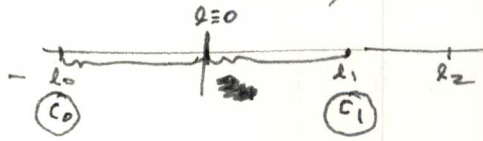


(0) Consider a code (C_0) that goes past $\frac{A}{\alpha}$ a distance ℓ_0 let it have a pc = pc_0.

(1) consider a code (C_1) that does goes up to only ℓ_1 before $\frac{A}{\alpha}$ (it doesn't get to $\frac{A}{\alpha}$) — but it has very hy pc pc_1, its $pc^{\frac{1}{\ell}}$ is > that of C_0

Let us extend C_1 so it codes $\ell_1 + \ell_0$ symbols further – up to C_0's code. The amount of cc that we expend for this extension will determine the pc of the extended code. Presumably, if we use enuf cc, we can get pc > that of C_0. — so it would be a uniformly better code than C_0, but it would cost more.

Now: Say we want to go past ℓ_0 (where C_0 has gone to), to ℓ_2.



If we used cc = B to code C_0 from ℓ_1 to ℓ_2, that's $\frac{B}{\ell_2 - \ell_1} = \frac{cc}{\ell}$; we get $f\left(\frac{B}{\ell_2 - \ell_1}\right)$ for pc/symbol

so the pc of the extended code is $pc_0 \cdot \left(f\left(\frac{B}{l_2-l_1}\right)\right)^{l_2-l_1}$

T. pc of the extension of C_1 would be $pc_1 \cdot \left(f\left(\frac{B}{l_2+l_0}\right)\right)^{l_2+l_0}$

so the Q is, which of these expressns is greater?

well, $f\left(\frac{B}{l_2-l_1}\right) > f\left(\frac{B}{l_2+l_0}\right)$ because f is an ↑ funct.

and a fortiori, $f\left(\frac{B}{l_2-l_1}\right)^{l_2-l_1} > f\left(\frac{B}{l_2+l_0}\right)^{l_2+l_0}$.

— So it would seem that $\frac{pc_0}{pc_1}$ would have to be very large to compensate for the inequality of these 2.

However, it could well be. Say l_2 is very large, so $f\left(\frac{B}{l_2-l_1}\right) \simeq f\left(\frac{B}{l_2+l_0}\right)$.

So the q. is: How does $\frac{pc_1}{pc_0}$ compare w. $f\left(\frac{B}{\quad}\right)^{l_1+l_0}$?

pc_1 tends to be $>> pc_0$ simply because C_1 is a worse code for a much shorter corpus than C_0 is.

If l_2 is large, $\frac{B}{l_2}$ tends to be the mean cc/corpus symbol $= \alpha$ (46:17)

Say $(pc_1)^{\frac{1}{l_1}} > f(\alpha)$; $(pc_0)^{\frac{1}{l_0}} < f(\alpha)$.

My impression is that under these circumstances $\frac{pc_1}{pc_0} > \left(f(\alpha)^{l_1+l_0}\right)^{-1}$.

So that continuing code C_0 would be more promising —

But check on this, in more detail!

More exactly — or: compare $pc_1 \cdot \left(f(\alpha)\right)^{l_1+l_0}$ with pc_0.

since pc_1 is greater than average for as far as it goes, & pc_0 is less than average for as far as it goes, it would seem that this is > this

So this inequality is true & the C_0 code is more promising.

I think the hypothesis $l_2 >> 0$ is fine — since we assume a lot of cc in the future.

So this suggests that we should usually chose the code of larger pc — no matter how short its corpus is!

However, this seems unreasonable, since we could then end up w. excessively cc to code the entire corpus!

So paradox!

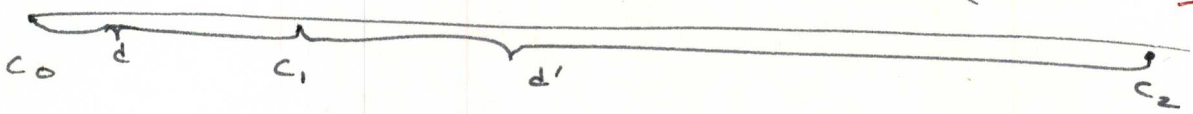
3·30·82 TS:

Well part of the drifty may be that I didn't consider the extra cc [redacted] that I'd expect to accrue to continue C_1's radar.

Say C_1 is a distance $d \equiv l_1 + l_0$ ahead of C_0.

Say we want to [redacted] code to a point d' past C_1 $(d' \geq 1)$, & we expect to expend cc = k to do this (k is a variable).

T. expected pc [redacted] starting w. C_0 [redacted] will be $pc_0 \cdot \left(f\left(\frac{c}{d+d_0}\right)\right)^{d+d'} \equiv pc_0 a_0$



T. expected pc at C_2 starting w. C_1 is $pc_1 \left(f\left(\frac{c}{d}\right)\right)^{d} \equiv pc_1 a_1$

~~Say, as $d \to \infty$, $\frac{c}{d+d'}$ is constant $= \alpha$.~~

$\frac{c}{d} = \left(\frac{c}{d+d'}\right)^{-1}$ $\qquad \alpha^{-1} = \frac{d}{c} + \frac{d'}{c}$

Say $d \uparrow$ but $\frac{c}{d}$ is const $= \alpha$

$a_1 = \left(f(\alpha)\right)^d$

$a_0 = f\left(\frac{c}{d+d'}\right)^d \cdot f\left(\frac{c}{d+d'}\right)^{d'}$

as $d \to \infty$ $f\left(\frac{c}{d+d'}\right)^{d'} =$ [redacted] $f\left(\frac{c}{d}\right)^{d'} = f(\alpha)^{d'}$

So $a_1 = \left(f(\alpha)\right)^d$; $a_0 = \left(f(\alpha)\right)^{d'} \times \left(f\left(\frac{c}{d}\left(1 - \frac{d'}{d}\right)\right)\right)^d$.

Say $f'(\alpha)$ exists : $\left(f\left(\frac{c}{d}\left(1 - \frac{d'}{d}\right)\right)\right)^d = \left(f\left(\alpha \frac{1}{d}\left(1 - \frac{d'}{d}\right)\right)\right)^d$

$\frac{f'(\alpha)}{f(\alpha)} \equiv \epsilon$. = log deriv. of f $f(\alpha)$

.256 $\qquad \doteq f(\alpha) - \alpha \frac{d'}{d}\left(f\left(\alpha - \alpha \frac{d'}{d}\right)\right)^d$

$= \left(f(\alpha) - \alpha \frac{d'}{d} f'(\alpha)\right)^d$

$\frac{a_0}{a_1} = \left(f(\alpha)\right)^{d'} \cdot \left(\frac{f(\alpha) - \alpha \frac{d'}{d} f'(\alpha)}{f(\alpha)}\right)^d = \left(1 - \alpha \frac{d'}{d} \epsilon\right)^d \cdot f(\alpha)^{d'}$

.32 $\qquad = f(\alpha)^{d'} \cdot e^{-\alpha \frac{d'}{d} \cdot \epsilon}$ $\qquad \alpha d' = d' \cdot \frac{c}{d} =$ expected c to go distance d'.

is this dimensionally reasonable?

.33 $\qquad = \left(\frac{f(\alpha)}{e^{\alpha \epsilon}}\right)^{d'}$ $\qquad$ So one compares $\frac{pc_0}{pc_1}$ w. this

to see whether C_0 or C_1 is better.

$\alpha \epsilon \equiv \frac{\alpha f'(\alpha)}{f(\alpha)}$

Now, $\frac{f(\alpha)}{e^{\alpha \epsilon}}$ is [redacted] known if $f(\alpha)$ is known, so let it

$\equiv g(\alpha)$. $\qquad$ What this does, then, is translate

a distance, d', into expected ratio of pc_0 to pc_1 :

The point, here, is that this translation is a funct of α. ie. $g(\alpha)$.

T. functional form of $g(\alpha)$ is obtained from R / S, which is obtained by studies of pc v.s. cc for various searches.

We endup w. a goodness criterion for a particular code :

It is $pc \cdot (g(\alpha))^d$ (or $pc \cdot (g(\alpha))^{-d}$) :

where pc is t. pc. of t. code & d is t. no. of symbols coded (first some arby / indoor words — or from t. beginning of t. corpus).

so : $(pc)^{\frac{1}{d}} \cdot g(\alpha)$. Since all codes compared have t. same $g(\alpha)$

$pc^{\frac{1}{d}}$ looks like its t. Gorc !! : seems unreasonable!

Check t. reasoning carefully !

---

clearly, if $pc_1 g^{d_1} > pc_2 g^{d_2}$; then $pc_1 g^{d_1+d} > pc_2 g^{d_2+d}$ for arby d ($g > 0$). since $pc_1^{\frac{1}{d_1}} > pc_2^{\frac{1}{d_2}}$ is also true then $pc_1^{\frac{1}{d_1+d}} > pc_2^{\frac{1}{d_2+d}}$ must be also true for arby d : which seems wrong. If t. $\geq$! $\boxed{pc \cdot g^d}$ is t. Gorc: ($g \equiv g(\alpha)$)

This is not t. same as $pc^{\frac{1}{d}} g$ being t. Gorc.

taking logs ! $-b\cos t + (\log g) \cdot d = Gorc$.

$$\ell - K \cdot b = Gorc$$

ℓ = length of corpus

$\log_2 pc + d \cdot \log_2 g = Gorc$.

$-b + d \log_2 g = Gorc$

$d - \frac{1}{\log_2 g} \cdot b = Gorc$.

$\frac{1}{k} = \log_2 g$.

The value of g is computed from $f(\alpha)$; ($\alpha \equiv \frac{cc}{\ell}$) (47.20)

$$g(\alpha) = \frac{f(\alpha)}{\epsilon^{\alpha \frac{f'(\alpha)}{f(\alpha)}}} \quad (49.33) \qquad \epsilon \equiv \frac{f'(\alpha)}{f(\alpha)} \quad (49.28),$$

The value of α to be used depends on how much cc one has available : how long (ℓ) t. / entire corpus is

The form of $f(\alpha)$ is known in general way : It starts at 0,0 then approaches a horizontal / asymptote that is < 1. It will be ≈ 1 for NRTM., be < 1 for NRTM

we have to know only $f(\alpha)$ and $f'(\alpha)$ for the values of α we are using. We may be able to obtain continually revised estimates for these probabilities as we work on the corpus & get more info on pc ⇒ as a function of cc.

Ech checked over the derivation of $g(\alpha)$:

it seems o.k., but:

$$\left.\begin{aligned} \frac{pc_0 \cdot q_0}{pc_1 \cdot q_1} &= g(\alpha)^{d'} \cdot \frac{pc_0}{pc_1} \\ g(\alpha) &< f(\alpha). \end{aligned}\right\}$$

$q_0 pc_0$ = expected pc. of corpus using c_0 to start,

$q_1 pc_1$ " " " " " " c_1 " " .

$$\frac{q_0 pc_0}{q_1 pc_1} = (g(\alpha))^{d'} \cdot \frac{pc_0}{pc_1}$$

Now $g(\alpha) = \dfrac{f(\alpha)}{e^{\alpha \frac{f'(\alpha)}{f(\alpha)}}}$.

$f'(\alpha) > 0$.
$f(\alpha) > 0$, $\alpha > 0$
so $\alpha \frac{f'(\alpha)}{f(\alpha)} > 0$.
$\therefore e^{\alpha \frac{f'(\alpha)}{f(\alpha)}} > 1$.
$\therefore g(\alpha) < f(\alpha)$.

so $q_0 pc_0 = \left[\dfrac{(g(\alpha))^{d'}}{} \cdot \dfrac{pc_0}{pc_1}\right] \cdot q_1 pc_1$.

$< \left[(f(\alpha))^{d'} \dfrac{pc_0}{pc_1}\right] \cdot q_1 pc_1$

$f(\alpha)^{d'}$ is the pc factor that would be involved to cover the distance from c_0 to c_1 at average $\frac{cc}{k}$. However, if this were done, we would have less cc (i.e. less α & hence less $f(\alpha)$ i.e. less $(pc)^{\frac{1}{d}}$) for the rest of the corpus! so we must end up ⟨.

$\dfrac{q_0 pc_0}{q_1 pc_1} < (f(\alpha))^{d'}$ so the factor is $(g(\alpha))^{d'}$ w. $g(\alpha) < f(\alpha)$

.30 Another Q is: is $f(\alpha)$ a "good" function? i.e. is it indep of things we want it to be indep of! E.g. after a good coding seq. will it be the same (for conditional pc's) as after a not-so-good coding seq.?.

.37 Also, $g(\alpha)$ may vary during the course of the T.S. — even tho α is constant. — say due to hard-to problems being encountered, or just different kinds of probs.

→ 51±.01

4.2.82 TS

5 1/2

other than the difficulty of §1.30! finding $\rho(\alpha)$ can be a big problem! Studies of past history can give very ruff estimates. Then, ~~use~~ making decisions on the basis of assumed value of $\rho(\alpha)$ should, if I use wrong value, degrade system performance. How can I detect this? — can I use it to servo $\rho(\alpha)$ to a better value?

Also, $\rho(\alpha)$ may vary during the course of the TS. (§7.37) I'd like to have $\rho(\alpha)$ "updated" .... how much far back should I go in the corpus to get data to compute $\rho(\alpha)$?

Also, just how critical is knowledge of $\rho(\alpha)$ in controlling the search.?

4.282 T.S

A
B
C

Some Gaul. discn: Source mgmt. ideas!

1) That in certain kinds of T.S's, Least gives a highly overinflated idea of t. cc of a search. (39.18) In certain other kinds of T.S's t. approxn. is 39.01 — .21.

2) I'm really not sure as to what t. present over-all search sequences
.10 system is. One possl. version: One has t. entire T.S. available from t. start. One starts an L-search at t. beginning, & traces out all codes in Σ Lc order, up to a certain cc expenditure, (≡A). Then, using t. "best" initial seq. (using t. $\ell - k \cdot b$ score of 39.28) one continues it by looping to .

The size of "A" can be determined in this way! As one 4.382 finds codes in Σ Lc order, one obtains a certain $\frac{cc}{\ell}$ for work thus far. As t. cc in this last chunk
.20 of t. search ↑, I think $\frac{cc}{\ell}$ ↑. (see .252). When it gets as low as α (which is t. mean $\frac{cc}{\ell}$ that one expects to use for t. corpus), one finds t. "best" code thus far & loops to .

.10 ff certainly looks like a Gaul. srch algm.

.24 SN T. reason (In line .20) I think $\frac{cc}{\ell}$ must ↑ during a "batch", is that if it didn't, I'd continue in that batch! Particularly if $\frac{cc}{\ell}$ stayed < α!

How to compute the $\frac{cc}{\ell}$ of a "batch"! As one finishes each code in Lc order as far as it will go, one writes down its $\ell - k \cdot b$. As a result, one knows at each time, what t. "best" code of t. batch is.
.30 The $\frac{cc}{\ell}$ of t. batch is always "t. total cc spent on t. batch thus far, divided by t. ℓ of t. "best" code of t. batch < "best" being curt. gov $\ell - k \cdot b$ >".
.32

SN One dirty! The $\frac{cc}{\ell}$ of a batch may start out (after only a little cc has been expended) w. a value > α.
— So perhaps one must spend a min. amt. of cc on a batch before beginning to use t. stop rule implied by .30 — .32.

Another concern: As one picks the best code in a batch & starts to continue it for the next batch! I don't see just how one might sometimes back track to a different code of the previous batch.

One way might be if the present batch thus far did nothing but worsen the $\ell - k \cdot b$ score of the code that is the continuation of. When it had worsened it so it was worse than the 2nd best code of the previous batch, we then start a new batch (or a continuation trial() on that 2nd best code.

Can a continuation of the $\ell - k \cdot b$ score? **Yes**: if $\Delta\ell < k \cdot \Delta b$ i.e. $\frac{\Delta\ell}{\Delta b} < k$

.15 (Δ's part. the new continue)

This gives a new significance to the constant, k. → 55.01

It means that for the entire corpus, we expect $\frac{\ell_{corpus}}{b_{corpus}} < k$.

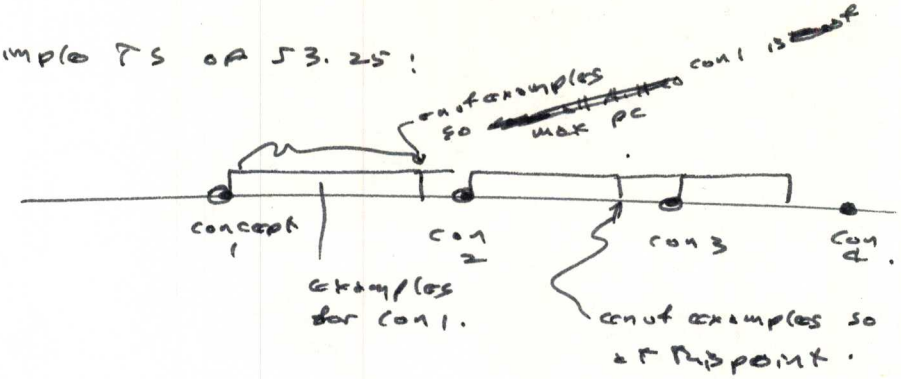
It would seem that this is not so reasonable. Say we have a MTM corpus. If all of the proper concepts are devised by TM, the best of the corpus = Σ best of each of the concepts. On the other hand, ℓ, the length of the corpus, depends on how many examples we give on the average, for each concept for be learned. So, viewed this way $\frac{\ell}{b}$ for the whole corpus is a fixed constant, independent of α, ($\alpha = \frac{cc}{\ell}$ = work per length of corpus).

.25 The ideal, simple, corpus, consists of a seq. of concepts: each followed by enough examples so that there is much likelyhood of the correct concept being the one w. max pc. for those examples (enough examples to elim. A.M. concept solns.). Presumably I'll be able to devise a search algm. that can work all TM's of that kind.

.35 A more diffl. T.S.! The new concepts come later once have enough examples in the old concept to elim. all A.M. solns. This means that one must carry along a lot of A.M. solns. that will take a bit of time to elim.

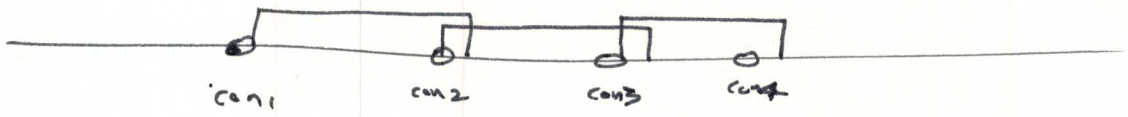
T. simple TS of 53.25 :

.05



T. more diff. T.S. of 53.35 : same as above but concepts are ~~close~~ spaced closer :

.15



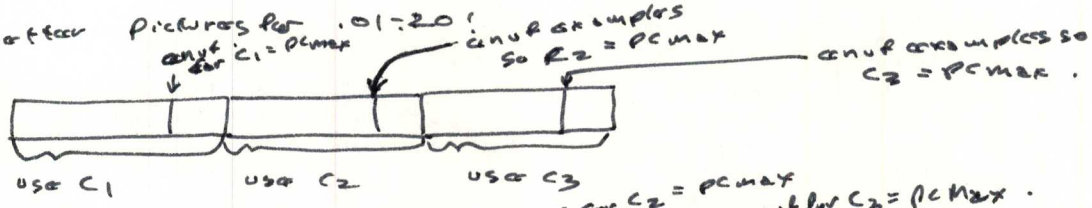
We have to wait untill after con 2 starts, before con1 becomes of max pc. For a while after con2 starts, we have probs using both con 1 & con 2.

.20

Nvr., for f. time being, concentrate on f. ~~[illegible]~~ diffys of 53.25.

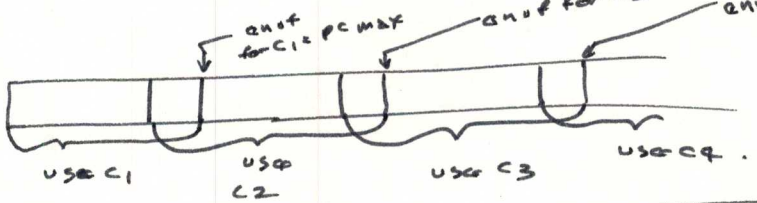
Better pictures for .01-.20 :

.25 like .05 → examples



like .15

.30



16×16 ×8 = 80

For 2 T.S. of type 53.25, (≡ 54.25) w. no "overlap" of concepts, its possl. to solve f. T.S. w. a [illegible] relatively simple Algm: In fact, I had such an algm earlier; see 39.35 – 40.08.

See just how well it works; also: ~~will have to~~ can't be modified to work w. pairs of TS's like 53.35 ≷ (≡ .30 ≡ .15)

15 × 14 / 39

.01 : 53.15 Further interpretation of 50.28 ($\ell - k \cdot b$ = gore) :

$\ell - k \cdot b$ is the amount one is "ahead of average" : i.e. On the average one has $1/k$ bits (D) bits per bcost/per bit of corpus (ℓ)

So: after ℓ bits of corpus one has, on the average, $\frac{\ell}{k}$ bits of code & bcost.

$\frac{\ell}{k} - b$ is the amount one is ahead / in bcost $\ell - kb$ is simply a constant k, times this. Its not clear as to whether $\frac{\ell}{n} - b$ or $\ell - kb$ should be used as the Gore. So far in my work on this, there is no difference between the 2 Gores.

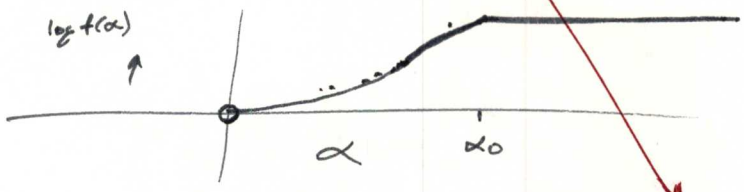
.12 Q: Is the $1/k$ giving the average (b) bcost/per bit of corpus (ℓ)

.13 nearly the same as the $\frac{1}{k}$ obtained from $\frac{1}{k} = +\log_2 p(\alpha)$ (50.30, 49.33) $= +\log_2 f(\alpha) \propto \frac{f'(\alpha)}{f(\alpha)} \cdot \log_2 e$?
50.28 R

$\alpha = \frac{cc}{\ell}$; $p c^{\frac{1}{k}} = f\left(\frac{cc}{\ell}\right)$; $\log_2 f(\alpha)$ = mean bcost (b) per bit of corpus (ℓ) for using cc per ℓ of corpus of α.
= average

I expect $\log_2 f(\alpha)$ to come to a max & stay there after an adequate α has been used! Not an asymptotic approach for the corpus of type 53.25 ←

& probably for type 53.35 also. ←
Note that these are both corpi w. finite solns. within the CB's being used.



.25
.26

I think the answ. to .12 is No! Thus $\frac{1}{k}$ for average bcost per bit of corpus is $\frac{1}{k} = \log_2 f(\alpha)$; The additional term $-\alpha \frac{f'(\alpha)}{f(\alpha)} \cdot \log_2 e$ (of .13) is important.

So: the $\frac{1}{k}$ of 5.28 is significantly smaller than the average bc per corpus bit.

The assumptions about the corpus that resulted in the gore $\frac{\ell}{k} - b$ seem to be much different from the models of 53.25 & 53.35.

However. Note that if α is large enough (.25), (> α_0), which we want to be true in 53.25 - .35 !
we get on the flat part of the $f(\alpha)$ curve, so $f'(\alpha) = 0$ and so, in .13 the 2 k values are the same. In fact, I think however for

53.25 i .35, we want α to be [redacted] α_0 (55.26) so we are sure to "capture" the concepts needed.

4.4.82

Consider a type 53.25 [redacted] T.S., in which each new concept is followed by one or examples of it, so that it's likely that only it, will be a concept of max pc — way out ahead of the others — after all those examples are done. (This kind of T.S. (if it is worked by TM.) enables TM to go along storing fewer concepts in Mem.

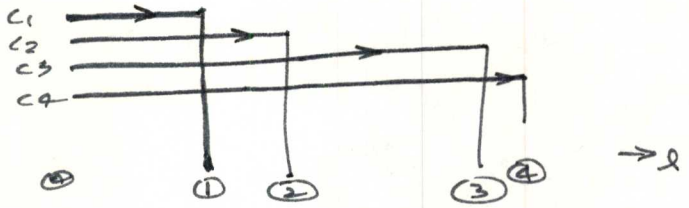
So, TM starts out the T.S. at the beginning, (but say he has the entire seq. of probs (in order) available to him if he wants them). He then tries concepts in LC order, testing them out as far as they will go, or until LC threshold is exceeded. This is done until the $cB = A$ is reached. At this point, (the T.S. is constructed w. "A" in mind) the first concept has been acquired, (since its LC is $< A$) and so TM is able to work out some distance in the T.S., until that concept is no longer adequate (Just before that to problem α_1) pt., there were only a few or maybe 1 concept(s) that fit — so there wasn't very much to carry in Mem.

As soon as the failure at α_1 occurs, TM can either go back to α_0 & expand more cc looking for a soln., or start looking for conditional modifns. of the initial soln. at α_1.

If we like, we could pursue both courses of action simultaneously. Here, we would have to make trials at α_0 for $cB > A$ only, since we already have all solns. for $cB < A$. At α_1, of course, we [redacted] could start w. small cB's & work up. As soon as (and if) we find a soln at α_0 that is either ① of equally high pc. or ② a soln. that gets us past α_1, we modify our search [redacted] by ① changing condl. probs at α_1 or ② moving the after work point from α_1, further out to (the end of success of the newly found α_0 concepts as far as the newly found α_0 concepts work).

Def Hyp. it is characteristic of the "Type 53.25 T.S." (by definition) that by the time we get to α_1 (from α_0) the only remaining concepts that fit within it that have any thing near max pc, will be concepts that are "essentially good"... that we want to return for modifn. at α_1.

More generally, if t. T.S. is "not exactly" of type 53.25, we may end up w. several concepts of hyish pc that work out to diferent distances. I guess what we would do then is pick t. concept of max $\ell - k \cdot b$. If t. concept we find this way is not t. concept of largest ℓ, then we will have at least 1 concept that is longer than t. chosen one. By starting conditional concept trials at this point, we will have > 1 concept that works out to that point, & we must Σ their pc's, & consider continuations of each, in view of t. pc of each up to that pt.



Say we have 4 concepts starting at ⓪; C_i works out to ⓘ.

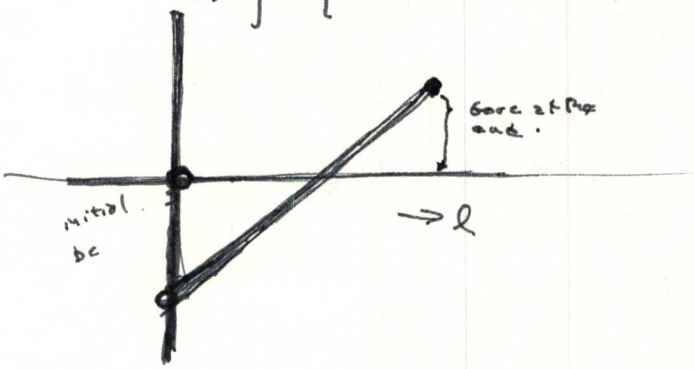
If we decide to start conditional trials at ⓘ then the pc of interest will be that due to t. sum of 5-i diferent concepts.

the $\log_2$ of the total pc at each of those points should enter into the $\ell - k \cdot b$ Gorc;

— In general, use of this Gorc may result in our deciding on an ℓ point that is not t. terminal point of any particular concept (on 2nd thot, maybe only concept terminal pts. are poss (..... remember that each concept increases b cost only at points ⓪, when it is defined).

(No! see 2)

So, for a particular concept, $\ell - k \cdot b$ as a funct of ℓ looks like:



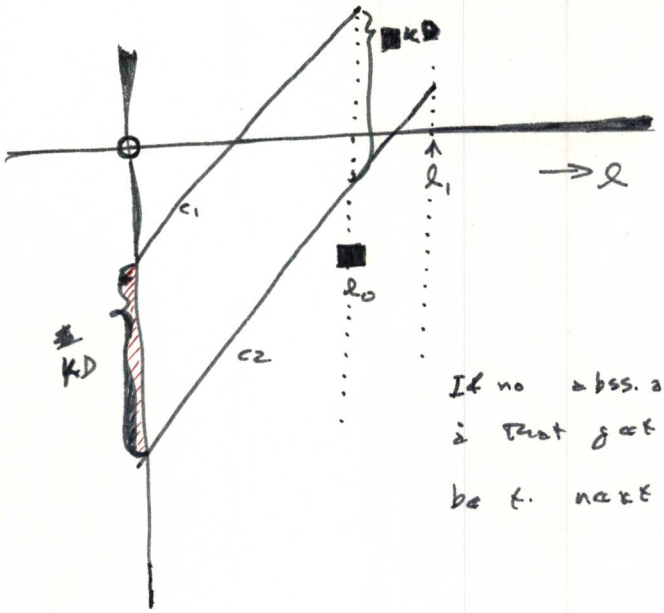
It always has slope +1, & its initial b cost determines its intercepts on x & y axis.

$\ln(p_1+p_2)$ v.s. $\ln p_1 + \ln p_2 = \ln(p_1 \cdot p_2)$

My impression is that if one sums Bc's of 11 concepts, that even so, the Gorc is $\approx$ max at t. end pt. of t. singular concept whose individual Gorc is max. — Unless a bunch of them end at t. same or about t. same point.

If t. Max gorc is not t. abs. of max ℓ, Then: If we start searching for conditional continuns. at ℓ_0, t. concept C_2 will be equivt. to one created at ℓ_0, but w. a bc of just D ($D = b_2 - b_1$ exactly)

$\ell - k \cdot b$



If no abss. are found at ℓ_0 which have bc $< D (= b_2 - b_1)$ & that get $\mathcal{B}$ at least as far as ℓ_1, Then ℓ_1 will be t. next pt. of conditional continuation trials.

If $k \approx$ t. average value of $\frac{\ell}{b}$, Then we should chose $k \rightarrow$

The $\ell_i - k b_i$ of all trials (ℓ_i & b_i being (send pt. / terminus) values) averages to zero.

These 2 criteria may be difficult: & I don't know which I really want.

.20 if $\Sigma(\ell_i - k b_i) = 0$ Then: $\frac{\Sigma \ell_i}{\Sigma b_i} = k$.

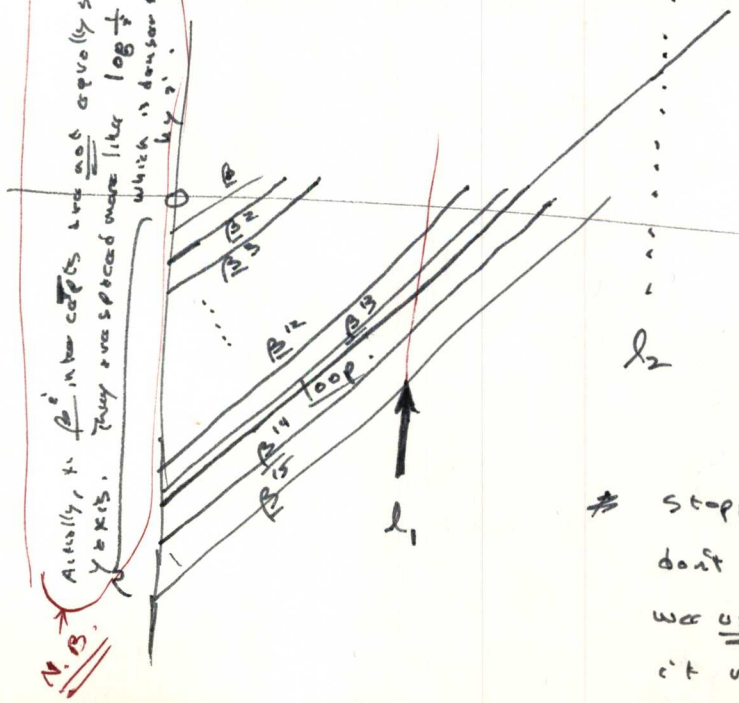
This is $\neq \frac{1}{n} \sum_{i=1}^{n} \frac{\ell_i}{b_i}$.

This looks better: also it yields same result for k & $\frac{1}{k}$ being t. constant.

So, we could start at t. beginning of t. corpus, do Lerch for cc = A, (or CB = A).

Then adjust k so that Then chose t. abs. w. $\ell_i - k b_i = \max$.

.25 What occurs w. t. β^{12} KS/loop problem:



If our CB limit caused us to stop at $\ell = \ell_1$, t. loop soln. would not get much wt. (personally, I'd like it to run to β^{15}).

I guess t. idea of 53.25-type TS's is that A is large enough for us to go out to $\ell = \ell_2$, so t. loop really does come out ahead.

* Stopping at ℓ_1 is wrong anyway, we normally don't stop at a particular ℓ point, but we usually trace out each abss. until it no longer works.

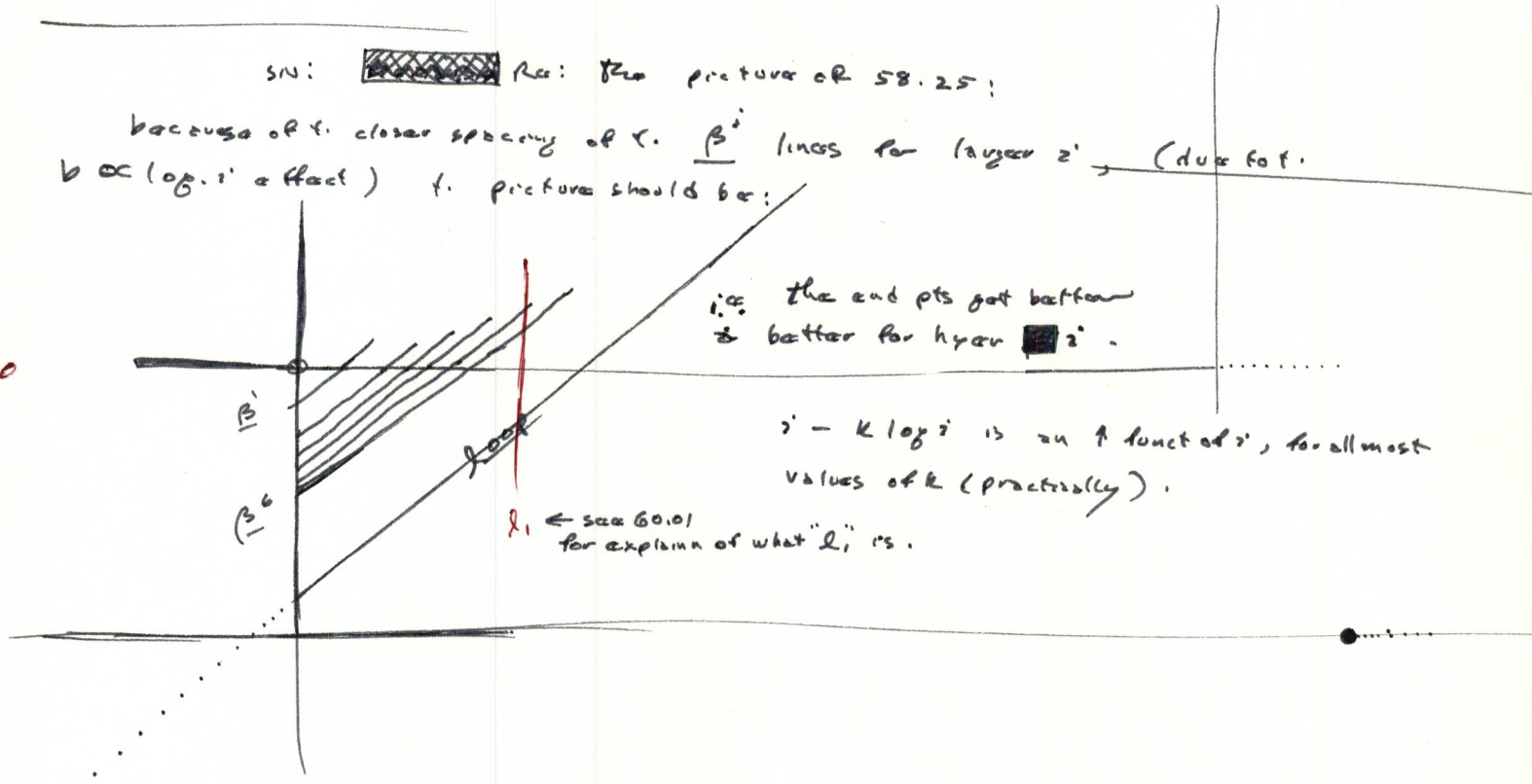
One Q about 58.20: do we want to give all trials = wt. in $k = \frac{\Sigma \ell_i}{\Sigma b_i}$?

Actually, it's my impression that we want to adjust k so that $\ell - b \cdot k$ averages to zero but for a population of trials that had $\ell - b \cdot k = \max$ ... i.e. the average is over/ many/ bases at .. the "stopping points," rather than averaging all values at a gen. stopping pt.

The value of k is $\approx$ the mean $\frac{\ell}{b}$ obtained over a long series of "stopping pts": This was, I think, the reasoning on which the $\ell - b \cdot k$ power was obtained.

SN: Rec: The picture at 58.25:

because of the closer spacing of the β^i lines for larger i (due to the $b \propto (\log i$ effect) the picture should be:



.20

If the T.S. is a real "53.25" so A is adequate, & the initial k value is not too bad, (presumably, k is adjusted as TM goes more data to make better $\frac{\Sigma \ell_i}{\Sigma b_i}$ estimate; — Move "stopping pts") Then probably the presently contemplated search algm. will work: i.e. start, search for $CB = A$, carrying each abs. out into the future as far as success or CB allow (whichever stops it first). Then use $\ell - K \cdot b$ to select the abs. to continue on as a "stopping pt." for "conditional abss": & conditional bc's.

.01 Next, what about 53.35, in which t. new abs. types (≡ concepts) come significantly closer
.02 together than in 53.25? We run into t. red line, l_1, of 59.20; The new concept introduced at l_1 causes all those abss. to "terminate". So we end up with a lot of shorter l trials, that are all consis up to t. pt., l_1.

We retain [redacted] in many, all of t. abss. that were consis up to l_1. — Then we then do a set of trials for conditional continn. at l_1. If t. abss. all end at l_1, their differential Goves are all dependent on their mutual bc's only. There is a standard method of search for continuations of a set of codes w. different pc's. We start out by continuing (and/or modifying) t. abs. of max pc. — we do this until t. bc's of t. modifns bring t. continn. below that of other "discontinued as yet" abss — at which pt. we make continn. trials on those abss. We do our trials in pc order for each LC level. Then when we exhaust that LC level, we double t. bc level & start over in pc order.

After we run out of cc (CB=A), we stop & look at — t. l_i's & b_i's of our continns. [redacted] if it is a true 53.35 [redacted] T.S., we will end up w. abrupt termination of several abss., so we loop to .02 ↗.

If [redacted] the next concept comes further out, like in a 53.25 TS, Then we tend to have a single abs. w. max Gove., & usually w. max l. So it may work o.k. in both t. 53.25 & 53.35 - type TS's.

Note that both are MTM type setups, in which t. CB's are large enuf to discr. t. relevant concepts. I think this is t. same as saying we are on t. flat portion of curves of 55.25-26. So $f'(x)=0$ & $\frac{1}{k} = \log_2 f(x)$.

[redacted]

So it may be that t. present search algms. are enuf for MTM-type setups if t. CB. is large enuf for all concepts needed. (Ref. Note possl. objection 69.30-40)

I should write up this "MTM" soln." in some detail, for a review art., & so I can criticize it better.

4·5·82 TS

.01 Perhaps the most impt idea is (the model of 53.25 is of 53.35), in which we characterize the T.S. as having a seq. of new concepts that are sequentially introduced, w. sequences of problems following each concept.

The idea is this: That we start the T.S. by doing an L search w. $CB = A$ for an operator that will work the T.S. as far into the future as possl. We try all operators in pc. order & stop the trial when they fail or when the trial exceeds the $CB = A$. When we finish, we have used a set of operators of different bc's (b_i) & that have worked different nos. of probs $(\equiv l_i)$. We have this set of pairs, (b_i, l_i).

Malcolm Samic
2 Millie Dr. Lex
861-1350
also 6921

SN There is some possy that we will want to test each operator as far as it will go (i.e. not fail). This might be true if the testing of the operators uses little cc. composed so the construction of the operator.

We then order them via the Gov $l_i - k \cdot b_i$. [k is a constant that we estimate. If $[\bar{l}_j, \bar{b}_j]$ is the set of pairs that have max Gov (that is, i.e. if (l_i, b_i) are the set produced by a set of trials & testing of the same problem in the T.S. then, let $\bar{i}$ be the i for which $\max (l_i - k b_i)$ is max, then & only then will $l_{\bar{i}}, b_{\bar{i}}$ will be a member of the set $[\bar{l}_j, b_j]$), then $\sum (l_i - k \cdot b_i) \approx 0$. so $k \approx \frac{\sum \bar{l}_i}{\sum \bar{b}_i}$]

We pick $\bar{i} \Rightarrow l_{\bar{i}} - k \cdot b_{\bar{i}}$ is max over $[l_i, b_i]$.

We then use the point $l_{\bar{i}}$ as a new starting pt. from which to make modifns in the operator(s) that have worked up to $l_{\bar{i}}$. Sometimes there will be only 1 op. that has worked as far as $l_{\bar{i}}$, but often there will be > 1.

If the T.S. is of type 53.25; then there will be a large enuf no. of examples of each concept, so that there is only one amount concept that is consis w. all the examples by the time the next concept comes up.

In type 53.35 T.S., there are fewer examples per concept so that when a new concept comes along, there are several previous concepts that are simultaneously "delayed". Most of those are A.H. & would be delayed anyway, if the T.S. continued when & introduced this new concept later.

4.5.82 TS.

53.35-Type TS's can take a lot more cc to do than type 53.35, if we use the algm of 61.01—.40! Say we get a new concept at a certain point, & there are 4 abss. that have been consid up to that pt. — (at which pt. they become inconsis:). One of the abss is "True"; the other 3 are A.H.'s would be elimd. by more examples of this concept (but, so if it were a 53.25 TS. If the "True" abs has a pc. that is a factor $\alpha <$ the A.H. abs. of max pc., then I think we will need a c.B. a factor α greater than that is needed if the "True" concept were the only one (like in a 53.25 T.S.), to discr. the next concept. This is because on trials modifns. of the abs. of highest pc. first; then "True" concept, starts out w. a penalty factor of α when modifn. trials are made on it.

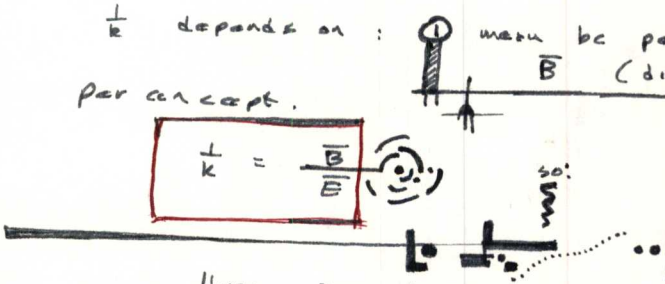
On the other hand, the A.H. abss. that last as long as the "True" one does couldn't be "all that bad" & probly have some useful mass creating trials for the next concept, so maybe we don't lose as much as a factor of α.

Routz
#3K/yr.

On the value of "k": $\ell - kb$; $k \approx \frac{\ell}{b}$; $k = \frac{b}{\ell}$.

$\frac{1}{k}$ depends on: ① mean bc per concept used $\bar{B}$ (dims.: "b"). ② mean no. of examples per concept $\bar{E}$ (dims.: "ℓ")

$\frac{1}{k} = \frac{\bar{B}}{\bar{E}}$ so: $\frac{1}{k}$ is larger for 53.35 than for 53.25 because $\bar{E}$ is smaller for 53.35. $\bar{B}$ is the same in both cases.



However: In 53.25 & 53.35 ① The arguments about $f(\alpha)$ & saving cc, etc. are irrelevant if we are in the flat region of $f(\alpha)$ (i.e. $f'(\alpha)=0$).

32 ② In these types of TS., I think the trial w. max ℓ value is Always the Best to use; i.e. it does mark the point at which the next concept is introduced.

In 53.25, .35 we want to know when the new concept occurs: that's all! So, the only possy of error would be if there were an A.H. penetration

of the next concept, that didn't "really" solve it.....

Probably this wouldn't happend often. If it was infreq. enuf, we wouldn't could simply "backtrack" (Tho I don't see how that would help!).

In general, whatever we do, we will occasionally run into diffys where we get "out of phase" w.t./ concepts, or where we didn't introduction of.
really acquire a concept but were able to continue to some degree, past it
for some distance. In such cases, first, one must detect that these things did occure: Next one must decide how to backtrack. Third, one must see if, in general, the kind of backtracking that one did, could really help w.t. observed diffy.

.....

Well, Say one had a gbs. that did get far beyond other abss. ("hy'l"), but it had hy bc. (≡ low pc). One chooses this abs. to continue on, but if things don't work out, one moves back to a prev abss. that didn't have as hy a l, but had lower bc—— this is a kind of BackTracking.
So, when learning along a T.S., one must preserve in memy, the alternative branches of decisions that one made, where the "rd. not taken" was of pc not much < the "rd. taken" .... so one can readily backtrack.

So, in general, when one picks the abs. of largest l, one also retains in memy, abss. of lesser l, & perhaps hyer pc. Hvr., if they are of less l & less pc than we may not want to put them in memy! or we could put them in very inexpensive, slow access memy. (Many of this save would be, at first, a part of / fast RAM. When this section filled up, it would be put on disc, say.

Before going on to NMTM-type TS's, I want to work on MTM a lot. Perhaps try to make coins. of a form that are adoptable w. as not too much modifn. for NMTM. Some things that seem to

SN 53.35 is sort of like having a "teacher" in that new concepts are not introduced until it appears that the old ones have all been acquired.

→ make NMTM much harder (a distant from) than MTM are:

4.6.82 TS

on NMTM

.01 That first, one doesn't try for a "perfect fit", so to speak, but one settles for a certain level of $\frac{\Delta bc}{\Delta l} \neq 0$. Second, one is always running into problems & parts of problems where one's $\frac{\Delta bc}{\Delta l}$ is very low — very poor prediction, sometimes pure randomness (as far as one can tell).

Somehow, one allows the bad $\frac{\Delta bc}{\Delta l}$'s to accrue & one concentrates on other "areas" of the corpus in which one's $\frac{\Delta bc}{\Delta l}$ is "improvable".

The problem is: how to break up the corpus into parts & how much cc to spend on each part, & in what order. For many corpi, the order is somewhat indicated .... which is what the TS concept is all about.

Later, one may go back & try to work those probs that seemed at the time to have no solns. This "back tracking" could be done

.17 w. different condl. probys a/o w. larger ΔB.

A thing I may have overlooked in the MTM T.S.'s: Perhaps — TM can more along the T.S. & not work certain problems, but be able to recognize those probs. correctly & not give "wrong answers" but give no answers at all. Actually, for MTM type problems, the solns. usually are divided into 2 parts: The first is for recognizing what kind of problem is to be solved, the second is the algm for solving that particular type of problem. So, if we successfully solve the first part, even but not the 2nd, we ought to get some kind of credit!

[This can be dealt w. using the standard soln of 53.25 − .35 : [ie. 61.01 ff, but w. modifn. 62.32]. At a particular "stopping point", we have, for each trial,

① a Δb cost, & corresponding Δl, if we consider only perfect solns.

② Δb cost, for the recognition part only, & a Δl obtained, which tells how far we are allowed to either go if we commence to solve the prob. or recognize that we cannot.

Actually, perhaps we want to know the no. of probs actually solved in the corpus using the method of ②, & not give positive credit for "I can't's".

What this does, is break the set of probs into 2 parts, so one can work on them separately — certainly a great simplifn.

.35 ①.②, so one starts at some "stopping pt", runs the [such at $CB = A$ until its done. But instead of running the abs. trials out as far as they are cost, we run them out as far as they will go, — allowing them to select themselves the probs they will try to solve. We do have cut off, if they try to solve a prob. & fail. Also cut off if [i.e. get wrong answ.

.01: they [redacted] overspend "time"! This spending can be both in working problems correctly & in deciding to discard [redacted] problems as "unworkable by that abs."

.03 At this stopping pt., then, we have all these abss. that work on subsets of probs out to various distances. We want to note which distances were cut off because of failure (≡ error) & which were cut off because of running out of "time" (≡ CB). .06

Problems → **The method of combining &/or discarding some of these abss. & how to characterize the next "stopping pt." Just what is the prob. to be solved at the next "stopping pt."?**

Actually, the "stopping pt." is not localized in the corpus. After each "stopping pt." we have a set of (abss) objects, .03 – .06: Certain of them have never failed. Others have failed after (many / a few) successful trials. Given a new problem, one could apply all (applicable) abss. to it to obtain possible solns. A rough estimate: If an abs. has worked on n probs correctly before either failure or "time" running out, we give its soln. a prob. of ~ $1-\frac{1}{n}$. (maybe $(1-\frac{2}{n})$ if it failed at n) .18

We combine these probs by estimates to obtain approx./ relative probs of different solns. obtained by different abss. .20

Both the $1-\frac{1}{n}$'s & the pcs of use.

This changes of pc w. 552 is probly wrong — T.V. 94.16 — to .30

Usually they will all give the same soln. — but we will combine them this way when they don't. For abss. that have been always consistent, we may want to run them out further to see if they stay consistent — if these abss. [redacted] are impt. in modifying a probty estimate of a particular soln.

Anyway, at the next "stopping pt." we use the conditional probty of all of the abss. used thus far to obtain a new set of abss via Lsrch. This time, we try to fill in the probs. not worked by the abss. thus far. This is done by creating entirely new abss. & by modifying old abss. In particular, we want to modify abss. that have been consistent up to a certain pt., & then fail. .29

.30 64.35 – 65.29 looks like a very good improvement, expansion, genus. of 53.25, .35. I think a fairly reasonable T.S. could be worked both within the forms 53.25, .35; using 61.01 for w. modifier 62.32.

But 64.35 – 65.29 looks like a great improvement! The conditions on the structure of the TS are much relaxed. In addition we have automatic breaking down of the corpus into probs of different types — essentially / parallel, mixed sub-corpi — w. each sub-corpus being a separate problem, able to use' info., abss., & sub-abss from other sub-corpi solns! Also the Operator(soln) is broken into parts, so credit for successes & errors can be localized.

4·7·82 TS.....

(1) T. discn. of NMTM of 64.01-.17 seems to be more amenable to t. [redacted] New approach (for MTM) of 64.35-65.29: Very Good!

Hrr., it might be advisable to try writing & working a 53.25-.35 type T.S. [redacted] first. It would seem much easier to get t. methodical reasoning rite for how "transfer learning" occurs; how/ successful use of an abs. & its pc, etc. In particular, I want [redacted] to get at least a simple TM working to some extent before going onto some thing more complex, & t. 53.25-.35 system seems [redacted] relatively simple to analyse, yet it seems capable of significant interesting learning. Its main deficiency seems to be that t. T.S. has to be very carefully written. Its main interest is in (1) Exploring t. constr. of TS's. (2) Getting t. math worked out for how t. pc's of [redacted] abss. are a function of their [redacted] successful & unsuccessful use & of t. pc's of t. subabss used to generate these abss.

T. constraints on 53.25-.35 : # of ann of ec (≡CB) for t. concepts [redacted] involved, & an ex of examples betw concepts; These [redacted] to some extent replace t. presence of a teacher who's observing t. student & figuring out whether t. T.S. to date has it → t. pupil needs more & larger C.B. a/o needs more examples. If too large a C.B. seems to be needed, t. teacher brings in [redacted] "easier" probs of smaller cjs (≡ CB needed to solve it). ↑ t. no. of examples of a concept can remove confusion in mind of student as to which concepts are t. very basic ones. (helps Get Rid of R.H. concepts in TMJ case)

Also, A human teacher can tell a student to spend more time on a particular set of probs.

4·8·82 When, on m t. new kind of T.S., an abs. trial is terminated because it failed, then at t. next "stopping pt.", we try to modify that abs. so it will be consi for a larger no. of probs. If t. abs is terminated because of CB, then we merely continue that abs. for t. next "stopp. pt."

7.8.82 TS

The new kind of T.S. & the way I expect to solve it seem much closer to the way humans deal w. these problems than 53.25–.35 & its [redacted] proposd. soln.

Also, this business of having various kinds of problems & various methods for

.04 solving each kind, lends itself to that v.g. "Plan" that I worked on some time ago, in which we try to solve a problem by xfmg it into a class of probs. we [redacted] already know how to solve.

80 TS 73.01

For a Bibliogy on Plans: see TS 8.11.81 ~~...~~ n p140 ; w/ line .20.

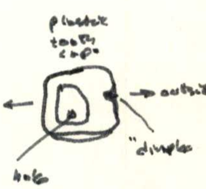
This "xfmn" plan specifically: 80 TS 73.20 for Bibliogy : 80TS 68.10–.40 ~~...~~ is a dscn. of the plan itself.

80TS 68.10–.40

Another possy. goodness about this latest approach is that it may genz. [redacted] easily to NMTM. [see 64.01–.17 for a discn. of some diffys in NMTM]

One thing that would seem natural, would be the working of a "large corpus" by means of

.17 a PEM. The way this mite be done: Simultly, one creates a "~~device~~ algm. that selects out part of the sub corpus (the part to which the pem is to be applied), & one creates a pem for that sub corpus. ~~...~~ Say this ~~pem~~ the particular pem_i has a $pc = p_i$. If the Lsearch CB = A, then we will allow $\Sigma cc = A \cdot p_i$ for the creation of the pem & for using it to trace out [redacted] as much of ~~...~~ its sub.corpus as possl. [redacted] The pem_i trial ends when either ① it runs out of cc ② it finishes evaluating its s.c. Actually, ② can't really occur, ordinarily. [redacted] Pem_i will simply spend the rest of its cc. [redacted] using the Algm. of .17 to try to find new parts of the corpus to which it can be applied. (I guess I'm assuming an infinite corpus ..... If the corpus is small enuf so that TM can actually examine all of it, then it could and its trial [redacted] when it had applied pem_i to all of the corpus to which [redacted] pem_i was applicable.)



I guess we could have, for NMTM, a mixture of both small & large subcorpus

def predictors [redacted] I.E. A predictor could define its s.c. (≡ sub corpus) & also decide [redacted] if a pem or small corpus treatment was to be used. Also, a single small s.c. algm. could divide the corpus into several s.c.'s, each of which is individually worked by the same s.c. algm.

In general, this idea of breaking up the world into parts that are to be treated in diffnt. ways, seems very good. We could also ((have MTM-type probs mixed w. large & small corpus NMTM probs.

One problem I don't remember having solved: How to get quantitative (say numerical) params for a Pam — like linear Regn. or Maxm. I'd like the linear regn. easy, to be discovered in a way similar to how a human might discover it. e.g. To predict the next element of a numeric time series by noting that up to now, the linear regn. equ. $x_n = 3x_{n-1} + 2x_{n-2} - 7x_{n-3}$ has worked not badly. Or, more basically: If $x_n = ax_{n-1} + bx_{n-2} + cx_{n-3}$ has worked well in the past, then try it for predn. Then, we ask, what's a good way to get a, b, c so that this equ. is as good as poss'l.

The discovery of lin. regn. can be made after TM has discovered other related predn. systems. e.g. In previous problems, if we found a hy p.c. function $f(\cdot) \Rightarrow x_n = f(x_{n-1}, x_{n-2}, x_{n-3})$ for most of the s.c., then we could use $f(\cdot)$ for predn. Then the prob. becomes: how do we find good forms of $f(\cdot)$ & how do we find good numeric params for them?

E.g. How could one lead a human into discovering linear Regn.? Well, maybe by starting off w. Time series in which it is easy to guess the exactly right coefs. Then series in which there are small integer coefs that give small m.s. error. Then " " " the coefs are harder to guess (more ec. needed). Hvr., if TM has experience solving simult. linear equs., he could be made to realize that he could obtain min m.s. error by solving simly. equs.

Hvr., the relation of small m.s. error" to "good predn" would have to first be decid., & I suspect that this is a very large cjs discovery — having taken many human man-yrs. of Z cost for human decvry of it. or conclusions!

In general, the various methods of time series predn., & linear regn. in particular probly did take much mankind much Lc to decv., so we shouldn't expect TM to decv. it very easy w.o. much specialized training segus or some kind of "wired in", a.H. concepts that make the decvry of these ideas more likely.

Sleep Marker

One way to do this: Consider all known methods of predicting numerical time series. Factor these methods into as few hype basic concepts as poss'l, then wire-in these concepts for TM in factored form (so TM can know the pe's of the sub-concepts).

[Actually, it may well be that there really aren't many ways — i.e. classical statistical ways & even very many in no. Hvr. newer "pattern recgn" ways are more numerous. "clumping"; "stock Grammars" (many kinds of Grammars); separation of clumps by linear & non-linear manifolds; nearest neighbor & variations;

One problem (again, I don't know if I ever solved it), was the division of methods into apriori codes of corpi & codes obtained (like Markov or linear Regn) after one has examined the corpus.

A (perhaps) new way to look at this sort of predn problem: TM is an. as input, the Turing series up to incl. X_n. The output is to be predicted. This is on the basis of the set of I/O pairs $(\underbrace{X_0, X_1 \cdots X_i}_{\equiv (X_i)}; X_{i+1})$, $[i := 1 \mid n-1]$. ie. $[(X_i), X_{i+1}]$ ← (corpus)

.17 one way: TM does (X_0) predicting X_1 as a small corpus predn. next, using the previous pc's as a starting pt., he does (X_1) predicting X_2 & etc .... eventually, predicting ~~X_n~~ ~~X_{n+1}~~ X_{n+1} from (X_n) using the params of $(X_{n-1}) \to X_n$ ~~for~~ for condl. probys.

What this means is that TM looks for optimum params (both by quantitative changes & by ↑ & ↓ no. of params) by modifying the optimized params of the previous prediction. This method has the disadvantage of the $\beta, \beta^2, \beta^3 \cdots \beta^{12}$ induction sequences in that it will not find e.g. global maxima. This is not a difficulty in linear regn. & in certain of its variations, but it can be a problem in fitting over several "local optima" in the param set.

[I'm not even sure about linear regn. — is there no local optimum in the number & /o nature of the coefs? — "local" relative to "Hamming distance" — i.e. the no. of params that have been changed, or the absolute ↑ or ↓ in the no. of params.

.29

.30 Actually, as I usually look at it, if I consider the finding of a good Parm to be the problem in "the large corpus" problems, then, in the case of linear regn., I haven't reduced the corpus size enuf to do a simple 2 such. The search space of sets of coefs is still too large for most problems. In the method of .17 – .29 the search space is of reasonable size often, but still I must limit the no. of codes stored, & must keep the no. of dimensions ("order" of the regn coefs) not very hy, since no. of possible modifications of the previous code is exponential in this (as is also the no. of codes that need be stored).

.01 A more reasonable approach is that even for large corpus problems once must cut down the search space usually, by searching for a prodn. method. I think this is what he means to do. A "prodn. method" would be like: "Use clumping" — then we look for good ways to clump.: "Use linear regn.", there we try diffnt orders of prodn. also diffnt. coif types; "Use N.L. regn." we try diffnt functional forms; "Use Makem", this involves no search (... if I develope it completely: there is some Q about how to start out the codes).

.10

So: In general; First try all s.c.'s as "small corpus" if it doesn't work well, then try to reduce the corpus size. This can be done in several ways: (1) By sequential breakdown: This is the technique used when we postulate the corpus is a T.S. and we treat it as 53.25 or .35 or by the more general (I guess) method of 64.35–65.29. (2) By searching over the space of prodn. methods. (like .01–.10).

So, the first step is to get a TM that can work small s.c.'s. Next, the problem of reduction of a larger s.c. to a small s.c., becomes itself a small s.c. problem.

So: first we get TM to work a sequence of small corpi, arranged in a "training seq." -like order. At this point TM cannot work anything but small # corpi — by doing simple Lsrch. The method of 64.35–65.29 is also a "small corpus" (such method, but TM doesn't know at first how big the corpus-to-be-worked is. Here is just enough CB for Lsrch, so he finds codes for as much of the corpus as he can, within that CB. Having done this, he moves on to the next (undoubtedly infinite) chunk of corpus & conditionally codes whatever he can in the (usually) same CB, and so on. 10:47P

To consolidate our ideas: Some examples of these 3 kinds of both 64.35–65.29 & Corpus situations: 53.25 & 53.35: The corpus consists of individual problems — it has been pre-cut up into these problems. In 53.25–.35 the example is the T.S. of learning alg. notation — starting w. unary & binary functs, then learning the operator "Eval" involving substitutions.

In 64.35 - 65.29 we could work t. same problems as we T.S. as was done for 53.25 . . . , but we need not be so careful about ordering example problems.

In t. "Grand operator" consists of a bunch of obs., that identify various problem types. Each ob is then tied up w. 1 or more op. which then xforms t. prob. into t. soln. Here t. input of each op. is a/ (total) problem descn. Its output is "yes or no" — i.e. its operator can or cannot work that prob. Or, we have obs w. > 1 output. The outputs are t. names of/ (1 or more) operators that can work t. input prob — or, t. output is Null — saying that we can't work it.

If we have many single obs, w. corresp. ops., then t. "grand operator" is t. "sum" of these.

In 53.25 . . . we have a single "Grand Operator" also, but we do not initially factor it into obs & ops. Also this "Grand operator" is supposed to work for all probs in t. corpus thus far. The point at which it stops working is our next "stopping pt." for modifn of that Grand operator. These modifns. can be global on t. Grand op.

In 64.35 . . . after a stopping point, we only try to modify those ob-op combins that failed, & we also try to devise entirely new ob-op combin's & forms. So our "error" (i.e. "error responsibility localization") detection is much more highly localized than in 53.25 — (v.g.!).

64.35 . . . is more tolerant of teacher errors than 53.25 . . . . If a problem is put into t. T.S. that is beyond TM's c.B., or often if there is simply an error in t./ prob. & / soln. e.g. in t. T.S., then 53.25 . . . will get stuck at that pt. & will not be able to continue. 64.35 will simply pass by that problem, & try to work it later, using t. same c.B. as before, but different/ cand. & perhaps, different sub-obs. available.

Anyway, this "getting stuck" property of 53.25 . . . is useful in that it tells t. teacher that there is something wrong w. t. T.S. We can, anyway, get t. 64.35 system (or anything to work TS's devised for 53.25, & make (t. 64.35 system tell us when it has been unable to work a prob. in t. T.S. after it occurred in > 1 stopping pts" — so perhaps 64.35 . . . is "uniformly better" than 53.25 .

Other than th "Getting stuck" — having to work all probs int. T.S. as they come along, — T. big difrncer betw 53.25 & 64.35 is that / Y. Grand Operator of 64.35 is a v.g. elementalization of t. Grnd.op. of 53.25! [i.e. t. operator is divided into an "ob" part that characterizes each prob. & the rest / bunch of operators. t. ob part assigns t. problem to one or more operators.

(~~One~~ response may want >1 soln. to a problem: ① It may be that later, one method will work where others fail ② T. methods may have diff'nt pc's for diffnt probs — so we may be able to ~~[redacted]~~ estimate / guess this apri & assign t. fastest operator to each prob. ③ each op. may have diffnt sub-abss that we want to give credit to to & apply their conditional pc's.

Since this aspect of 64.35 is an elem. of 53.25, we will want, eventually, to allow t. ~~[redacted]~~ less el. ~~[redacted]~~ view ... say we should integrate. some of 64.35's obs., integrates so ops., perhaps make a more genl. relation betw. t. obs & ops.

T. least el. view of 64.35 is a Grand operator that looks at each problem & either proposes a soln. or not, & goes on to t. next problem. After a "~~mini~~ run" (betw "stopping pts"), it should have some means of localizing t. cause of any errors it made, also modify t. condl. pc's of various sub abss that were used.

N.B. When in t. Normal 64.25, we modify an operator because it fails, we have to try t. new op. on all of t. probs that t. old op. succeeded on, before we are sure we have a "uniformly better" replacement.

An alternative, would be to try to find some ob. to determine where t. old op would fail — so t. new op ~~would do~~ would have to deal only w. a subset of cases in which t. old op was inapplicable

sleep Marker.

4982 Hm., its not necy that t. new op work in all cases where t. old op. would — ~~what~~ what we want is a new ob-op pair & we combine it w. t. old ob-op pair to create a new, better, pair. → 73.18

.02 .35

Defs

sleep Marker SM

4.11.82 One of t. things that 64.35 ff... (& perhaps 53.25.?) do getus involved in, ~~was~~ is t. automatic search for Obs. that categorize & row ca. — That rejecting a problem very quickly (as being "not fitting" a partic. ob) is

a very useful [redacted] property of that ob., & should be selected for in a good system

.02 Also, t. partial Globalizn (≡ non-localn) of t. ob. concept to obs which produce [redacted] selection decisions (over a larger field of selections) rather than simply yes-no decisions on whether an object

.05 "observed" has [redacted] a particular property or not.

In particular, t. "plan" referred to in ~67.04 would be v.g. for this kind of T.S., because it would involve manipulation of obs & attempts to find global/obs like .02-.05 to categorize things more rapidly. It would involve attempts to xfm probs of [redacted] arby types into probs of types known to be solvable. Also we try to find sub-obs that help tell what characteristics t. soln. might have.

Dental decisions:
1) what to do about Bridges
2) Wisdom Teeth.
3) Overbite.

.18: A thing that has been bothering me: that when an abss. fails, & a new obs is formed I don't think people [redacted] naturally ordinarily try it on all of t. probs. in which t. old abs. worked. They might try it on a representative few, or somehow avoid that enormous checking problem. Perhaps they do a little checking on t. past then if it looks good, try it on some new problems. If it still works, do other tests on it — but not nearly simply working more probs of t. past. ... Perhaps we evolve methods that can tell quickly whether t. new abs. will work in all cases in which t. old one worked. → 79.01

.26

$\left(\frac{5}{110}\right)^2 \times 250 = .43w.$

1 in some corner, 10 v some zone

$\frac{1}{20}$ A 6V

$\frac{2}{10} = .2w.$

.27 SN SM An olde problem: When humans are trying to find a concept that will fit some data, I think there are conds. under which, when they do finally find one that fits, they assign a very large [redacted] pc to it. — larger than would be justifiable on t. basis of t. usual analysis. Well, one possl. justifn.: In certain kinds of [redacted] situations, like puzzles or problems specially devised by teachers, there is, indeed, only 1 hy pc soln., & [redacted] no alt. solns. of comparable pc.

.·1. (73.26) → One approach to this: Say we have concept α with a history of 100 consi cases, that it has just failed. We devise concept α'. [redacted] we test it on [redacted], say 5 probs past [redacted] (including) t. failure of α. [redacted] Its consi.: we try it on 20 randomly chosen probs from [redacted] t. past of α. — it still stays cons:. Well, we could just give α' an empirical [redacted] success no." of 5+20=25 w. no failures. Hvr., note that α' [redacted] probly has a much lyer aprip than α, because α' is only a slite modifn. of a fairly successful concept (i.e. α). — so t. apsi of α' will be >> that of α. This hy apsi may be one factor [redacted] that could give what looks like t. effect of 73.27.

9:48PM When we actually have to do prediction ... get probs values .... if we have to use obs. α', then we may want, at that time, to check out α' on more examples of t. past. On t. whole, this's mite save much cc — because otherwise, we may not ever use α' for predn. & it would be a big waste to spend so much cc. on it. → On t. other hand, we want t. pc of α' w. some accuracy (say within factor of $\frac{3}{2}$) to use it to [redacted] modify [redacted] pc's of [redacted] its sub abss & of abss that are modifns of α'.

Diffy w 64.35 → Going back to 64.35 ...; At that point, I was thinkg of t. Global op. as being a simple unordered collection of ops., Each w. its own ob. that told when it could be applied. I guess I was thinking there would be only a small set of these ... just as in 53.25 .... I only kept a small set of obs that had been "consi-up-to-now". Hvr., it may well be that there are really an enormous no of ops that are testable at any "stopping pt." — So what would be an optimum (or even acceptable) way to handle this?

Another thing: In t. original 64.35 ... I had that of each op having its own little "trigger" ob. ✱ More recently, it has seemed likely that t. ob part might be somewhat global ... it could look at any pblem & decide which, if any, op was applicable. This is more general that just having a bunch of individual, indep. obs. Hvr., t. assignment of responsibility for success a/o failure becomes less clear than it

t. obs were all/independently working. Note 72.02 that t. Grand op of 64.35 is an elem. of t. Grand op of 53.25 .....

.03 4·12·82 A search method that is "between" 53.25 & 64.35 !

T. Grand Op / O_3, has, for input: any problem: Its output is either ① an attempted soln. to t. prob. or ② t. statement "[redacted] ∧" (≡ "I can't work that prob").

At any "stopping pt" we apply O_3 to successive problems that have not yet been worked (starting at t. easiest unworked prob). We continue on till [redacted] O_3 fails, or until CB is exceeded: This doesn't look so good. Superficially, its [non-e] of 64.35, but actually, it doesn't work well. It does not use L srch at all .....

.17 An alternate form of 64.35 that's in t. direction of 53.35 !

O_2 (t. Grand op. of [redacted] 64.35) is this pool of little ops w. their respective obs. As we come to an as-yet-unsolved prob. in t. T.S., we use O_2 to try to solve it. This means all of t. little ops of O_2 are applied to it. — (& of course, some say "∧"). How L srch is to be used is unclear. (— Say each "little op" has its own pc. (T. but meanings of such [redacted] pc's is unclear!))

I can probly get a meaning for t. pc's out. basis of pc of coding t. entire corpus. Right now. The pc's of all possl. operators sum to 1 .... They are obtained out. basis of pc's of sub. abss. used in constructing the ops.

p25 T. pc. of t. Grand op. is t. product of t. pc's of t. little ops & is v. very small! → Hvr. its reciprocal pc can be acceptable ... (?) → 76.19

Hvr., it would seem that t. total cc of trying out all little ops for a fairly large no of cases, would be about t. same for 64.35 & 75.17 ↓ 2nd !

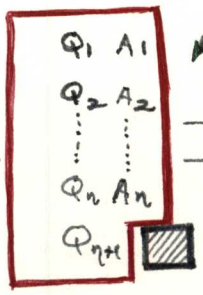
[redacted] Maybe not: in 64.35, we spend a $\max_{\lambda}$ cc of $\frac{A \cdot pc_i}{[redacted]}$ on t. [redacted] little op, in working probs w. it. In 75.17, I really don't know just how much cc to spend on each [redacted] "little op".

.30 T. thing is, in 64.35 I can get an upper bnd. on how much cc it takes to get a certain op. to do a certain bunch of probs, — regrdless all other ops working their probs as well: If it take cc = α to work t. seq. of probs, then we need $CB = A \sum \frac{\alpha}{pc_i}$ (pc_i = pc of t. i^{th} little op. in question). Remember that this estimate is for normed pc_i, so if we only have, say, 10 little ops., pc_i usually wouldn't be very small: Also, remember we get all t. other probs done for other ops, if they can slide under t. $CB = A \frac{\alpha}{pc_i}$ wise. → 76.10 spec

I think [redacted]'s way of analysing a Q.A. type TM is to try to get t. output to be of max pc. wrt t. input as given. This is conditional [redacted] pcost.

[redacted] So: my version of Chaitin's defn.

Hwr., superficially, it would seem that this special form of condl. pc. is not necy:



Here we want to get a probly distribn. for t. content of t. black box, on t. basis of t. content of t. Red Box.

optimizing compiler.

.10 75 40 → Well, t. analysis of 75.30 – .40 isn't so exact; Every operator's ob. has to operate on every problem. This may be very costly!

— Unless (as is likely) we devise some more economical Global ob. that decides which ops to use on each problem. Hwr., if we do take t. avgmts / sets of little obs i make x by ob. out of them (which is of much less cc than t. Σ of their cc's) than when I want to make trial changes in t. little ob-op pair, during these changes which it will be of more cc, than if I had/t. only little op-op pair to change.

Otto! No poetry recently

Set recital is doing lots of music.

Is writing optimizing compiler... using M.S. methods community sub-modules

.19 12:30 P ① [redacted] while t. pcost t. "end op" may be very small (75.25), its incremental pc (at this stopping place) may be quite reasonable.

② For a rather large set of probs in math, one always knows if one's answ. is right or wrong! E.G. solving equns., integration, solving some diff equns., some kinds of "proofs", N.P. probs, etc. For such problems the decision that "one can't solve them" may take much time! Hwr.

.25 → I think t. 64.25 method is closer to t. way humans work probs. Its deviation from human-style probs solving appear to [redacted] cause difficulty in theory & practicality. So perhaps by examining its devn. from human methodology I can fix it up, where needed & / or point out t. directions needed for future development.

4 13 82

5PM In line w. .25: First try to get 64.35 into as exact a form as poss'l.

[There are several unclear things: like ① What is actual cc of solns. relative to more economical soln. methods (if such exist)? ② Just how are t. pc's of little ops computed & how are they updated? ③ In "②", are t. cc of t. various little ops related to their effective pc's in any way? ④ Is it poss'l. at present to draw up an exact devn. of t. method? ⑤ How does one move from this method to a more economical, human-like search method?

.01:

Def

G·op ≡ Ground Operator. L·op. = little operator. (is component of G·op)

Derivation of the idea of 64.35……; G·op consists of a collection of L·op$_i$'s.

G·op ≡ [L op$_i$] . Each L op$_i$, when faced w. a problem, will either eventually say "Λ", (I can't solve it)

or give a [redacted] trial soln. So; [redacted] Q$_j$ is the jth problem; L op$_i$ (Q$_j$) = Λ or

= A$_{ij}$ (A$_j$ is the single correct answer). A$_{ij}$ may or may not = A$_j$.

Def

L·op$_i$ has assoc. w. it pc$_i$: the value of pc$_i$ will be updated after each (μ run

sp (≡ stopping point) is completed. A sp is a μ Run. All pc$_i$'s

[redacted] are constant during a μ run, but .

To do a μ run, [redacted] each L op$_i$ is applied to successive unsolved problems,

(the problems are [redacted] linearly ordered in the T S), until it gets a wrong

answ. or until its total cc for this μ run is [redacted] A·pc$_i$.

A is a CB assigned to the T.S., computer instruction pair.

After all L ops have been tried on the μ run, the pc$_i$'s are

updated, & a bunch of new L ops are created. [redacted]

There will be (always) a threshold, ∈, ⇒ there is no point in creating

L ops w. pc's < ∈, because A·pc$_i$ is too small for anything

to be done that could solve a problem. So we construct

all L ops. w. pc's ≥ ∈. After updating, some pc$_i$'s will drop

below ∈, & we wouldn't carry them in many any longer.

[redacted]

For L ops that failed during the last μ run, we will try modifications of

them in attempts to devise new L ops that will work better. Ti problems

of how to do this, & how to assign pc's to the resultant

new L ops (& updated old ops)…… I haven't yet solved.

4 14 82
1 30 AM Ti probs of generating new ops from the cond'l. pc's of old ones, or by modifying

old ones …. I think I understand. New ops & modifications are made on the

basis of the codes of a gen. old op. — & all of the codes of all of the old ops.

— but what about pc's & of the new op & the updated old ops?

First, consider a set of L ops that is consistent over a certain set of problems. Then

the set of Q$_i$'s for those probs, plus the d corns of the set of L ops, gives a complete decision

of that part of the corpus. (Ti part of the corpus for which all L ops give "Λ" can also be

coded in some way, but we need not consider that at present). The code for this

set of Lops should be viewed as a code for an unordered set of objects. We can have a common definitional facility ... shared betw all of t. Lops.

Viewed in this way, it would seem that if t. ave no. of probs ↑, à all of t. Lops remain consi, then their individual pc's should not change. If ave, hwr, one of t. Lops becomes inconsi as t. no. of probs ↑, then we can keep all t. pc's of t. Lops t. same, if we somehow fix the inconsi op. so it now always gives "Λ" as output — usually this will ↓ its pc a little — we otherwise retain all abss. in that Lop, so that t. pc's of other Lops that share some of these abss, will not change.

Another way, would be to delete t. inconsi Lop à ↓ t. pc's of all t. abss it had in it, because their sse's all ↓ (usually by 1).

An imp. Q: [redacted] which is better, [10 Lops consi over a certain set of probs], or [15 Lops consi over a larger set of probs]?

Also, compare to [10 Lops, all consi thus far, plus a 11th Lop, consi for a very large no. of probs., but inconsi for 1 new prob.]

[redacted]

A possl response to t. failure of a Lop (after many successful probs for that Lop.):

1) to Modify t. assoc. ob., so it could distinguish betw. t. past successful à t. new failed Qj. This would be adequate for t. Lop to now "not fail" on that formerly catastrophic problem. It would seem reasonable to try to modify t. failed Lop to deal w. this "bad" problem. Since t. failed op that t. "bad prob" was "its type" (it) is more likely to be "close" to t. correct Lop for that problem.

The only way we'd have to change pc's of Lops, is if some Lops disappear or are modified, or new Lops are constructed.

SM 4:20AM.

1PM SN 1) For a much better system use content addressable memory (CAM) to control [redacted] t. set of Obs calling t. proper Lop. Use of CAM is not nearly because its faster, but because [redacted] of better cc.

2) Perhaps at t. next stage of TM work, I could start out w. an "Expert System" of some sort based on a set of "Rules" à retrofit induction capability into it — i.e. t. ability to devise new rules à modify old ones à get probabilities from Raw R.w. data.

Q: Could I use MTM system (w.o. probly outputs) or would it be easy to have a NMTM for such a system?

.01 T. structure of t. GOP! Each ob. can be assoc. w several LOPs.

Each LOP can be called by ≥1 ob. — in which case the "or" of those obs forms a larger single ob. that calls that LOP.

When an ob is assoc. w. ≥ LOP, those LOPs pool their pc's — but this would seem to not be of importance, since this fact doesn't seem to be used in TM's operations or updating algm.

— So as far as practice is concerned, we can consider

.07 **GOP as being an unordered collection of ob, LOP pairs.** This collection is a kind of "corpus" that we have to derive. (I think this is conceptually impt.) — That is, t. relations of t. LOPs to each other, rather than LOPs to t. problems, that determines t. pc's of t. LOPs — Hvr, see 81.01 → And 85.03-.10 impt!

.08 How do we derive a collection of unordered objects? Well, we can have a common pool of definitions. Ordinarily, we would derive each of t. (finite) objects in some order, perhaps ending each object deriv. w. a "stop" symbol. If there are N objects, t. resultant pc would perhaps be too small by a factor of **N!**, since we have N!-1 other derivs of this sort using diffnt. orders. If we restrict our methods of deriving each object, so that no references to orderings are made, then this factor of N! will be vitiated.

On t. other hand, we may want to have one LOP deriv. refer to another. This could be done by naming t. LOP, in which case t. pc of so doing would be $\frac{1}{N}$. Referring to t. same LOP again (from t. same LOP deriv.) we could have a "ibid" symbol, so t. reference would have pc $\gg \frac{1}{N}$.

When we try to devise a new LOP to replace one that has failed, we try LOPs to minimally modify t. failed LOP. (This uses my version of Chaitin's conditional entropy). However, note that what we are trying to do is maxz t. pc of t. new LOP. We do have a sort of local max if we maxz t. condl. probty of t. new LOP wrt t. old LOP, 80.0 Spec.

SN Q! In my defn, was t. entropy of / condl. probty of x w.r.t. x = 1? T. work was ~ 81 TS 131 ff:

I think t. answ. to this must be **No**, for normed condl. probtys., Because

$\sum_{\text{all } y} p_x(y) = 1$ I think; so $p_x(x)$ could not = 0 unless $p_x(y) = 0$ for $y \neq x$. } But check this.

.01: 79.32: but to get a v.g. max of pc, we would have to either start from scratch & do an Lsrch, or backtrack 1 or more Lop's ..... i.e. backtrack into the history chain of the Lop that failed.

Note: say We have this Lop$_i$ w. low pc$_i$: Then at every Mrun, we will very do few problems w. the Lop$_i$, since its CB, A·pc$_i$, is so small.

.10 What I hadn't realized: the action is in 4 parts: (1) the Mrun; We do with all the Lop$_i$'s out to CB = A·pc$_i$ or until they fail (2) we update the pc's in view of the failures & ∴ # of occurs of Lop's in the "corpus" (3) we do Lsrch for new Lops that will solve as many unsolvd probs as poss'l. (4) we update the

.16 pc's & again (1) loop back to Mrun.

Of course, either (2) or (4) can be omitted (But not both!)

Q's (a) Why use Lsrch in the Mrun? (b) Just how is (3) done?

.20 Well, my impression was that originally, I had the idea for each Mrun, of (1) continuing each consi Lop$_i$ until CB = A·pc$_i$ was reached, Then (2) use Lsrch

.23 on modifns of Lop's that failed, using CB = A·pc$_i$ for trial Lop$_i$'s

This could leave a larger no. of probs unsolved! Say we had several Lops that were correct, i.e. they identified certain kinds/probs & always worked them correctly. If the GOP consisted of those Lops only, then the GOP would never evolve .... it would only solve probs in the area of those kinds & would never Lsrch for new Lops to solve other kinds of probs.

.28 Well, we could (3), out of A.H., in each Mrun, do an Lsrch, starting from scratch, (i.e. no rand pcs) but using current pc's of sub Abss. I vaguely think that I sort of had this in mind when I first invented 64.35... -- Perhaps use a larger than usual CB in "starting from scratch" since the prob. is harder? (Then it should

.32 get progressively easier as we have better & better Abss. & pc's to work with.).

.37 So: .20 – .23 then .28 – .32. Some uncertainties: (A) Moreover It's not clear that the same CB's should be used in the 3 modes of operation, or just what should be the limits on (1) (.20) (B) Just how was (2) to be done & how did it differ from (3)? — in (B) the trouble is w. attempts to make new Lops always being based on the Abss & pc's there of all Lops used. — this is, in both (2) & (3). I guess in (2) the new Lop is automatically a modifn. of the last failed Lop. In (3) it's a modifn of Λ!

So perhaps (B) is clear. (A) is not so clear, but maybe not as impt.

.01 One Criticism of 80.20 ... 29.07 in particular: Say we develop [redacted] as Lops that covers all the probs that Lop1 & Lop2 do & others as well, & does them at less cc. We ought to get rid of Lop1,2, because the resultant GOP has larger pc. by the factor $\frac{1}{pc_1} \cdot \frac{1}{pc_2}$. 80.20 makes no such provisions. In general, [redacted] Lops get no extra credit for solving a large no. of probs — & [redacted] no credit for solns. of low cc. → Hrr., see 85.03 −.40

.05 [redacted]

.06 The way I seem to work probs: I look at the first prob to be solved — categorize it. From the category, I may be able to recognize that it's in a solvable class. If so, I use that class alg'm. to solve it. If I can't recognize quickly a solvable class, I try to x'fm it into a solvable class of probs. [redacted] (This can take lots of time because there are so many possys). Failing that, I may try to solve it in a new way, [redacted] presumably by combining old soln. methods & parts of them. If I solve any prob, I see if I can generalize the method & include it in the indexed set of prob. types that I can solve.

Learning while working probs as sub probs.

.18 41582: 1210A In the large, I don't consider the possy. of solving a problem "incorrectly": In most probs, one knows one has solved it when one does, or one has a measure of how good one's proposed soln. is.

In MTM, the prob is always = the same: (eg. find a hyp c relation betw. the given Q_i's & A_i's, that works for all Q_i:A_i pairs up to now. I guess, the problem is to make [redacted] hyp c modifns of the last successful GOP. If GOP is formed of many Lops, we have "localization of error" to help us.

.27 SN I think, given a bunch of probs in order of diffy, one tries first to find the [redacted] "easiest" (earliest in list) prob that one can't immediately solve. One then spends some cc on it: if/not yet solved, one works more probs that one can solve, then goes back to the unsolved ones, w. better sub abss & better pc's, spends more cc on them, then, if/still not solved, go forward again & work more probs, etc. loop→

.24

.31

A trim on this would be some true "Backtracking" — i.e. say one had really messed up some distance back & got on the wrong trail. Perhaps try this if .2> −.31 (still) fails after much cc on it.

In .29 there would not be much pt. in "going back" unless one solved some new probs — discarded s/o created some new Lops — otherwise all abss & pc's would remain the same as on the previous trial before!

It may be nec'y to distinguish betw. 2 kinds of prob. solns.: One is a soln. of very
.02 small cc, almost always small enuf cc to be acceptable. T. other is a soln. that takes
.03 a difrnt. amt. of time for each prob., & may or may not solve it in acceptable cc.
L srch. may be needed w. probs like this.

Also 2 kinds of "no soln": one is ∧ by all Lops: T. other is due to cc
a cutoff for Lop not producing rite Answ. These 2 situations differ in that
in t. first, ... t. "no soln." occurs after TM has seen Q only. In t.
.10 second, both Q & A are seen.

Apple : desk
fruit : woman

If we apply a Lop_i to a prob_j & after cc = A·p_ij we got no soln., then
[redacted] Actually its a class of .02–.03.
— But its not exactly a soln. or a non-soln.

So : What is t. problem just now? : 53.25 ... seems O.K., but excessively non-el.
64.35 ... seems more el., but I'm not sure about t. details of how its supposd to work —
& I'm not sure it will work. On t. other hand, it seems a lot closer to human
prob. solving than 53.25 .... { 81.06–.18 desrbs human prob solving most similar to what is discussed here. 80.33 – 81.05 (in particular 81.01–.05) are criticisms of 80.20 ... .
At present 80.20 [redacted] –.40 seems to be a reasonable version of 64.35...,
79.01 –.08 is a discn. that seems to clarify just how t. pc's of t. parts of
GOP are assignd & just what they mean.
(81.06–.18 to some extent)
(81.27–.40) is a prob. solving method meant to be close to human prob. solving.
SMI ≈ 3.15A

G. suggs. using small jump prob. solving, then, after little success, use backtrack for less el. soln.

.25 In 80.20 ff when Lop_i faild, I tried to modify Lop_i so it would work.
Perhaps there is a more general way to do this — i.e. one looks at GOP as
a whole, looks at relationships betw. op_s & op_s of success Lops, & one looks
at perhaps how many probs they have solved. From such studies, perhaps
it should be clear as to how close a trial Lop should be, to t. faild
Lop that it is to replace. Viewd in this way, perhaps after failure
a Lop should be discarded & all new trials started wo. using modifns.
from that faild op ... but just start from scratch, using pc's from present sub abss of
"un-faild" [redacted] Lops

Also, After a trial Lop has been constructed, one should perhaps
at least try it on all probs of t. past in which its ob says its relevant.
This may result in an ↑ in t. no. of unsolvd probs. — since several probs
solvd by t. now faild Lop. are no longer solvable. Hwr. t. cc of this looks
.37 very large! Say we are interested in ⓐ large pc for GOP ⓑ large no. of probs that
GOP is known to work for. If we drop an inept Lop, ⓑ ↓ much, but ⓐ ↑ somewhat.
ⓒ (see 83.12) ... low cc for these desirata. → 83.12 spec

Modem
RO
RS R
AS R

On t. other hand, modifying a fwild op (that has been successful for many past probs) by, say, modifying its ob only (so it recognizes how t. prob. on which it fwild differd from previous probs), seems like a very good, efficient method! Using info from other (still consi) Lops may be not v. g., because t. fwild op is t. one that "took care of" t. probs in t. past that one now wants "new goins" of!

4·16·82 12:19A

128k
12k

SM 4:20A (Not much work done this session!)

12:82 9:12P Consider that we want to do these in low cc;

What is best strategy?

When we drop a Lop because it fwild: we would like to know "why" it fwild, so we might be able to devise a new Lop that we could fwall for sure, would work wherever t. old Lop worked, yet also work in some new probs — or at least be able to not fwail where t. old one did. This technique could be a big cc saver!

Anyway, [redacted], after an old Lop is discarded (because of failure), we note that t. set of unsolved probs has suddenly ↑. In particular, we just obtained a bunch of new "easy" probs — i.e. nearer to t. beginning of t. T.S. All of t. old Lops have been tested on those easy probs, but any newly constructed Lop will not have. In fact any newly constructed Lop will not have been tried on all previous probs.! Well, maybe that's not so bad if, IF t. assoc. ob. of t. new Lop is able to quickly discard all "Λ"-type probs of t. past.

Another possy. is to not nec. test all new Lops on all probs prior to their creation. We could just test a few & depend for our confidence in t. new Lop, on t. fact that it was derived from abss & pc's that were obtained from t. past of t. corpus. (Tho to an impt. extent, if t. Lop just discarded was t. only Lop that worked many past probs, then t. abss & pcs of other consi Lobs, will not much reflect in to in those past probs.)

Anyway, [redacted] If we don't test all Lops on a common past, then Gop is not known to be consi in t. entire past. Well, for purposes

403D
4,0,3,13
16445
Poke 16445,
Peek (16445)
AND 8
Poke 16445,
PEEK (16445)
AND (255-8)
00001000
1111 0111
IBM, GE, XIO.
Fujitsu — Machine tools

of prediction, that would not be necy. i.e. to make a prediction, we just use whatever relevant consi Gops are around.

Hvr., viewed that way, we would want to avoid testing Gops out. past, since it would ↑ ■ Likelyhood of failure ∴ ↑ our cc. in finding new consi Gops! → Hvr., see 85:03 – .90

T. strange thing about this present approach, is that t. confidence we have in one of Gop's predns. depends not on ssz of problems, but on t. pc of t. Gop — its "simplicity" in spite of t. consi requirement. In a way, hvr, this is like induction via coding a sequencial corpus.... t. thing relevant to predn. is only t. pc. of a code (that is consi wrt. t. corpus).

Hvr., one of t. things in t. present problem that makes it much different. from sequencial predn. is t. unorderdness of t. set of problems ... while t. probs are orderd heuristicly, they are to be regarded, for predn., as unorderd.

.19 In ■ sequencial predn., if a code works o.k. up to a point, then becomes inconsi, we can always (go back &) stop t. code before it becomes inconsi, & "restart" t. code, ■ w. condl. pc's w.r.t. t. partial-corpus-thus-far.

.23 The pc. of t. punctuation symbol that says to stop one code & start a new one, is not hard to compute.

Hvr., ■ with unorderd probs, we can't stop a code so easily. We have to say in some way, what set of probs it worked on, before it was "stopped" — no ■ so easy to do.

4.17.82. 12:40A

SN .19–.23 is a method of sequencial coding that I don't think I'd considered befor. Is it any good? Well, sometimes: The part of t. corpus where t. old code fails becomes close to random, w.r.t. predn., if this method is used: This is because t. new code string just started & it could start practically any way. T. method is actually reasonably good if t. corpus "changes scene" so to speak, & is continuously — so a new method of coding may be needed for each "new scene".

This sort of occurs w. unorderd objects in an operator T.M., but TM has to tell from t. Q (& ≡ input# • = I) that a new kind of situation is occuring.

A more extreme case occurs when we are extrapolating an unorderd set of objects. e.g. ass from an unknown grammar; Poems; ....

One of t. peculiarities of Q/A induction is t. apparent indep. of t. confidence we have in our extrapolated answer upon t. ssz ... t. no. of Q/A's:

.03 **A** possl. way to ~~s~~ study this: consider a NMTM! i.e. problistic answers.
~~well, I think what~~ Well, instead of having deterministic Lops, we have problistic Lops.
~~As we~~ This is equivt. to several ops for t. same ib, these ops having difnt pc's & sometimes giving difnt. [redacted] Predns. (≡ A's) — t. rel. probs of t. difnt A's depends on t. resp pc's of t. ops., at first, but w. large enough ssz's of Q,A's, it converges to t. rel. freqs of t. difnt. possl. A's.

I don't know a good way to treat t. problem of rel. freq. of various possl. outcomes, in which we know that ordering of events is irrelvnt. — i.e. we know its a Bern seq., but we don't know t. $\vec{k}$ ([redacted] $\Sigma K_i = 1$) of t. Bern seq. I did some work on this recently. } Yes, I do: i.e. t. Bern seq.

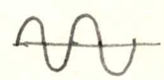
$\frac{k+m}{n+1}$

.18 Perhaps a soln is: $apsi = \frac{apri \cdot N + a}{N + b}$.

N = no. of [redacted] symbol in alphabet.
apri = apri of ~~t.~~ symbol of interest.
a = no. of occurrences of symbol of int.
b = " " " " any symbols.
apsi = apsip of symbol of int.

for a = b = 0,
[redacted]
apsip = apri as desired.
for a, b large
apsi = $\frac{a}{b}$.

$\frac{NK+m}{N+n}$



T. only Q is what value for t. param, N?

Using N = no of symbols in alphabet, seems reasonable, but I'm not sure. Using t. trick of writing t. alphabet, say, then letting t. probty of each symbol = t. freq. ratio
abcd of symbols before it, gives $\frac{1+a}{N+b}$.
— i.e. apri's all = $\frac{1}{N}$ in .18.

Well, I think that t. aprips can have ~~dif~~ difnt weights as well as difnt values.
e.g. say we have observd symbol α for a ssz of 100 & α occurd ~~7~~ 7 times. Now, for t. continu. of this seq. α has an apip of .07 with something like a wt. of 100. This enables us to combine these params w. info appearing later in t. seq.

.35 → { So, in general, [redacted] do pc's always have wts. assoc. w. them? }

Perhaps if t. entire ~~sp~~ pc. is assoc. w. naming t. ~~object~~ abs. ont. basis of how it was constructed (i.e. wrt. t. pc's of its sub-abss) — then maybe t. wt. is/ always 1 in such cases.

.37 Another impt idea: Perhaps using t. ideas of .03 ff, if ~~a~~ Lops is used consi w. a very large no. of probs., then we should ↑ t. apsip of it — i.e. its pc should ↑!
This is a Big difnce in my thinking of MTM operators! ——→ 88.05 spec

.01 T. discn. of 85.03 — .40 perhaps clarifies t. ssz idea in MTM: If a gn. Lopi has ~~spryof~~ pci, Then, if it has been applied w. consistency to n problems, it has a proby of ~ $1 - \frac{1}{n}$ of being consi.

Perhaps more exactly, $\frac{pc_i \cdot z + n}{z + n}$ — tho I'm not sure about t. wt., z. So we should keep track of ssz for each Lop.

Also, if we've just created t. Lopi, Then if the ob's are ⇒ its easy to select out probs of t. past where Lopi would apply, Then chose out a bunch in t. past, at random for testing Lopi. The idea is to mainly see if Lopi is consi, to see if we should use many in storing it — also to get its pc w. some ruff accuracy to it can be used to help form other abss ¿ Lops.

If we ever use Lopi for predn. we could Then select many more applicable probs from t. past to check on its consistancy — t. no. chosen would depend on how accurate we needed.

.20 t. present predn....

85.03 — .40 ¿ 86.01 — .20 comes as a great relief! I really felt very uneasy about t. ssz being almost irrelevant!

I will have to check on 85.35 — When do pc's have wts ≠ 1? — are things like continued Bern seqs. t. only type of situation?

4:25P Also 85.03 — .40 may be somewhat of an impt. Breakthru! It may clarify ¿ simplify many of t. diffys I've had. In particular, it looks like solving NMTM is in one impt way, easier Than solving MTM! — because NMTM has obviously problistic info — ¿ MTM has problistic info That isn't so clearly problistic!

.30 4·18·82 1:11A So: Go back to 80.20..... (Note 82.37 — .40) ← summary of desirata.

An impt. Q is: What to do when Lopi fails? Well, we could continue to use Lopi, ¿ get its ratio of correct replys: So after large ssz, say it get ⅔ of its answers rite, we could use it for predn. Hvr., t for a MTM-type corpus, This Lopi would be a very poor predictor! So we will probly discard Lopi ¿ try to find a better Lop. It seems intuitively reasonable That one way to do this would be to try Modifns of t. faild Lopi ... but what is a good theo. justifn.? Also w. such theo justifn., I want to know how much cc to spend on it, relative to other kinds of attempts to find better Lops.

One justifn. of trying to get a new Lop That works t. probs. That t. faild Lopi worked!

Osiris
ZI 'slow
Wst.
Go wclock.
Iversatil.

4.18.82 ~5 L

— That it is more *likely* to get a larger $\frac{ops}{cc}$ than by "starting from scratch" — This is
because t. faild Lop$_i$ *did* work/^many^ probs that no other Lop worked. (← If it did not,
then we are *not* interested in modifying it to devise a new Lop ... also, we aren't
so much interested, if t. faild op worked *only a few* probs that other Lops didn't).

.05 If this is true, we may want to use t. faild Lop$_i$'s *ob*, *intact*,
& try to modify its op-part only. If we get something that works, it
will work for all of t. probs in which Lop$_i$ worked in which no other Lop applied.

.08 Another approach — (which I considered previously) was to keep t. op part
constant & try to get a difrnt ob that could recognize that new prob. on
which Lop$_i$ faild, is a difrnt sort of thing than t. probs it had (successfully)
accepted in t. past.

Apr. 81
Intel 30.2
Amd : 24.3
Apple 17.4
5.4M profit
first ¼.
2.12M
previous yr. ago.
sls up only 12%
from previous yr ¼

4.19.82 ⟨2 AM⟩ What is t. problem? : well an apparent problem is fixing 80.20 .... (see 86.30ff).
Re: t. ^three^ desirata of 82.37, we could satisfy ^t. first 2 of^ them w. a L search like in 52.25 —
but t. cc would be too hy .... " not al. enuf ".
T. problem of 86.30ff is: what do do when Lop$_i$ fails?

.20 3 Things: ① First find which probs Lop$_i$ solved that no other Lop solved. → System should be ∋ this is not (too difft.)
② simply discard Lop$_i$ & spend time starting from "scratch" in L srch
for better Lops.

[3 desirata: 1) large no. probs worked. 2) hype of Lop 3) lower cc of work involved]

.22 Defs ③ . try modifying *ob$_i$* (≡ ob of Lop$_i$) using *same* *op$_i$* (op part of Lop$_i$)
.23 ④ " " *op$_i$* using same *ob$_i$* as before (see .05)

as before (see .08)

SN Obs may be in 2 parts: 1) easy: rejects most 2) hard ... may take much cc.

~~SN using t. ideas of 85.03 — .40! Modify ↑ part op$_i$ if it gets longer ∋ of correctly worked probs. see 85.37~~

Trouble is, I don't have a clear theory/understanding of
what's going on — so I can't devise an optimum strategy.

.28  The ob being *modified* *could be* changed so as to chose
.29 a difrnt existing op. (This is *much difrnt* from .22) → (89.15)

How limited in (ultimate Lops available), is t. process of
modifying a Lop by keeping t. ob & op tied together, then first
modify t. ob, keeping op const, then modify op keeping ob const, then loop ?

An Ob can be in 2 parts: Part 1 is a boolean selector
mechanism. It uses criteria common to many other obs &
is very fast — can probly be implemented by a Content addressable
Memory (CAM). If this part of t. ob says "ok.", then t.
second part starts. This second part is *not* so simple & takes

more cc, i can't be implemented by a mechanism common to many Obs.

<It may be that there exist now special ll compilers that do just this sort of thing — "single instruction stream, Many ll processors">

Single inst stream, Multiple processors; SIMP or MPSIS.

~1.6% of IRS returns Audited for 1980

~1.5% for 1981 returns due to ↓ in IRS staff

.05 85.40 : I think the way this works is: we have ob_2 and its associated op_2. Initially (apriori) the pc of op_2 w.r.t. ob_2, is pc_2.

After n correctly solved probs, the pc. of op_2 wrt ob_2

.10 it's sort of $\frac{k\,pc_2 + n}{k+n}$ (k = apriori) As $n \to \infty$, it $\to 1$,

so giving the list of Lop's (if all have very large SSZ) is almost an adequate coding of the corpus. (I.e. we wouldn't have to know their SSZ's (= the no. of probs each has solved)).

However, if the SSZ's are not so large for various of the Lops, then a complete descr. of the corpus involves (1) descr. of all of the Lop's

.20 (2) The list of choices made by each Lop ..... Usually, these choices will all be the same (for consi Lops); but we may want to retain some Lops that are not completely consi, if we can't yet find any Lops that are (completely) consi over the problem set we are interested in.

→ However, note that if we can find a set of Lops that are all consi, we can use a Special Abbreviated Form of the simple code for the corpus, that just codes the Lops & writes in all the Q's. — Which is the kind of code that I'd been considering

→ but for 85.03 - .40 : Is there something wrong w. a code of that sort? (low

— i.e. perhaps it's of such low apriori, that its pc cancells out all the gain in pc. assoc. w. not having to specify the choices of each of its Lops.

Also, 85.04... was impt. because it got rid of a (spurious) Simplifn. in MTM. ~~Could~~ Should I go into the coding of the Q's — would this help understand the creative problem better)

SM ≈ 4AM 4·19·82

4·20·82 2AM : Note: If we an Lop is inconsi, we have to first, make the low pc choice — & then derb. the "Answer" corresponding to that Q — This can be very expensive in pc. If we are lucky, we might find some other Lop that gives the right answer — in which case we have to code that Lop. In such a case (where the Lop fails) we could simply write out the Answer itself as its code, and the

"Prediction power" of our [redacted] system would be zero! i.e. ≅ random Answ.
Hvr, if t. Lop was correct m times out of n, it would still have
a probty of $\frac{k p c_i + n - 1}{k + n}$, which is useful. — This is t. probty of its predus being correct. The probty of $\frac{k - k p c_i + 1}{k + n}$ (≡ $\frac{k(1-pc_i)+1}{k+n}$) of being wrong, is also t. total probty distribu. over all other answers. We need not go into the coding of these or t. actual distribu. involvd.

Re: t. problem of wts (≡ ✱ of 85.18; ✱ of 88.10). For a simple Barn Scep. 2 TB 141? ("coding w. definitions") does give a wt. of ≈ 1, but I'm not entirely sure — Also I must consider just how t. aprip of t. various defns. was computed.

.15 : 87.29 A natural way to modify a failed op_i, is to try all other consi op_j's — one's that have not been selected for t. faild prob. This could also be looked upon as a case of 87.28 — i.e. modifying t. ob_i (but not keeping t. op_i constant).

.20 Another possl. "Use" of t. faild op_i: It may work o.k. w. different existing obj's. This is one way of modifying ob_i & keeping op_i constant. — It is a rather easy-to-do thing. Perhaps low cc .... but I don't know how promising its pc is! [redacted] Hvr, it does not solve ~~the problem of the present failure~~ the problem that op_i fails on. — It may solve some other problems, hvr.

.25 O.k.: So when a Lop fails, I have all those ways to try to get a Lop that will work for t. faild prob & for probs where t. Lop did work o.k. (but no other Lop was "called". relevant!))
How much cc I should spend on each of these tasks/types is unclear;

.30 [The "ways" are: 87.20-.23; 28-29; 89.15; 89.20]

Viewd in this way: Tho it can't solve t. faild prob., it certainly could solve other probs that haven't been solvd thus far. When a Lop fails on a prob. we aren't merely committed to trying to find an Lop to solve that prob. All we really want is maybe a good return on our investment: i.e. Max $\frac{\Delta pc}{\Delta cc}$ — we can work on trying to solve any probs that would help hereafter

T. "ways" of .25 — .30 can be viewd as "PLAN"s. T. goal is max $\frac{\Delta pc}{\Delta cc}$ & each aspect of t. plan gets a certain pc .... This pc. perhaps being based on some prelimy obs on t. situation.

There is some Q on what $\frac{\Delta pc}{\Delta cc}$ means! in particular Δpc. Say we are trying to code this bunch of [Q, A]'s. We have part of it coded. What is the relative Δpc's involved in finding a ~~[illegible]~~ consis' Loop for Q_{10}, A_{10} rather than Q_{20}, A_{21} ? —

Eastern A.I.; Brantley

I'm not sure all of the forgg. is nearly relevant. Perhaps go back to fo/ actual Training Sequs. & see just how impt. these Q's are. — Also — consider what humans seem to do in various training situations & see if any of the forgg Q's are relevant.

4:10 AM **One way to run a T.S.:** work probs w. each Loop until there are no remaining probs in the corpus that it can work. When this is done, one works on the remaining probs only, using the / set of faild Loops as "conditions" for condl. pc's. Still, we have the Q of what to do about faild ops.

4:40 A SM

$\frac{e+5}{1+5} \approx \frac{5}{6}$

4·21·82 1.50 AM **Well,** lets go back to the alg. notation T.S's — unary & binary operators, then the "eval" operator .....

Re: the "Eval" op: consider β, β^2 .... β^{12}.

tail

say β works for ~5 probs, then fails. We then try to modify β so it will work. Above way back, when I ran into trouble, I was assuming β had reasonable pc, even after β failed — so I considered the pc of β^2 as something like the pc of β times the pc of the integer, 2 ~~[illegible]~~ mult. by the pc of the "end" symbol, Δ.

Now I question the hy ~~[illegible]~~ pc of β, since it was used only in a faild Loop.

The pc of the sequence of 5 ritts is / "wrong" answer would be:

$pc_\beta \cdot \frac{1}{5} \cdot \frac{1}{5}$ ← pc. of End symbol

pc. of "not β"

$pc_\beta \cdot \underbrace{\frac{1}{2} \cdot \frac{2}{3} \cdot \frac{3}{4} \cdot \frac{4}{5} \cdot \frac{5}{6}}_{\text{pc of 5 "}\beta\text{'s:}} \cdot \underbrace{\frac{1}{6}}_{\text{pc of end symbol}} \cdot \frac{1}{6} \cdot pc$

pc of the "raw" form of the non-β answ. that did occur.

$\approx pc_\beta \cdot \left(\frac{1}{6}\right)^3$

No! this is all wrong! I'm not at all sure!

I need to go over this coding of operator T.S's more carefully.

SM 4 AM

4218275

956 P.

.01 Say one has $n-1$ occurrences of symbol α in 1 of β. (These are unordered events). Then considering that each code for α is t. same in its prefix as in its code (for α & β), as we have to use α's to define t. codes, what's t. p.c. of t. whole code we can get by optimum coding.? A constraint is that

.05 t. code for $n-1$ α's must be a string of $n-1$ repetitions of t. same code (w. possibly a new symbol) 4.22.82 2:40 A --- Actually .01 – .05 is not exactly correct: As stated, we probly get Huffman codes. If we also consider t. possy of other codes, so that we sum probtys to get final probys, then we may get more exact answers – even for a finite corpus.

$\frac{1}{2} \cdot (\frac{1}{2})^{10}$; $\frac{1}{2} \cdot \frac{1}{4} \left(\frac{1}{2}+\frac{1}{4}\right)^{10} = \frac{1}{8}\left(\frac{3}{4}\right)^{10} = 7 \times 10^{-3}$

$(2048)^{-1} = 4.886 \times 10^{-4}$

An example: Say we have 2 ways to we want to code $\alpha^{(10)}$:

we have, say, 2 ways to code α, one has pc $= \frac{1}{2}$, t. other pc $= \frac{1}{4}$.

If we use both defns, t. pc for this packet only t. pc $= \frac{1}{2}$ defn,

t. pc of t. corpus is $\frac{1}{2} \cdot (\frac{1}{2})^{10} =$ · $.488 \times 10^{-3}$

If we define both codes, t. defn. plus has pc $\frac{1}{2} \cdot \frac{1}{4}$; & t. recs of t. corpus has pc $\left(\frac{1}{2}+\frac{1}{4}\right)^{10}$: t. product is $\sim 7 \times 10^{-3}$

So in this case it's best to use both defns.

In general, for pc = p it will be marginally o.k. to include a code of pc = ϵ for $ssz = 10$ if : $p \cdot (p)^n = p \cdot \epsilon\,(p+\epsilon)^n$ | $\epsilon = \left(\frac{p}{p+\epsilon}\right)^n$

$\frac{1}{\epsilon} = \left(1+\frac{\epsilon}{p}\right)^n = \left(1+\frac{\epsilon n}{p n}\right)^n \approx e^{\frac{\epsilon n}{p}}$ $\epsilon = e^{-\frac{\epsilon n}{p}}$ $\ln \epsilon = -\frac{\epsilon n}{p}$

or it's worth while to use t. code if $\frac{\epsilon n}{p} > \ln \frac{1}{\epsilon}$

This is a good approxn. if $\frac{\epsilon}{p} \ll 1$ and n is large.

Otherwise use $\boxed{(1+\frac{\epsilon}{p})^n > \frac{1}{\epsilon}}$, which is exact

Maybe not correct

writing a multiple defn. (e.g. t. code says α & β both represent t. obs. to be coded) may have a diff pc. Then simply $p_{c\alpha} \cdot p_{c\beta}$.

T. forgg. analyses suggests that using >1 code for a symbol can result in t. effective pc of that symbol being very close to its relative frequency — much closer than using a single code representation ... as by Huffman.

4.22.82 TS

Plasticity Defacetrack

consider now, our defining the new concept symbol, which = a^b. a & b have pc's pc_a & pc_b, resp.

Lets represent the concept a^b by the code sequence, α. Suppose α occurs n times in the subsequent corpus. The pc. of the whole thing is then

.06 $\approx \underbrace{pc_a \cdot pc_b \cdot pc_\alpha}_{pc\ of\ a\hat{}b} \cdot (pc_\alpha)^n$ .....

Now: What objection was there to giving pc_α a very hy value (≃1), so as to make .06 ? Well, perhaps we should have some symbol in the code that represents an alternative to the new symbol.

— Then [redacted] $pc_\alpha \equiv p$ we want to maxz $(1-p)\,p \cdot (p)^n$ ← pc of alternates symbol

$= (1-p)\,p^{n+1}$; $= p^{n+1} - p^{n+2}$ | $\frac{d}{dp} = (n+1)\,p^n - (n+2)\,p^{n+1} = 0$

$(n+1) - (n+2)\,p = 0 \therefore p = \frac{n+1}{n+2}$

so p = (n+1) occurrences of α / (n+2) poss. " " α. ← (?).

Metagenerally;

$\frac{d}{dp}\ (1-p)^n\,p^m$

$= n \cdot (1-p)^{n-1} \cdot p^m + (1-p)^n \cdot m\,p^{m-1} = 0$

$-n\,p + (1-p)\cdot m = 0$

$\frac{p}{m} = \frac{1-p}{n}$ for max pc. so $p = \frac{m}{m+n}$

4.23.82 : 4:30 AM : Perhaps a good approach at present: Assume that the problem of [redacted] assigning pc's to simpler seqs of abss. for the Operator induction, has been solved. Then, using only a few properties of this soln., continue w. the Trg. Seq. work. See just what properties of this soln. are needed. Then, looking at that pc evaln. prob. in a more genl. way, try to solve it.

.28 4.25.82 2.16A Have been reading IAIC II: sections [redacted] 4.2.1 – 4.2.2, pp 233–236. This deals w. pc's involved in making defns. & includes costs of punctuation.

T. problem I want to solve now: [redacted] for an operator TM. Say one makes a defn. Lop_i of pc = pc_i. After n [redacted] consis & m inconsis cases what probty do we assign to the next case being consis?

— Well, how do we do the coding?: First we code GOP, which contains [redacted] Lop_i. Then we code the m+n choices that were made. This would seem to make the pc of the choices in/p of pc_i a funct of m & n only.

Q→ T. Q is, is this correct, or should it be a funct of pc_i also, & if so, in what way?

I&C II (92.28) has a way of mixing t. pc's w. min, but it's not immediately clear how this could apply to an operator TM. In I&C II, t. codes for t. corpus & for t. defns are on an = footing & are coded "together". In an Operator TM, t. codes for defns are in t. dcrn of GOP, & are separate from t. dcrn of t. sequences of choices.

One of t. diffys of this way of coding t. operator inducer is that (large ssz w. hy consi) is not reflected back in t. pc's of t. Lops involvd. — i.e. no "rewards for good behavior"! — so no localizn of "credit" — & it's (this) "localizn of credit" that I mainly want to get from this particular approach to Operator induction!

It would seem that somehow t. coding of t. Lops & t. coding of t. choices must be somehow made to interact.

One way to see how this might be done, is to write out some (non-approx) method of doing operator induction — then see how & if I can get t. desird interaction.

→ The I&C II approach to condl. probys used defns of zgms to obtain this effect.

An approach to doing this in Operator TMs:

say t. I/O pair A→B occurs many times. We then define t. obs. A→B ≡ α. If ≡ A→B n times, then t. corpus becomes α occurring n times. If A→C occurs 1 time out of n, we could define β ≡ A→C & then t. corpus is n-1 α's, 1 β. — (unordered). If a new A occurs, we can code it as either A B or A C; i.e. α or β. — t. rel. pc's being ≈ n-1 to 1.

Here t. coding of t. operator Dcrn & t. coding of t. corpus seem more mixd (?).

SM 4Aug

.35 12:30P In t. forgg, we could use t. total freq. of α & β in both t. GOP dcrn. part & t. selection instructions part, to determine t. pc's
.35 of α & β. T. 2 parts may have to be treated diffntly
.36 since one is ordered & the other is not.

One way to deal w. t. diffy of .35–.36: Consider t. unordered dcrn part as t. parallel "sum" of all possl. ordered dcrns.

One Big Q: I had previously considered MTM operator TM's as being described by the operator only — this operator being chosen so that all answers were correct for the entire corpus. Such a system would, I think, involve a non-univl. machine, since all possl. corposes are not describable: e.g. an undescribable corpus would have A_1, B_1 ; A_1, B_2 as part of the [D.I] set; w. $B_1 \neq B_2$ (i.e. 2 possl. distinct answers for the same Q).

2 PMSw: Motorcycle left going North from OAK.

Then — it would seem that the foregg. system would give a shorter decrn to the corpus than one in which alternate answers to the same Q (can / must) be considered. —

.16 Perhaps the MTM case is a "special situation": That there exists a bunch of Lops that are capable of dealing w. the corpus adequately. — That ssz of the probs is irrelevant. — That the TS must be devised so that each problem is solvable using an Lsrch w. CB ≤ A (← some fixed value). — That the probs (int. T.S.) must be ordered to a large extent; but if they are somewhat misordered, then

.20 certain probs will not be solvd the first time they are tried w. CB = A — since the relevant sub-abss haven't been discovrd yet — so other probs must be solvd as we periodically go back to earlier probs to see if the recently devrd abss. now make it possl. to solve the old probs w. CB = A,

.27

42682 940P It seems that (in line w. .16) MTM is significantly distant from NMTM in that (at least for operator TM's) the ssz is irrelevant (as long as it is > 1). Superficially, it seems simpler than the NMTM machines. The search routine I'm thinking of is for .16–.27

.36 For a new problem, we are prepared to expand CB = A. We get (usually) a bunch of solns. We use these solns. on the subsequent probs, & as we solve more & more probs. fewer remain consis. When all of them drop out, we expand CB = A again (loop to .36).

If a prob. is not solvd in CB = A, we go on to the next prob & do CB = A, etc. until we solve a new problem & get some

new solns. using abss that we didn't have before. Then, we go back to t. previously unsolvd probs & try to solve them using t. new concepts. (inline w. 94.20-.27)

An alternate strategy might be to wait until several new probs have been solvd, or until an "adequate no" of new abss. have been devisd. .04

T. forgo strategy will work, if t. T.S. has probs in it that have solns. that build on previous solns. — each soln. having Levin < a certain constant, A.

One trouble seems to be that most abss never get used by pc. This is because abss are created or "used" only when a previously unsolvable problem is newly solvd, (using CB = A). Thus — so if this happens N times, $\leq SZ$ for most things (unless one soln. contains > 1 case of a gen. abss) is $< N$.

Now I guess N can't be very large because then NA (which is t. total cc expended by TM on t. corpus) would be too large. [Is "N" t. total no. of new concepts in t. T.S. ?

I'm not so sure of this.

42782 12.01AM

Any way, t. Poggy suggests that t. T.S. wouldn't fit pc's of impt. abss very much — in which case t. $\frac{cc}{pc}$ of various solns will be very hy — i.e. A will be very large.

So superficially at least, t. MTM problem in this form looks like it may not be practically solvable by t. method of 94.36 — 95.04 This Q is perhaps solvable. What I can do is look at t. pc's if we regard it as a MTM prob., then look at t. pc's if we regard it as a NMTM prob — Do this for some simple T.S. — say t. one I've already worked on for unary & binary funcs & then t. β^2 v.s. loop soln. for "Eval".

Now: T. MTM problem is 94.16 — 95.04. Then At all times we have this GOP, and when an unsolvable prob arises that's unsolvable by t. current GOP we do a CB = A search to find a soln. T. search is over GOPs that are minimal modifns of t. old GOP. Those "minimal modifns" are ordered by ~~Kandi~~ my version of Chaitin's "conditional entropy". This conditional entropy — or [illegible]

Cost of modifying t. old dcrn — can be done in various ways. To get a good idea as to what some good ways are, look at human solns of t. T.S's, & see what kinds of "modification operators" are usually used in searching for soln.

.05 For NMTM, T. discn. of 92.28 — 93.40 (93.35-.40 in particular) gives some idea as to how to treat this kind of problem.

.10 On second thot, maybe MTM & NMTM need not be so distinct. Say we treat t. prob as NMTM, but it really is a MTM T.S. w. CB=A. After many many problems, it becomes clear that t. best way to code these things is to use values very close to 1 for certain conditional pc's & very close to zero for others. This effectively makes t. soln. about identical to the MTM soln. — i.e. t. pc of t. "string of choices" is ≈ 1, & so it can be almost disregarded.

Also, it is still true, that t. consi GOP itself plus t. set of Q's constitute a complete dcrn. of t. couprs — so, in line w. conventional NMTM theory, this dcrn should be gen. a lot more pc than t. some GOP plus Q's plus a bunch of decisions.

So; there does seem to be a problem here; That t. MTM T.S. doesn't seem to give v.G. pc's for hierarchical discovery of abss.

There is no reason why a MTM T.S. has to be "discoverable" w. practical A CB=A. It may be that one has to pretend it's a NMTM in order to get a search technique that works.

.30 Part of T. problem is devising a sequential buildup of t. GOP's of T.S. construction, so that t. pc's (or more exactly t. $\frac{c}{pc}$'s) are acceptable. (i.e. ≈A).

Consider a human trying to solve t. MTM T.S. would such a human treat it as a MTM or a NMTM? — Well I'd guess as a NMTM, but humans usually talk as if they were dealing w. MTM — i.e. single (deterministic) values, non-probabilistic concepts.

SAM

SM

12 40 P A human solving a bunch of MTM probs: If subsets of them were all conceptually very similar, t. human would give t. subset a ssz of ≈ 1 only.

Perhaps, more generally, t. human would regard t. ssz of a bunch of probs as being ex t. no. of new concepts needed to deal w. the set of probs.

So these last 2 sentences support that T. MTM model may be workable by normal Human means & possibly via t. techniques of 94.16 – 95.05

A very mildly N-MTM T.S. is a MTM TS w. a few errors in it. T. best code for this is t. [redacted] dcrn of t. GOP for t. correct T.S., plus a dcrn. of t. errors.

Most (if not all) current A.I. work on learning & induction of concepts, involves MTM-type TSs only; & t. approach is always (I think) to find "T. 'correct' GOP".

In working on a pure MTM T.S., t. technique of using Lewis entropy in order of incremental conditional entropy of t. previous GOP that was consi> .... would such a search algm. be able to develope & use "plans" & other heuristic devices? — Or would such devices only be developable only in a/ larger NMTM T.S. & then be transferable to MTM sections of t. NMTM T.S.? 4:15P

— This may well be true, that many heuristic devices that are used by humans to solve MTM TS's are, indeed, derived from NMTM TS's & could not be derived directly from t. actual MTM TS's worked on only

If this is true (& it seems reasonable), then, for a MTM machine, we will have to give it some "plans" & other heuristics, if it is to be able to solve diffl. probs.

I suspect that Pat Langley et.al. are giving their machine fairly complex heuristics — but that these heuristics might be in hierarchy unalterable by t. kinds of machines they are working on — that a NMTM type T.S. machine might be able to learn such heuristics.

.30 SN T. Problem of a Mis-ordered T.S.: Say we have a MTM TS. which is workable by t. algm of 94.16-95.05 — almost but all new probs are workable w. CBEA t. first time. This is an "ideal T.S." Now, say we move some permuter t. ordering of some of t. problems. This Alg. will still be able to solve t. T.S. in some cases, (but w. [redacted] (augur cc.).

.36 In other cases, it may not be able to solve it at all — This could occur if some of t. problems had spurious solns that were retrieved by TM instead of t. correct solns. — which weren't yet accessable (because needed abss were not yet found). e.g. something like t. β^{12} or β^{5} "solns" for t. "Eval" problem.

see 110.10 ff also 113.38

{ Could we devise search strategies that would get around this difficy, & be able to deal w. as poorly ordered T.S. as poss(. $?

.03 A very critical T.S. would be one in which each problem reorders t. abss. of the previous prob if it was to be solvable in < CB=A. In this case any changing of t. ordering would make t. T.S. unsolvable by t. technique of 94.16 – 95.05 ⁘⁘⁘⁘

abssy

No! It would be solvable

1 2 3 4 5 6 normal order.

1 2 5 4 3 6 disturbed order.

.10 On second thot, TM would be able to solve this TS. using 94.16 – 95.05' t. order of solns. of t. disturbed T.S. would be 1, 2, 3, 4, 5, 6. After solving 2, it tries 5, then 4, both unsuccessful: then it works 3, goes back to 5, then fails, then works 4, goes back to 5, works 5, then works 6.

Note 4 is tried once & 5 twice, unsuccessfully: so wasted 3.A of cc.

.20 Perhaps an impt. diffy in nice T.S.'s as well as somewhat misordered misordered ones, is t. diffy of 97.36–.40 – spurious solns (like β_{-}^{12}).

I think 94.16–95.05 it's very close to 53.25 & 53.35'. In 53.25 there are only examples to eliminate spurious h.H. solns. In 53.35 there aren't enough examples so spurious solns. are carried along w. A in cc of computations.

64.35 ff is for a particular (more & l.) model of t. GOP: 65.30 is very enthusiastic about! In this model t. GOP is t. sum of many Lops. The method of change of pc w. ssz discussed on 65.18–.20 is probly wrong if t. ideas of 94.16–96.30 are correct.

.35 In 94.16–95.05, one thing is unclear: we apply t. GOP to a problem & it's solved. We go on to t. next problem & t. GOP works on it. How much cc are we allowed to spend before we decide this GOP can't solve it? — CB=A? Also, shall we allow GOP to say "Λ" ("I can't solve it")? For it to be able to say "Λ" could save much cc.

I think I discussed this before; don't know if I arrived at any conclusion here.

Discn of this: Probly using CB=A would not be bad. This is sort of like t. decision of what ratio of time to spend before TM1 & TM2 work —

→ See 101.32–.36 for soln. !

42782TS

.01 T. idea is that one couldn't be off in final efficiency by a factor of > 2. → 100.09

42882 1230A

describe t. corpus

So: Comparison of 3 methods: α 53.25 & 53.35 $\quad \alpha_1 \quad \alpha_2$ → actually 53.25-.35 descrb. t. T's. 39.35 — 40.08 descrb. t. search algm. all together called α; 39.35 - 40.08

β 64.35 ff $\quad$ all β

γ 94.16-95.07 $\quad$ " γ.

Actually, they aren't so difrnt. γ is $\approx \alpha$, but w. some improvement by having a method of dealing w. some mis-orderings of t. T.S. (see 97.30 - 98.20). In both α & γ, GOP is a general (completely) operator.

.10 In β, we elementalize t. GOP into a bunch of Lops (≈ 1.7 t Lops) which are ob; op pairs. This arch. has t. advantages discribd in 65.30. Also, β has a feature that enables us to carry along fewer A.H. solns. — when a new Lop is created, it is tried out on as many as possl. probs as it can do in t. CB (say A?) This can save a lot of mem by getting rid of A.H. solns. rapidly. It does mean that TM must have access to much of t. future of t. corpus — rather than work on probs sequentially. → 100.30

This is unclear: i.e. can it spend A on each problem or does it solve probs until A is expended or until it gets a wrong answer?

.18 I think that .10 characterizes how β differs from α & γ.

SN In all of α, β, γ; (verify) this dirty: when one makes a modifn. of t. Gop, one has to t. new GOP on t. entire past corpus, as well as any new problem. Previously, I felt that I might do verifn. on a sampling of t. past, but w. t. most recent concepts of T. MTM, this sort of statistical treatment wouldn't fit in well.

In β, where we have individual Lops, it may be possl. to verify an new Lop on t. entire past.

An alternate possy: to just verify Gop (or t. Lops) on t. probs in t. past at which a failure had occurred — so that it was necy to devise modifns off. old Gop (or Lops). Actually, if a Lop, say works at those points, it will probly work at t. rest — so perhaps on t. average, it wouldn't take too long to verify cons on t. entire past, since this will be done for only a few Lops. Or one could verify t entire past consi of a Lop only when that Lop had to be used in predn. (Well, what about use of that "Lop of unknown consi in modifying pc's of other Lops? — perhaps we will always have to know for sure if a Lop is consi).

On t. other hand, for practical prodn., it might be dirt. to verify new Lops (or Gops) on selected samples of t. past. This would give t. probs. a certain statistical uncertainty in addition to t. uncertainty obtained by various Lops giving diffrnt Answers to a gn. Question. By analysis of SSZ & perhaps use of random samples of t. past, we can make an estimate of t.

likelyhood that a fn. Gop has been consi for all probs up to now.

If we have a certain Gop & we have completed its pc on t. basis of all of its sub abss., then [redacted] the accuracy of this pc estimate is indep. of whether that Gop is consi or not.

— so t. probty of conistancy over t. entire past is a separate calcn. that we make.

2:24A So: It looks like β is O.K., (even better,) w. t. refinements of γ.

β is an elzn. of γ, but a v.g. one, & certainly close to t. way humans do it.

Probly I should write a detaild review of just what α & γ are & just how β is an elzn of them

SM: on some working examples we should work using a "consi" Gop.

09:99.01 11PM When a gn Lop_i was devel its $\frac{ccr}{pcr}$ was $< A$. [i.e. ccr ≤ Apcr] s. for new problems giving Lop_i & CB of A would probly be much too large!

On t. other hand [redacted] using t. limit A·pcr seems too small.

64 ×27 64×100 1600 lbs. ×5 8000 lbs

42982 2AM Ah! Perhaps a new/rich idea (or a new life on an old one!)

When trying new GOPs (in α & γ), after one is found, that works t. present problem, use it to work new probs on till t. total cc exploded & conditional pc of t. Modifn to produce t. new Gop [redacted] > A. This way, we are sure we spend up to $\frac{ccr}{pcr} = A$ on each problem if necy. We go as far as we can here, then we devise a new set of trials w. t. same "work into t. future" & we go as far as we can within t. CB.

A trouble here: T. "correct" soln. may require CB = A = $\frac{ccr}{pcr}$ for 1 problem.

So it would go a minimal distance into t. future.!

SM

30: 99.18 7PM NP T. trick of 98.03 = 78 is impt! This is done often by humans! Unable to work a problem, so work others that involve devising new concepts — then go back to t. unsolvd prob., hoping t. new concepts will be adequate.

33 9:40PM There is another kind of "Ansr" to a problem that I've not considerd: I have considerd those in which T.M. says either "Λ" or gives an answ. Another possy is a soln. like Lsrch, [or successive approxn. to solve an eq. in which t. method will work for all probs in a certain class, but may take "too long". In such cases (like t. soln. often a (p. eq.) t. method never will never give a wrong answer — but if we use a [modified pre-interrupted C.B., in 2A] It will, at t. end of that time, either have a correct answ. or say "Λ" — but it will use all of cc = A to say "Λ"! → See 101, 37 for ideas on this.

38 Perhaps outlaw answs of typ .33 for t. time being, & return to this diffy later.

Ah! See soln. on 101. 32—.36!

very nice!

—T. idea being that each soln. takes a more or less known, fixed time (≡cc).

4:30 82 | 12:30A

5M | 3:45A

10:30P O.A.! a review: first α & β:

α: the T.S. is described in 53.25 - 53.35! T. search algm is described in 39.35 – 40.08 (Actually, 39.35 is the beginning of a very long development. Various improvements & variations of 39.35 – 40.08 are discussed. —T. discn. goes at least to 47.20, in which I devised the f curve ↑ ↓, which I think I later found irrelevant to most of the present work on MTM) It may be relevant to NMTM.

Hwr., 39.35–40.08 is the only w. some possl. improvements:

.11 Start out working the T_1's w. an initial GOP_0 & using $CB = A$, test all Gops in order of "distance" from GOP_0. (Distance = conditional pc — my improvement of Chaitin's def'n). Test each Gop to $\frac{cc_i}{pc_i} < A$. No need to double A", etc.

Retain all solns. & see how far into the future each will go. Retain the one(s) that go farthest. Then, for the next prob, loop to .11 (but with GOP_i instead of GOP_0)

There are several diffs in this method, but they are all solved by 101.32 – 102.01!

.22 1) When a Gop has been found & one "sees how far it can go into the future", I guess "Λ" is regarded as failure. —But how much cc is allowed for each problem (100.33–.40)?

Allowing $CB = A$ seems wrong, because one usually isn't allowed $cc_i = A$ when looking for new Gops, i.e. $\frac{cc_i}{pc_i} \geq A$, so $cc_i \geq A \cdot pc_i$ which is usually $\ll A$ (since $pc_i \ll 1$.) (see 100.09)

This (.22) prob. was discussed on 98.35 ff ... < which eventually evolves to 100.09 >.

100.38 says that we (problem) outlaw # solns.

.32 that can take very variable amts. of cc.

Another possy. is to allow only $cc \leq A\,pc_i$ for each soln. — even as # for probs. after the first one.

GREAT!

5.1.82 | 12:10A | 52+

Final!

.36 $\frac{cc_i}{pc_i} \geq A$ for all problems, this procedure must work!

Since T. T.S. is defined so that (T. definition T.S. is given in 53.25 & .35)

.37 : 100.34: For successive approx. probs & Lsuch probs. of this sort., One can work them w. $CB = A$, but only after studying them, so that in $\frac{cc_i}{pc_i} \geq A$, we have pc_i very large – say close to 1. "studying them" means we know by studying the prob. approximately what cc is needed to solve

t. problem using that algm. If it is almost certainly << A, then pc of [redacted] that particular solm. method is very hy (i.e. close to 1).

.03 ... ↑ well, not so clear as to what's happening here!

.04 pc_i is not t. proby of Gop_i solving a particular prob.

.05 → Σ pc_i = 1 (or < 1 if unnormed), but it is likely that > 1 Gop_i can solve a particular prob., so if several Gop's have ~1 proby ~1 of solving t. prob (which is often true), then Σ will be > 1. !

Well, in t. least formulas $\frac{cc}{pc}$; pc is something like the probability of success in solving t. present problem of t. algm. being considered. pc is t. proby that algm being considered will solve t. present problem. Σ pc_i < 1 → t. proby that any of t. algms being considered will solve t. present prob. ← only if impos. it's impossl. for >1 algm. to solve t. problem!

.11

.12

I think this takes care of objection .03 → — The t. mechanics aren't entirely clear. I guess we have this algm that looks at a problem & computes ~ what cc is needed for algm_2 to solve it — or more exactly, it obtains a proby distribn for t. cc of soln. Then, t. Σ proby of all solns w. cc < A is t. pc. we are interested in.

Well! .05 seems to contradict .01 - .12

First; t. pc of any Gop is t. proby of that Gop being created by random input to a Universal Gop creating machine; t. random inputs are prefix codes so Σ pc ≤ 1. These pc give apriori probys (= aprips.). pc of a Gop can also be conditional upon previous (Gop) output of that umc. (i.e. Given that t. umc has produced Gop_0 as output; with continued input, what is t. proby that

.27 .29 it will produce Gop_1 as output? ← not quite! Since t. codes are prefix codes, one can't "continue" from — Hm., I think there is something conceptually important relative to my new defn. of Conditional Entropy. The resultant eq. for another sorts of finite objects (prefix codes). is:

.36 $Gop_0 = M(P_j, \Lambda)$ $j = 1/\infty$; P_j are all t. possl. prefix codes for Gop_0. T. first arg of M must be a pref. set.

The / a lowable 2nd arg is also a prefix set I guess.

The probty that P_j created GOP_0 is $2^{-\ell(P_j)}$.

GOP_1 is created by M, but w. info about the correction of GOP_0 —

So $GOP_1 = M(q_{ij}, P_j)$ Here q_{ij} is P_j taken on all possl. values for which this is true, but in addition P_j must satisfy 102.36. T. probty that q_{ij} & P_j created GOP_1 is t. product of (probty that P_j created GOP_0, which is $2^{-\ell(P_j)}$) × (probty that P_j created GOP_0, t. probty that q_{ij} created GOP_1 / which is $2^{-\ell(q_{ij})}$)

so t. unnormed condl. proby of GOP_1 is $\sum_i \sum_j 2^{-\ell(q_{ij})-\ell(P_j)}$.

But I don't have a "nice way" to say it — (like 102.27-.29, which is wrong!)

.17 ⇒ Well anyway recfourning is nos motions! — i.e. 102.03. I know what t. pc of GOP_0 is GOP_1 are, i.e. t. condition $\sum pc_i < 1$ holds for both o.a. (I guess for GOP_1 also ... but I haven't checked just now).

— but what about t. "probty" that 2 gn. successive approx n. method will converge to accuracy 10^{-3} w. $cc < A$? T. whole idea of CBI is how to go from aprips to apsips after one gets data (i.e. via Bayes). (≡ partial corpsis). ≡ pc

---

.25 (≈ SN) T. result of 101.32-.36 is that TM spends only a very small fraction of its cc working problems. Only $A \cdot pc_i$ is spent on each problem, & pc_i is very small. On t. other hand, $\sim A$ is spent on each search. As a result, t. total cc needed to do a corpus is $\approx < N \cdot A$; N being t. no. of new concepts in t. corpus. We assume that t. T.S. is perfectly ordered. If there is any mis ordering (like 98.10), then t. method of 98.10-.20 can be used .... & this can ↑ t. total cc of t. corpus a great deal.

Apparently from t. diffy of 102.03, t. way I picture it now, we search for GOP_0 to solve t. first problem, using $CB = A$. We then go into P.S. (probsolving) mode, using $CB = A \cdot pc_i$ of GOP_0. We stay in P.S. mode until we fail. (either ∧ or no soln. in $cc = A$ or wrong soln.), then we go to search mode for GOP_1 / using $CB = A$ — using conditional pc from GOP_0 to GO_1 & loop to

5.1.82

Other than 102.03, an impt. problem is how to deal w. all of t. parallel solns. that are outputs of t. search mode. Well, 103.25 suggests a way: When each new GOP_i is created w. success for 1 problem, it is tested out on successive probs until it fails. T. no. of probs it solves is N_i (for t. i th successful trial version of GOP_i). (This testing in prob. solving mode takes little cc) — We then retain in Many only the trial GOP_i's w. Max N_i. When we finish t. search, we have only this/ (presumably) small set of GOP_i's, all w. t. same N_i.

.40 for consumers

70's son

20's son

.10 This may be a huge no. of GOP_i's! One way would be if they are all essentially t. same, w. some sort of (perhaps obscure) way of reducing into one another: In which case they would all have t. same pc. We might have several sets of these — each set t. same: So to get true pc's we have to add together (same) t. /pc's of all t. members of each set. Hrr, 2 GOPs may be operationally identical, but Heuristically different.

The GOP_i's in t. retained set (those of Max N_i): Take one's of highest pc will be given more wt. in trying to create conditional operators for t. next problem.

.25

.26 → Still (maybe) big ⊖: I decided sometime during t. last day or 2 or 3, that only t. GOP_i's that went farthest were to be considered. This is an impt. decision & I don't remember t. reasoning of it.

.29 → Go back & see if I can find t. reasons! → Around 96.10 is where this decision occurred most recently

.29 & 102.03 seem to be t. impt. probs just now!

I think most recently, it was an idea I got after reading 39.35 ff.

.33 2:50PM — SM — Heavens! — 4 AM

Say I have this T.S., & it's a T.S. of form 53.25 or .35, but I don't know t. A value. So I start out w. some small value of A — say A_0 — & I work on t. T.S. try to code t. T.S. using it & if. If it turns out to be impossl., I start out again w. $A = 2A_0$ & go as far as I can go. If it doesn't work I do $A \leftarrow 2A$ & loop, etc. T. impt. thing is that t. cc for working t. T.S. is $\propto A$, so we

Great!

.01 do not waste anywhere much cc by repeating this after $A \Leftarrow 2 \cdot A$

.02 One thing that makes this a not absolutely clear algm. is γ: in which (At least) we do 98.03 -.20. This last Algm. is not clearly defined — there are various params that must be fixed, so one can tell when t. sub-search has failed so one will go back to $A \Leftarrow 2A$.

.10 On second thot, $A \Leftarrow 2A$ doesn't necessarily double t. sub-search cc. It could ↑ t. no. of GOP$_i$'s we retain in storage w/o ↑ t. cc of storing those algms. ↑ no. of Algms stored does ↑ cc of search (by a factor over t. "A" value) because we have to consider unpruned conditional pc's from each of those GOP$_i$'s. So we have to find ways to reduce t. no. of GOP$_i$'s carried as much as poss(.: (see 104.10 -.25)

.10b Apparently, then, (other than t. difficulty of (.02 a .10)), it looks like t. cc needed to learn a T.S. is cc t. least of its most diffic. abs.) Since to solve it, one must use that some value of A in search for all of t. abss. in t. T.S. This seems kind of wasteful — i.e. it would seem that one shouldn't spend so much cc on t. easier abss. ... but there is no way to tell which of t. abss is truly easy — i.e. often one does not find out till much later in t. T.S. that one went wrong somewhere back, but one doesn't know just where t. error (i.e. settling for a lower cost abss. than one should have) occurred — so one starts over from t. beginning, after $A \Leftarrow 2A$.

I think this is wrong: should be $A \cdot C$

An ↑ in A need not multiplicatively ↑ t. cc of doing t. sub-search — it can ↓ it by finding better abss. so that t. T.S. can be worked w. fewer abss. cc of t entire

.35 Def T.S. $\approx A \cdot \bar{N} \cdot C$ $\bar{N}$ = mean no. of Gops refined; C = no. of concepts in t. T.S. for which a (new) (t. past) check is needed.

Def

So, as we ↑ A we may ↓ C; I don't know if $\bar{N}$ will ↑ or ↓

Another / the worry: checking on a new GOP$_i$ in to t. past. 101.32-.36 suggests that this need not be a very large cc. In general, t. cc of checking valid GOP$_i$'s will be indep of A, so as A ↑, it will be of less impt. relative to cc spent in searching for valid GOP$_i$'s.

.01 : : Try to get a more exact version of 105.35 — which includes cc of working problems.

This may be larger than I thot! Since t. cc is ∝ t. square of T.

Def .03 T.S. length! Say each problem takes an/average cc of $\underline{w}$ to work it;

~~[scribbled out]~~ (This is only approx. — a gn. prob. will take diff. t.

Def .of cc — as t. GOP,'s change is "Mature"); on t. average, $\underline{m}$ problems per concept

Working all probs thru t. i^th concept will then take:

$m \cdot w \cdot i \cdot \bar{N}$; so total $= \sum_{i=1}^{c} i \; (m w \bar{N})$

.17 $\simeq \frac{c^2}{2} \cdot m w \bar{N}$. for [problem] s. v. s $\qquad \frac{c(c-1)}{2} \simeq \frac{c^2}{2}$

. $c \; A \; \bar{N}$ for [GOP, search] — Looks wrong! [CA] looks more reasonable!

so $\frac{\text{probs cc}}{\text{search cc}} = \frac{c m w}{2A}$. [Here A is the largest Lcost problem in t. [redacted] T.S.]

It would seem that since c could get very large, that eventually, t. cc spent on prob. solving would predominate. This c^2 effect is perhaps impossl. to avoid, if we are to check t. entire past each time. Even if we only check 1 prob. from each concept, this only ↓ badness by a factor of m. (≡ no. of probs/concept).

A more likely way is to check on selected probs in t. past — a no. of probs that doesn't →∞ as c→∞ !

The factoring of GOP's into LOPs may be very helpful here.

Another trick is random sampling in t. past. If an error is made (by not checking on some impt prob.), then one [redacted] ultimately [redacted] ends up doubling A/at least one more time than is needed. Hvr., this random checking on t. past is done also many times w. unchanging A — it would be well if that part of t. search could be usd to effectively check on t. randomly chosen probs...

.34 (9:54PM) Break for supper w. Steve, Marylinn;

.35 [9:38P] (11:18PM) Another way of dealing w. t. "$\frac{c^2}{2}$" problem: Only check a fixed distance back into t. past. I think that something like this may be done by humans. — See 136.01 ff for analysis of this.

At any rate, I don't think humans ~~[scribbled out]~~ do get into t. $\frac{c^2}{2}$ problem ..... I'm not sure as to exactly how they avoid it, but .35 may be part of it.

5.2.82 TS : More like IPC, hrr.

A method to do parallel search for G-O-P's, using many relatively indep. processors.

The problem is search of a tree w. some fixed $c_B = A$.

Def T. way it is done: One p.u. (≡ processing unit) is initially assigned to do t. entire tree search. T. rest of t. p.u.s are initially "unemployed".

.06 ≡ T. first unemployed p.u. picks an employed p.u. at random (there is initially only 1 employed), & asks to share his tree. The employed p.u. gives t. unempd p.u. t. "largest unexplored branch" in his tree. [Think this "largest unexplored branch" can be uniquely defined.

T. search of tree itself is a simplec strictly left-to-right exhaustive search of all nodes w. $\frac{c_c}{p_c} < A$.

The next unempt. P.U. does like .06, as do each of t. next unempd. P.U.s — picking employed P.U.'s at random to ask for jobs.

When a P.U. finishes his job, he becomes unempd & → .06.

When there are no more employed P.U.'s, t. job is finished.

There are some Q's about sharing of memory, when an unempd P.U. asks for a job. This need be done efficiently if we want to get good eff for t. entire task. There may be a better way to chose working P.U.'s (to ask for job) than by random choice.

Also, when a branch is broken off, t. P.U. on t. rest of t. tree must take proper note of it & not try to explore t. broken-off part.

When any P.U. finds a soln., it is reported centrally & that P.U. continues his search as before. Or, he just remembers t. soln., & reports it centrally at t. end, when all P.U. are unemployed.

.01: 10/7.20: In γ we have a trade off betw. t. disorder distance (= t. max misordering distance of a concept) & increasing A. Lets call t. ~~[illegible]~~

Def.03 Disorder distance, D. A T.S. w. disorder distance D means that no concept is out of place by more than D concepts. E.g. in 98.10, D = 2. D does not involve t. no. of examples used for any of t. concepts.

~~We can devise~~ Given ~~[illegible]~~ t. "D" of a T.S. ~~[illegible]~~ (width of type α), we can probly devise a search algm. that will do it in min cc.

Not so easy: Even if we know D, we still would have to know how many examples ~~[illegible]~~ each concept could have, before we could use t. search method of 98.03 - .20.

Def Well, using ~~[illegible]~~ 98.03-.20 we need to know d which is t. ~~max no. of problems out of order that t. T.S. has. distance out of place of~~ same as D (.03), but t. distance is measured in ~~[illegible]~~ units of problems rather than concepts. If, in a T.S., all concepts have exactly k problems (= examples) then

$d = k \cdot D$ exactly.

If we are given an upper bnd for t. ~~[illegible]~~ d of a T.S., but we don't know A, then we solve it.

If we are ~~[illegible]~~ an upper bnd for A but we don't know d, then I think we can ~~solve~~ it using 94.16 — 95.04 — i.e. ~~[illegible]~~ when a unsolvable prob is found, we continue to try to solve successive probs w. $C \bmod d = A$, until we solve one. Then we go back to t. first unsolved prob & ~~[illegible]~~ loop.

~~[illegible]~~

Hmm, this can be very cc consuming! & looks very inefficient.

If we know neither d nor A, then we can do a 2 dim search over A & d — i.e. each time we trail we go thru t. corpus w. larger values for both A & d. This can be very inefficient, because we could end up by making, say, A, much larger than it need be, just in order to do enuf increments of d, to solve t. T.S.

→ 111.11

At present, I feel most insecure about 5), 107.22 (& 6) 107.24 has bibliography

On 5) (saving only the GOPS solving most probs bears failure) Biblings:

104.26 | 39.35 ff | ~96.10 : The most recent decision to do this was made around 96.10.

96.10

SM 3 AM

.10 SN 1:30P If a spurious concept solves more probs than the correct concept, then we would discard the correct concept. This would make the T.S.'s of 53.25 — .35 unsolvable by present methods!

6:35P Actually, ~~[illegible]~~ an effect like .10 can be quite common! In writing the T.S. one usually feels that the concept one has in mind really will "fit best" & solve the most probs. — but in fact, this need not be so, and the unforseen concepts may actually go past (in the sense of solving probs for which the next concept was that to be needed) the expected concepts. If this is so, then the expected concept will definitely not be acquired, & further on in the T.S. when this expected concept is needed, it will not be available.)

5·3·82 1 AM The effect of .10 would seem to make less vital the Q of 107.22 — i.e. "Is it best to retain only GOPS that solve the max no. of probs?"

→ Another way this can occur is: say the cost of a certain concept is only A_1, but the cost of the most diff concept in the corpus is A_2 w. $A_2 >> A_1$. We will, hwr., have to use $CB = A_2$ on all concepts, including the one needing only A_1, & we may find unforseen, v.g. concepts this way, that can throw Monkey Wrench into the works.

→ Perhaps one of the main reasons that I find retaining GOPS w. max no. of probs solvd., is that it seems to work for the TS's of 53.25 — .35.

.30 I think I'm getting a much better idea of what the MTM problems! These diffys & variations of 53.25 really are imps.

The idea of just retaining GOPS that solve most probs, probly will have to be modified to retain more GOPS: but w. diff. criteria. I used to have the idea of saving the best 1000 (or so) "easy" codes" — i.e. codes for Max pc. For the present MTM diffys., I may have to devise some such scheme — perhaps a method that will not necessarily work all the time — (Up to now, I was considering constraints on the T.S. so that the search methods I was proposing would be infallible over those T.S.'s.

5.3.82 TS

T. main constraints considered: (1) finite CB (this is not a constraint — but if t. cost of t. most diff. concept is very large, t. cc for t. soln.

$\langle \approx A\bar{N}^{c}$ (105.35) $+ \frac{c^2}{2} MN\bar{N}$ (106.17)$\rangle$ may be too large

.04 (2) large am't of no. of probs per concept 53.25 (3) fewer probs

.05 per concept 53.35 (4) Some misordering of concepts, &

.06 (5) T. (GoP concept) able to work most probs / not nec'ly t. (Desired correct) ← in t. sense of being needed (as in t. T.S.

one: 110.10 — .30

111: 109.40 One sort of heuristic way to deal w. d, is after initial (first) failure to hunt around — w. CB = A — for probs that can be worked w. very slitely new concepts: i.e. searches w. small CB. } Humans seem to do this.

(small "A"). Then, after some of these are found, — w. these new "little" concepts, go back to t. previously unsolvable probs.

W. human teachers & students; A teacher will devise various probs to see if a certain concept has been acquired. If it appears not to, then he

.20 gives new probs directed toward t. acquisition of that concept.

J. Lennon
Ringo Starr
Paul McCartney
George Harrison?

5M 3AM

(4).05: In T.S's devised for Human students d20; i.e. there is assumed to be no misordering. — Hvr. the (5).06 diffy may, indeed, occurs in TS's for human students — & it is usually (often) dealt w. via .20 or by ↑A or by some kind of "Backtracking" by t. student.

5482 12 45A

> If superficially done, this can lead to a diffy like "β^{12} being learned rather than "t. Loop soln". To fix this, when various probs are worked at low CB, we do retain t. concepts derv'd, but later, we redo those problems at hyer CB. The idea is that every "new" problem (i.e. one needing L such) must be solved at t. same hy level of A ... or else there is danger of t. β^{12} v.s. Loop

.37 diffy

Actually (5).06) is a diff. prob. in teaching human as well as inorganic students.

The 2 "relaxing of constraints" of [redacted] (111.05, .06) are very impt. — if I could get TM to deal w. them, it could solve many sets of problems that were not at all carefully ordered into a T.S. — Perhaps close to a human chosing probs from R.W. that it could solve. — sort of devising its own T.S.

.07 One possl. way to deal w. 111.06: consider pc's of various Gop's involved. — So one might return a Gop w. fewer solvd probs if its pc is very hy! This was the idea of 47.20 ff. — but that developement was, I think, for NMTM. When I go to NMTM, some of those ideas may be useful for MTM.

.11 Another approach to 111.06! Up to now, we have been choosing for each new concept, the Gops that solvd most problems. We could extend this over 2 concepts so that stay for the first concepts we retain all Gops w. the largest & 2nd & 3rd largest no. of probs solvd. From all of these we "sprout" new conditional Gops, & then we chose the Gop pairs that gave the longest seq. of probs solvd. This can be ganzed. for seqs. of N concepts.

122.25 is more narrowly relevant

116.03

.22 Another approach to 111.06! We have as an ultimate goal, obtaining a Gop for the entire corpus that has max pc. However we can also use it as a kind of subgoal, sometimes. Let us consider all sequences of discovard Gops as possl. developements of the finale Gop. These Gop sequences will "stop" at various concept points and do their CB=A searches, creating a new Gop for each of what it finds to be a nearee concept. Many of these possible developement strings will terminate before the entire T.S. is done because they can't get solns w. CB ⊆ A. So anyways following out all these developements, we have several Gops that have done the whole corpus. We then can use all of them for predn. (run 11) or just use a bunch of Max pc.

.35 In .22 ff. we only pruned the trees when a branch couldn't continue w CB ⊆ A. Now, we could also "prune out" branches that were clearly far behind the others.

e.g. Branch 1 has pc = α_1 & has worked p_1 probs.

" 2 " α_2 " " p_2 "

If $p_1 > p_2$ & $\alpha_1 > \alpha_2$ then Branch 1 is uniformly better than Branch 2.

If $p_1 > p_2$ but $\alpha_2 > \alpha_1$ it's not clear as to which is better

If $p_2 > p_1$ but $\alpha_1 > \alpha_2$ " " " " " " " "

If $p_2 > p_1$ & $\alpha_2 > \alpha_2$ Branch 2 is better.

Around 47.20 I used something like $(pc)^{\frac{1}{2}} \equiv \alpha^{\frac{1}{p}}$

as the ~~criterion for~~ Gave for various developments. — This is consis w. 112.35

.10

Similar to 112.22 but less elusive (less "leaving things out"):

We have this set of Gops that code the corpus up to different distances.

Gop$_i$ has pc_i. ($\Sigma pc_i < 1$.) We then do an L search on continuations of each Gop$_i$. This gives a tree. The search can be done several ways.

.18 1) say pc_{ij} is the products of all cond'l. pc's along a particular branch. pc_{ij} is the pc of a particular sprout GOP.

Then use $\frac{CC}{pc_i \cdot pc_{ij}} < A$. Note: $\Sigma\Sigma_{ij}\, pc_i pc_{ij} < 1$.

Also note that this $\Sigma\Sigma$ gets smaller & smaller as the tree gets pruned by failure of certain branches. This & the CC of L search to $<< A$, so we can have A values that are very large, to compensate for the $pc_i \cdot pc_{ij}$ is becoming very small.

2) Same as 1) but only consider the branches in which $pc_i \cdot pc_{ij}$ is within the top 1000 (say) values. ("i.e. 1000 best codes").

3) Somehow use $(pc)^{\frac{1}{\ell}}$ (ℓ = no. of probs solved; pc = total pc of Gop)

.31

.32 to help prune the tree.

N.B. The foregoing ideas are based on various heuristic concepts, some from humans. — but the real criterion of "what's good" now is: How much corruption of the ideal T.S. of 53.25 can it deal w.? The 3 big corruptions (thus far) are 53.35, 111.05 & 111.06 ③ ④ ⑤ : How much can we corrupt the ideal & still have the search method be assured of a soln.?

.38 N.B. The tricks of 112.07 ff to deal w. 111.06 don't deal at all w. 111.05 (the "d" effect).

SM 3:45A

.01: 3:47P 9.10P As I see it, 111.06 can be dealt w. only somehow deciding to retain solns. w.o. Max M_i ($M_i \equiv$ [redacted] 106.05) is t.no.of probs. worked by a GOP_i after it its discovered ... just before it fails).

Dick — T. Q. is, what is t. criterion for retaining a GOP_i in memory?

.05 A possl. way to do various of these approaches that would normally use too much cc: Just use t. normal soln. method of selecting GOP_i of Max M_i. Then, if failure occurs, Backtrack using any of t. proposed solns from 110.10 to 113.32

.07 A particular kind of Backtracking that's of value is 112.11: we go back 1, 2, 3 ... N concepts, & we chose N of 112.21 to = Risk.

Backtracking can be avoided by carrying along all the seeming (apparently) sub-optimal branches that one would otherwise backtrack to, — but its a Q. of which technique uses less cc. I suspect that ordinarily, t. GOP_i of Max M_i would be finite concepts, so backtracking would be infrequently necy, & carrying along all t. branches needed to avoid backtracking would on t. average be more costly

.23 AHH! a way to unify diffys 4) & 8): In both cases one runs up against a problem & fails because t. rite sub-concepts aren't available. In 4) these missing concepts are in t. future. In 8) they are in t. past. (Later: I'm no so sure this is a v.g. "unification") → 115.04

.25 SN In organic Evolution, we cannot afford to carry along with us often concepts that are not used/ or not a/o critically enuf. In 112.11, our various proposed ways of dealing w.t. diffty of 110.10, we could carry along a large number of (apparently) sub-optimal (in terms of M_i size) GOP_i's — but organic evoln. would not carry them very long w.o. some sort of "use" of t. sub-abss. in those GOPs.

So, if one decides to not backtrack (.05) much, one MUST "carry along" a lot of sub optimal GOPs. Each non-optimal GOP carryd can be characterized at any pt. of t. work, by (a) How non-optimal it is (b) How long it has been carried w.o. any useful application. Both of these qualities can be parametrized in many different ways. In (a) the optimality

112.11: In org. evoln. so optimal concepts don't get carried very long, or they'll be lost. See .28

5.4.82 rs

can be measured by pc , $pc^{\frac{1}{2}}$ (113.31) , (maybe 113.18 is relevant?) . In (b) ("how long carried") we have to decide on "how useful" a particular application is — so we can determine if we are > or < threshold .

.03

.04 ; 114.27 : The Really Bad case occurs if the critical concept needed is (1) in the future & (2) its not the concept of max M_3.

A goal approach here: When failure occurs (in "normal" Gops of Haus method)

.10 Def Look at future & past $\frac{1}{2}$ distance 1 & depth 2 . If no progress, try distance 2 & depth 2a , etc. — use linear or geom expansion of distance & depth, untill one gets to a certain pre determined level. — If no soln. , A ← 2A & start over on the T.S .

.18 → It may not be necy to have the same A for the entire T.S . — This idea could potentially save a lot of cc .

.19 One can ↑ A for a problem until one gets a reasonable M_3 ← meas mot lobe.04? - i.e. no. of probs solved . , then continue. If one gets into trouble, backtrack, & ↑ A , ↑ a (.10) & ↑ t distance (.10). Just how these 3 increases are to be related is a Big Problem ! Anyway, as soon as the prob. is solved, one goes back to small A & loop to .19

A possible aggravation of the "d" problem ; (98.03-.20)

Order is 1 2 3 4 5 6 7 8 ; i.e. one has to get 2 or more concepts from the future before one can continue.

A natural way to do this : after solving 3 , & failing on 6 & 7, one works 4. Then, either try 6 or try 5. 5, because it is close to 4 , also maybe try 7 because its close to 4 .

Note that 1 2 3 . . . 8 are concepts points, rather than individual problems, so the program of working them is nearly more difficult than the foregg. outline .

.35 Another kind of T.S. diffy : say the (sub concept needed for a particular problem. Gop actually is not in the T.S. Then this can be remedied by either (1) Gross ↑ in A or (2) T.M. devises his own problems & these furnish the needed concepts. It may be that (2) is a normal hour device & so a TM w. suitable "background" would know that hour , & would automatically

.01: do it as part of ↑ of A → Hvr., in general, if t. T.S. lacks a certain needed concept, it is an inadequate T.S. & may be essentially unsolvable w. reasonable cc. → It may be that a MTM could not learn such a heur —— that only a NMTM could learn it. → 124.10

.02

.03: 112.21 ! Review of this idea ! 112.11 derbs it poorly & — but its important.

One way: At each concept. point, we retain the Gops having the 10 (say) largest M_i's. (This could mean only 10 Gop's are retained, or it could mean many more, if we ~~follow Gops w. t. same M_i to only~~ retain only Gops w. t. 10 largest distinct M_i's). As these Gops are sprouted (we note that they are sprouted at 10 distinct pts) they give rise to new Gops of distinct M_i's. Again we retain t. ~~10~~ Gops w. t. "top 10" M_i's — But these M_i's are added to t. M_i's of t. previous sprouting. We continue sprouting & retaining t. "top 10" — The M_i's used for rank, being t. total no. of probs solved at that branch tip.

.20

.21 A variation: ~~not to take to be~~ Retain not t. "top 10" but use some in ΣM_i function of pc_i & ΣM_i — (ΣM_i is t. total no. of probs solved to date by that Gop) to order t. Gop's

Def .25 Like pc_i (ΣM_i) ~~best~~ ($= \frac{1}{\Sigma M_i} pc_i$) NB at end of t. T.S., all ΣM_i's are t. same, so we retain t. Gop's of highest pc — ~~retained~~ which is what we want → 124.31

This measure is ≈ (cost of corpus / ΣM_i) ≈ bit cost per problem. → t.o. summation index!

.03 — .20 looks ~~much more like~~ not much like 112.21 & more like 112.22 - 113.10 . .03 - .20 doesn't specify a fixed N. —

So .03 ff may be ≈ but better than 112.21 ff.

Hvr., note that t. method of .03 ff could also (perhaps at least cc) be done by Backtracking (114.05). Normally, we would carry along t. "best 10" Gop's, say; but when backtracking, we go back several concepts & retain t. "best 10" ~~based on t.~~ until we solve a intractable prob. that necessitated t. Backtracking.

~~t. point seems like a mistake~~ .03 ff (a sum of t. stuff of 112.07 ff)

Seems like a reasonable approach to t. prob. of 110.10.

Hvr., t. Misordering problem is very diff'l: partly because we have no idea as to which (future) probs introduce new "concepts".

.38 The work on this prob. thus far: 98.03 - .20 ; 109.01 - .40 ; 111.09 - .37

This is t. idea of "δ" → 94.16 - 95.04

.01 : : 2:52AM As of now, we have ≈ 3 search params: ① A (≡cB)

.02 ② T. no. of Gops we will retain (or some such w param. assoc. w. 116.03 – .25)

③ The distance into t. future we will look for ~~missing tone~~ misorderings off. T.S. (γ) (see 116.38 for Bibliog).

T. only way I can think of dealing w. them is to do a 3 dim scan over them. I would probably do this scan over a "backtracked" region of t. corpus — otherwise ~~we~~ perhaps toomuch coosh. Anyway, t. ~~relative~~ path in t. 3 spaces is unclear ... t. relative jump sizes of t. 3 parms? — should I use direct, log or exponential values for any of t. parms?

"ON Backtracking" an Art. & Bibliog on Search: IE³ PAMI May 82 pp 309-315 MAY 1982

SM 3:20A

11PM How to deal w. all ff! Other than t. scan methods, I'd like to reduce it to 2 params (to reduce t. arbyness of t. search path) – & ideally, to reduce it to 1 param., so no arbyness!

Seems to me that I had a similar difficulty w. a 2 dim. scan & I was able to resolve it into a 1 dim. scan. This had to do w. relatively recent work on t. B^{12} v.s. loop soln. ~~In this~~ t. 2 scan params were A, t ∈ B for L such & (perhaps) t. no. of probs solved by t. Gop. (Gops weren't invented yet, however!).

First, just consider 2 params in .01 ff. A & other — skip ☐

②, t. no. of Gop's retained.

.26 5.6.82 12:18 AM One way to look at it: ~~Each~~ (Gop, problem) pair ~~can be~~ characterized by 2 params: ① How far down t. Gop's is) from t. top (in 117.02 ⑤) = x

② How ~~far~~ into t. future ~~is~~ is t. problem?

On t. basis of past experience, we can assoc. w. each x,y pair, a probability that t. associated Gop will "succeed" (i.e. that a sprout from it will occur that will solve its problem)

3 Defs: h(x,y), f₁(x), f₂(y), [its likely that t. probby funct will be of t. form $h(x,y) = f_1(x) \cdot f_2(y)$ — which makes ~~it~~ it easy to get].

h(x,y) is like t. prip of success for that Gop, problem pair — ≥ h(x,y) < 1 (or = 1 for normed h). So using h as a pc,

.37 we can do an ordinary L search.

The function h(x,y) is impt. We will probly have to give T.M. an initial form for it (so a wt. & a way to update it, statistically.

T. form of $f_i(x)$ [redacted] (x is how far down from t. top) will depend on t. ordering criterion for x. See 116.03 – .25. If t. ordering criterion involves pc of t. Gop, it may be easy to guess a form of $f_i(x)$.

At this point, t. Q is, what t. point is that I'm using in S. Lsrch of 117.37: This [redacted] seems to be related to t. (apparently) impt. problem of 6): (107.24): t. discussn. of (101.37 – 102.03 (& to 103.40)) is on t. meanings of pc's in various situations.

[redacted]

One big trouble w. T. "guesstimate" of 117.26 – .37: If t. pc's of various concepts are much distant [redacted] from those estimated by t. writer of t. T.S., then [redacted] t. (writer / trainer) can't be sure that he's written a T.S. of form 53.25 (or .35). [The idea of 53.25 – .35 is that a soln. for T.S. exists that has a certain max subj. s.t. each T. concepts are in t. rite order, & t. rite concept is always one that has M_j max.

How bad a deficiency is this?

[Hmm, he can't answer, because he can't be sure that there aren't spurious concepts that are "temporarily" better than t. ones he wants to be t. "right" ones".]

Hmm., t. technique of 117.26 – .37 is for T.S's that are not t. nicer ones like 53.25 & .35.

Anyway, there are 2 [≈ thms. involved w. pc's of Gops:

1) that by doubling t. A of a section of a search, t. cc of that section of search is approxtly doubld.

2) That it's safe to use A • pc as t. c.B. for problems to be solved by a Gop that has already solvd 1 problem.

I suspect that both thms are true, no matter how t. pc's are defined.

.24 O.K. so we have this set of Gops $[Gop_x]$ & this set of probs $[p_y]$ (p_y is y problems into t. future). We have this probability $h(x,y)$ that Gop_x will be able to have a spawn (≡ modifn) that will solve p_y.

$\sum_x \sum_y h(x,y) = 1$, say. From Gop_x, we devise a set of modifns of Gop_x — These are M_{xi} — [redacted] t. pc. of M_{xi} is pc_{xi} & pc_{xi} is t. conditional pc of M_{xi}, given all of t. codes for Gop_x (actually it works even if we are not just given Gop_x ... we don't want to be given all of t. codes — as we do in Chaitin's cond. probby).

.01 To do search we must spend a max cc of $A \cdot h(x,y) \cdot P_{x,i}$ on t.

.02 creation of $M_{x,i}$ & its attempt to solve P_y.

Note: if $P_{x,i}$ is normed, $\sum_i P_{x,i} = 1$ so $\sum_{x,y,i} (h(x,y) \cdot P_{x,i}) = 1$.

SN Diffy: In .01 -.02 we use this upper limit of cc for both t. creation of $M_{x,i}$ & t. working of P_y. Hrr., in my previous formulation, after $M_{x,i}$ has successfully solved its first problem we allow it t. same cc limit to solve subsequent problems only. No need to take cc to create $M_{x,i}$ — so maybe we should subtract out t. cc needed to create $M_{x,i}$.

Well ····· is it always possl. to divide t. soln. of P_y by $M_{x,i}$ into ① a cc for construction of $M_{x,i}$ & ② a cc for applying $M_{x,i}$ to P_y?

I guess it isn't always possl., but when it is, we can save cc. by only allowing t. remainder (after t. $M_{x,i}$ construction part of t. cc is subtracted out) for working on t. problem i.e. $\{A \cdot h(x,y) \cdot P_{x,i}$ — cc needed to generate $M_{x,i}\}$ is upper bnd on cc allowed for solving P_y. We would try to put our $M_{x,j}$'s into a form → This is possl. to compute.

If the cc to generate $M_{x,i}$ is ← t. cc needed to solve P_y, then t. forg. dis-cm. is not very impt. Hrr., if the cc needed to solve t. prob is much smaller, relatively, t. forgg. would be an impt. way to save cc.

T: forg. discm. assumes that problem working takes up a large part of t. cc of T.S. solving. If we check on t. active part of all new Gops we run into the $\frac{c2}{2}$ diff of 106.17 ··· If we find a way to deal w. t. $\frac{c2}{2}$ diffy (& I think it's necessary that we do so!) then it's likely that cc for solving probs will be a minor part of t. cc used to solve T.S.'s ···· i.e. searching for concepts will be t. main consumer of cc.

→ This seems very reasonable! If $h(x,y) \cdot P_{x,i}$ is very small, this means that we should (intuitively) expect a diff. search, & we will, indeed, need about $\frac{\text{cc of } h(x,y) \cdot P_{x,i}}{h(x,y) \cdot P_{x,i}}$ to find that soln.

.01 → We have in the Burg. assumed that h was a funct. of x & y & probly of the form $h = f_1(x) \cdot f_2(y)$. This would be true if we were heuristically "young" (= naive, untrained). With a clever search strategy h can be a function of many things in addition to x & y. h(x,y) (& we can generalize x to be any way of selecting Gops) can be dependant upon certain observations on the problem, y as well as observations on Gop x and it can also depend on (x,y), the xform from Gop x to $M_{x,y}$.

.12 — .14 The more h is an intelligent function of, the more literly it is, that the x, y values of the successful trial will give a h(x, y &, ...) That is very hy — so less A needed for the search.

.01 is very important It tells how we can use h(x,y) to control the search, by making the probly distribu. over x, & y a function of various observations on the probly, y & the Gop, Gop x & the spawning xfm from Gop x to $M_{x,y}$. Any other ideas of search, involving "plans", or whatever can perhaps be introduced as methods of modifying h(x, y, ...). (Additional things h is a funct. of.)

E.g. Say we got stuck at a certain point in the b.s., using just the "Gop of Max M_i" search algm. We then look at the prob. we failed to solve (obs). We then look for other/new probs that we may be able to find new concepts from, that would be helpful w. probs having this obs. A strategy of this sort would give a probly distribu over y (y is now not primarily an index of distance into the future, but is now being used as a way to identify problems)

.31 .12 — .14 seems to be very relevant to (107.24) & (see bibl.)

— T. prob/zmof how TM can solve probs (like successive approxn probs or Lsrch) that take sig diff. amts. of time for soln., & may for certain probs, not work for certain ce thresholds.

Another, even better development of .01 is to have h(x, y ...) be a function of what x's & y's have been examined thus far. This thus becomes a sequential search problem — perhaps like the "research methodology" problem of "devise a search algm. to find a cure for cancer". This is an algm. that gives initial experiments, then tells what to do, depending on the results of the experiments, ... & from some expts is loop

123.28 spec

5.6.82 TS

.01 : see 105.18 A major problem: How to keep A localized — how not to use t. same A for t. entire corpus. Perhaps related to t. $\stackrel{c^2}{\geq}$ prob of 107.07. Perhaps try to break corpus into "chunks" w. "A" constant over each chunk?

(so A will vary w. difrnt concepts (≡ problem sets))

What I may want to do now: Write detaild exposition of t. TS's 53.25, [redacted].35, also t. method of solving them. (perhaps go into 53.35 .... how much more cc is needed for storing solns, etc.) Then introduce t. various degradations of t. T.S., & write out t. present idea of solving it, in some detail. Then relist t. unsolved probs.

see 134.01 for derivn of 53.25 (+ ≈ 53.35)
see 135.01ff for derivn of a soln.

It may be worth while going into β probty soon. This may suggest solns. to problems.

Also, eventually go to NMTM for solns. to probs in MTM search. Essentially, that's what 120.01 is! — appln. of probty concepts that wouldn't occur in MTM!

(SM) ~ 330A

.25 ⟨230P⟩ [SN] A prob. I worked on befor: Don't remember if I got any good ideas for solns:

Say a particular modifn. of t. Gop_x; $M_{x,i}$ has 13 difrnt ways to code it. If I knew they were identical, I'd just try one of them & give it a cc threshhold of $A \cdot 13 \cdot pc_{x,i}$: Unfortly, as it is, I don't know they're t. same, so I give each a cc threshhold of only $A \cdot pc_{x,i}$ & I do this 13 times. Total pc is same in t. 2 cases, but in t. second, I lose cc by a factor of 13. (a)

[In t. fopg. work there's much attempt to deal w. (b) (.30 – .40) & little if any attempt to deal w. (a) (.25 – .29).]

Very serious / 2 serious badnesses / looks like a serious problem!

use of 120.01ff for sequential search algms. also.

.29
.30 (b) In general, effects like .25 can be multiplicative to obtain an enormous no. of effectively identical solns. E.g. say we have α codes for Gop_1, & Gop_1 is retaind. If each of t. codes mutates to β difrnt codes for t. same Gop_2, we have $\alpha \cdot \beta$ codes for t. same Gop_2 — This can continue to $\alpha \cdot \beta \cdot \gamma \cdots$ etc. Also, there is no way to drop out these redundant codes — so we retain them for t. entire corpus code! In fact, if there are C concepts in t. corpus, & $\bar{N}$ Gops; then $\bar{N} \approx k^C$, where $k > 1$. i.e. $\bar{N}$ will exponentially produce identical Gops.

One could suspect 2 codes if they had exactly t. same pc. — one might then try to prove identity.

Hvr. it may be also possl. that a certain code may Give t. same Gop as another code — yet have a pc of 2 or 4 or 2^n times as much

(I think it's commonly true. Any object has many codes of difrnt lengths to describe it. Is this ever or often true?)

122.01 spec

.01: 121.40 [5.7.82] (12:20A) A possl. approach to t. difty of 121.25: Devise a language so that redundancy of this sort is very unlikely.

.03 A (perhaps adequate) way to avoid t. exponential ↑ in N, t. no. codes: As far as storage of such codes, this could be done in "factored form", so t. amount of code storage would be ∝ C (t. no. of concepts), rather than k^C.

Superficially, this would seem to only deal w. t. problem of storage of codes: but not so! — I think that if we have a suitable notation for expressing t. set of codes (exponential in number) in storage that is first degree in C, then we can perhaps devise methods of obtaining new Gops, directly from that notation.

.17 e.g. say $[S^1_i]$, $[S^2_j]$, $[S^3_k]$ are 3 sets of finite strings: There are $n_1 \cdot n_2 \cdot n_3$ such strings.
$i=1/n_1$ $j=1/n_2$ $k=1/n_3$
The notation is meant to be t. set of strings obtained by concatenating 1 from S^1, 1 from S^2 & 1 from S^3.

Using t. expression .17, we obtain the set of conditional operator modifier codes $[S^4_l]$. $l=1/n_4$

One trouble may be that we still have to test $n_1 \cdot n_2 \cdot n_3 \cdot n_4$ different trial GOPs — & this number is exponential in C, t. no. of concepts. → Hvr. [123.01–.22] suggests that because of t. "factored" form of this set of Gops, it will take << $n_1 \cdot n_2 \cdot n_3 \cdot n_4$ trials to test them.

.22 The difty of 121.25 is avoided if we have "perfect" TS's like 53.25, .35, & we always pick t. Gop of max M_i. ("Perfect" means there is only 1 such Gop of max M_i — When this is violated (which would seem to be common) we are in trouble!) A variation on this was that of at 112.11–.21, but I got confused, & I went on to 116.03 to .25 (which was a good idea) which was somewhat different from. Anyway, t. 112.11 idea is something like this:

233A start loading

pretty loading Truck at Johnson Swift

.25 We start t. T.S. at a certain pt. & we get a set of Gops Gop_i, that (solve probs) go distances M_i, resp. At the end of each const. string we devise a new set of Gops that goes some distance $M_{i,j}$. We now look at t. set of numbers $M_i + M_{i,j}$ and we pick out the set of largest nos. & retain them / (this gives us e.g. pairs of concepts.) (Or we might pick just 1 of the set of largest at random & retain only it.). From this furthest point out, we loop to & get another optimum pair (or sets of pairs) of concepts.

Because of the registration (phasing) of every other concept, we may want to do this thing twice: Once starting at t. beginning of t. T.S., t. other starting at t. first concept.

.22 can also be done w. 3 (or more) concepts (instead of just 2) — & we would then do t. TS 3 times or do all 3 of them more or less at t. same time, & discard any of t. 3 that get too far behind in pc, (or in $\frac{\text{no. of probs}}{\text{no. of concepts}}$ = many probs. per concept.) When a seq. is discarded, one starts a new one near where it ended, using one of t. near-by concept points of one of t. other "series" of suitable phase as a starting point.

So, at all times, we only retain a relatively small no. of codes. This "small no." does not ↑ w. C. → 124.31 spec.

.01 : 122.40 : How to [redacted] deal w. the diffty of 121.25 ? A mixture of the ideas of 122.01 & 122.03 : (& perhaps some ideas of 122.22 !). In coding the corpus from one pt. to another, I think of this as being normally done in several ways : say using concepts α_1, α_2 & α_3 or β_1, β_2, β_3 & β_4 ; or α_1 then $\begin{pmatrix}\gamma_1\\ \text{or}\\ \gamma_2\end{pmatrix}$ & ϵ_1.

So we have derbd 4 possbl paths betw. the 2 pts. The gross multiplication of possbl paths derbd in 121.25 - .40 simply does not occur !

I think the reasons are as → (1) Language is devised to reduce this redundancy.

(2) Part of the language is the idea of "factoring" the terms. When we (on 122.17) work on that set of concepts to produce trials, we end up w. the relatively small set of hypc. modifiers, $\{\sum s_2^{q}\}$ w. $l = 1/n_4$ & small is n_4.

The testing of the very large set of $n_1 \cdot n_2 \cdot n_3 \cdot n_4$ Gops isn't really as hard as it looks, because the Gops are in factored form & we really don't have to do so many tests. One way to do this "factorize" is to express the Gops as the sum of Laps ... the idea of β (64.35ff).

There are probably many other ways to "factor" Gops (each method of factoring" is an "alzn") [redacted]

.22 I think I'll have to look at β (& other alzns if possl.) to see if

.01 — .22 [redacted] can be made meaningful !

SM ~ 3,30 AM

5882 12:10A The discn. of .01 — .22 seems rather peculiar ! It suggests that I mite find a solu. to the diffty by watching carefully how humans solve TS's, but it doesn't really tell how !

.28 : 120.40 : ① Some trims on this technique : Say we are going thru the T.S. w. $CB = A$. Then we go thru as many trials as we can w. $CB = A \cdot h(x, y, \cdots) \cdot pc_{x,y}$ — using various values of $x, y, \cdots$. Next time we do $A \leftarrow 2A$ & loop to → Hvr., this $A \leftarrow 2A$ is for the Entire T.S., not just this particular problem region.

② Say we do find a solu. for some γ value (γ problems into the future). Then I think we should spend a total of $cc = A$ on that γ, then continue into $\gamma+1$, $\gamma+2$ ... etc. as far as we can go — since it's likely that this γ^{th} problem is the beginning of a "concept set of probs" (examples for the same concept). When we finish the concept set, then we loop back to [redacted] $\gamma = 0$ & try to solve that problem again.

What I have to do now, is write a v.g. complete review of t. presently contemplated "solns" to T. S. prob., Then go back & see how it works w. my old & with notation-learning T.S., Then see just how I can implement t. ideas of 123.01 — .23 ! Also see if t. way I want t work t. T.S. gives me any ideas on how do deal w. o Then diffys on 107.0 (— .90.

SM. 3:10A

.10: 116.02 107.37 — 5:57P ? : Say a certain / needed concept is not included in t. TS. One way to deal w. This is for TM to devise its own probs. (as subgoals) to learn new (& likely to be relevant) concepts. Ideally, we would like TM to be able to learn such a technique & apply it automatically. Hrr., I suspect that only a NMTM could learn things of that sort. .15 → See 128.01 / 130.32

→ So: T. immediate goal is to devise a MTM that can deal w. T.S's of some diffy. (≡ deviation from ideal T.S. of 53.25).

T. greatest diffy would seem to be missing concepts. T. least al. way to deal w. missing concepts is t in $\langle B(EA)$. This may require trans-human IPC (i.e. > 10^{11} bits/sec.).

Possibly have device in this MTM of devising & "study problems" for itself to approach an (as yet) unsolvable prob.

Also, Be sure to have t β feature (64.35) of alerg. t. Gops.

After studying this relatively elementary MTM, go on to NMTM. See what differences are needed & see just how NMTM could discover "P/Gops" in hours of various kinds.

.31: (122.22) (122.40 spec.) (116.03 −.25) 5·9·82 12:13A A modifn. of 122.22 −.40: The idea of was to see which initial Gops were good, by continued use of each trial Gop for N concepts into t. future (N = 2 or 3 or more). After N concepts we would see which initial Gops had solvd t. most problems. — Those would be retained in memy & sprouted.

.35 An alternate (modifn) → say Gopi has pc = pci.

Then we see how many probs it can solve in cc = A·pci. This "problem solving" includes searching for new concepts if an older Gop fails. Just how this search for new Gops should be done is unclear. Should Lsearch be used? If so, with what $\langle B$'s.?

.01 One possy: At Gopi, pci: Say we solve 5 problems before we fail. We use up

.02 cc = α of CB = A · pci so we have [A · pci − α] ≡ β left.

At this point we sprout a bunch of Gopi,j's: w pc's pci,j. We then do an Lsrch w. CB = β : cc limit β · pci,j for the Gopi,j. This continues

until the CB limit for one of the nested Lsrches is

exceeded.

Then we go to the next Gopi,i+1 & continue like .01.

When the highest level Lsrch is completed, we retain the Gopi's that

solved the most probs.

Some diffys w. the forgg: (1) Hy pci Gopi's get more cc to work

problems, so they tend to be able to work more & it gets retained. Is

this spurious? — should we perhaps use [no. of probs solved / pci] =>

ordering criterion [≡ no. of probs solved / cc allowed] or (pci)^(1/no. of probs solved) = "pc/problem"

.18 (2) How would we deal w. misordered T.S.'s? Could the (apparently v.g.)

method of 118.28 — 120.40 be adapted to work w. it?

(3) Does it really work? Does it work w. the T.S.'s of 53.25, .35?

Will it deal w. the probs of 107.35, .17 (& missordering (#))

.22 Re: .18 perhaps no diffy here! We simply take the [h(x,y,...) · pcx,y · A]

of 119.01 and use that instead of CB = A · pci in 125.02, & continue as

.24 125.02 ff → diffy: 120.26

.25 124.35 ff may be a kind of soln. to the diffy of 107.05 (i.e. All concepts

CB as the most diffl concepts in the previous way of working T.S.'s). This is because

N.B. 124.35 can itself be modified by calling on the method of 118.28

(using h(x,y,...) ect) as a srtn. So 118.28 ff calls on 124.35 as a srtn,

which in turn, calls on 118.28 ff as a srtn, recursively.

We try to go as far as we can past a few concepts, given CB = A · pci. Gopi

Perhaps the forgg. diffy would be better if I used

Gerc = (pc)^(1/no. of probs solved) = pc/per problem [log ... decision] → [b cost / no. of probs solved] = b cost/per problem.

Diffys:

1) hy pci gives perhaps spuriously large no. of probs solved.

2) How to deal w. misordered T.S.'s

40 tracks

20 10 ms/record.

40/3 × 20 ms

5000/3 = 270 ms

+ 100 ms wait = 370 ms.

.139 V / .2 Ω = .6925 amps / ~700 ma

EM 330A

5·10·82 1030P (missed a day).

SN (sort of): In my present T.S. ideas, the T.S. form is fairly sharply prescribed. We just have to get T.M. to search for new concepts that are increments on previous sets of acquired concepts. Diffys arrise when concepts in t. T.S. are not introduced in t. proper order or when intermediate concepts are actually missing, or when finding spurious good concepts tend to prevent one from deeper search to find t. "right concepts".

T. search routines that I've proposed are rather simple (tho apparently very effective in many cases) & don't seem much like t. rather complex heuristic searches that I normally use in probsolving. ⓐ What kinds of problems (if any) need this heur. search? ⓑ Could heur search of this sort be usefully used in t. simple T.S. probs I've been discussing?

Occasionally t. extended search procedure for intermediate concepts may be no longer than just ↑ A or ↑ to "jump over" t. desired intermed. Concept.

SN The h(x,y) p(x,y search of ~ 119.01ff is can sometime be of very hy cc. What it looks for is ~~a concept~~ an intermediate concept do be used to solve for a hyer order concept. Hvr., if t. search of 119.01ff is of too much cc., It may be cheaper to ommit t. intermediate concept & try do dcvr. t. final concept directly -- ·· ob, of course very hy cc. T. method does this automatically.

5 11·82 1:30A

10:50P Gomez

26: 125.24: It would seem that 125.22 is a bit confusd! in h(x, y ···) x is t. "depth" of the Gop trial: it corresponds to unlikelyness of that trial or some other ordering corr of t. trials. Hvr, t. method of 124.35ff is meant to replace this "x" concept ordering concept — so it's not clear how h(x,y··) can be used.

What h(x,y··) means to do is express how an ↑ in x corresponds to an ↑ in y (y is t. no. of probs into t. future).

10:20P SM A possl. (apparently reasonable) idea of what 118.28ff & 125.22 ff could be! In 118.22 ff the GOPx is a setof Gops ob some point in time (actually time, not cc), of one's working t. T.S. Each of these GOPx's has either failed or gone as far as it could in its allotted cc. These GOPx's don't nec'ly end at t. same point (≡ problem). Say φ(x) is t. pt. (problem) at which GOPx first fails (or runs out of cc ··· in which case φ(x,··) is a funct. of A as well as x).

X is an ordering param for t. GoPk's. X might be φ(xboo) i.e. X could be t. no. of probs GoPk has worked. (However, we'd like X to identify GoPk uniquely — s' it might well be that several GoPk's have t. same φ.)

So maybe we use GoPj w. j identifying t. GoP., i φ(j) being t. no. of probs GoPj solves, i X(j) being t. ordering param.

X could also be $(pc_j)^{\frac{1}{pc_j}}$ — t. pc per probe solved by GoPj.

5.12.82 12:01A

$M_{j,i}$ is t. i^{th} sprout of GoPj, s' $pc_{j,i}$ is its pc.

.11 We use $h(x,y)\,pc_{j,i}$ to order our trials, on the prob that is y past t. furthest that GoPj got.

After we do those trials w. CB = $A \cdot h(x,y) \cdot pc_{j,i}$ we find some way to order t. $M_{j,i}$'s: T. ordering could be $(h(x,y) \cdot pc_{j,i})^{\frac{1}{\text{no. of probs worked}}}$

For each trial given j (s.t. X) & y & i we can compute $h(x,y)\,pc_{j,i}$. Then using t. ideas of 12 4.35 ff, we see how many probs we can solve (starting at t. y^{th} problem) using CB $= A \cdot h(x,y) \cdot pc_{j,i}$ with this CB, we go as far as we can go. If we fail before t. CB is reached, then in line w. 125.02 we sprout a new bunch of GOP's & use t. remaining cc on them. This remaining cc. is $\beta = A \cdot h(x,y\cdots)\,pc_{j,i} - \alpha$ T. pc's of the sprouts are $pc_{j,i,k}$. s' we use CB = $\beta \cdot pc_{j,i,k}$ on t. i,j,k sprout.

When we finish, we have all these sequences solved. of probs comming out of t. GoPi's. We order these sequences according to some score, X then loop to .11 5.

However, note that h(x, y ···) can contain info on recent history of t. search — so if there are some earlier unsolved probs. this fact can modify h(x, y ···) so we could go back to try to solve them (y would be <0 in such a case). Earlier unsolved probs will certainly exist if y was >0 for some previous solved problem.

SM 330A

65 chars/line
to lines.
2600 chars/p
2.5 M chars/p.

250
2.5 =
300pp/disc
285.7 100000
350

.01! 12 4.15 SN One way to work diffl. problems is by first working problems of intermediate diffy". (= "study problems"). The devising of such "study probs" can be regarded as a kind of (probabilistic) subgoal.

study probs may be sub-goals

But sub-goals may become crucial.

The study probs themselves are "close" in certain senses, to the real prob. that is to be solved. The major concepts needed for the study probs seem similar or identical to those needed for the real problem. The concepts needed for the study problems satisfy certain constraints — i.e. ① that may help form a soln. to those "real" probs., ② that the study probs themselves are in the real problem. These 2 kinds of constraints serve to define the concepts needed, & ∴ ↑ their pc. However, this "pc" is with CB = ∞. Actually implementing the defns inspired by these 2 types of constraints normally involves a big search — usually a search over candidates that need not all converge.

.20 One way to look at this: Say we are generating various concepts to be used to solve the "real problem". If a subset of these candidate concepts can be combined w. old concepts to form solns. of problems that appear to be similar to the "real" problem, then these candidate concepts are assigned higher pc than they would by narrowly considering only their method of construction.

N.B. This hype

If we have a concept that we obtained w. a very hy pc defn, yet this concept uniquely satisfys a certain hypc constraint, then that concept has the pc of the hy pc constraint.

This obvious fact may be useful in defining & assigning hype's to various concepts (obtained, apparently, w. lo pc. defns.

However the proofs of (uniqueness) may be diffl. It may be unique for a gn. CB — in which case we have to (very truthfully) give the C.B. ← (This may require very little info, since all we need is an upper bnd for the C.B. However, the higher this upper bnd is, the more cc we need to prove it! — i.e. we have to test all cases < that upper bnd. to show uniqueness. — Unless, of course there is some shorter trick than proof to show uniqueness.

.01 In 128.20, if we have the rite concepts in TM at the time, this generation of (new) hype concepts by selection via a hype rule from a (larger) population of objects that was generated via a hype rule will be a natural thing that occurs in conventional Lsrches.

.01 can be written: $\alpha \cap \beta$. α is a generative grammar β is a (selection criterion, selective) grammar (it determines if a candidate is or is not a member of the lang.). Here both α & β are of hype. If "and" is of very hype, then the pc of $\alpha \cap \beta$ is $\leq pc_\alpha \cdot pc_\beta$ — which is my.

I think that the Big Idea of 128.01 ff is that it's an example of how a "heuristic search procedure" can get automatically included in a normal Lsrch. Before this can occur, however, TM must have acquired (on bar gn. abinito) the proper concepts needed. A particularly (desirable, interesting) aspect of this is the "hour search procedure" I've been discussing has no obvious relation to Lsrch.

One impt. concept here seems to be the idea of "Generalized inverse" i.e. Given a algm, M, & a condition C (C(x) = 0 or 1, x is a string. 0 ≡ false, 1 ≡ True) to find the set of strings x_i ⇒ $C(M(x)) = 1$. We may stipulate a CB for the search involved. We could write this as $M^{-1} \cap C$, perhaps. Here "C" is the set of strings, x_i ⇒ $C(x_i) = 1$. M^{-1} is the set of strings x_i produced by all s_j (s_j is a binary number & also a binary string) with $s_j < a$ certain K chosen so that the cc needed to produce the set $\{x_i\}$ is < 1 (K is as large as possl. subject to this constraint). [8 + 3P]

.30 On second thot, this may not be so simple! Intuitively, when one solves a "study problem" one is probly allowed CB = A to do so. However, if we view the concepts used in the "study probs" as "intermediate concepts" in an Lsrch, they will not be allowed CB of as much as A.

.35 Another gen. diffy w.r. to ... disc.: In all of my search algms thus far, we have not been allowed to look at the problem until the trial Gp was constructed. However, in the intuitive pictures of the devising of "study problems", one first looks at the final problem, then one constructs "study probs" that are "in" to it in

.01 certain impt. ways. Quite difrnt. from t. "don't look at t. problem" approach.

A very extreme "look at t. problem" approach is exemplified by t. "large corpus" NMTM problems — like linear regressn. No! see .10 ↓

In "β" where Gop is t. sum of a lot of Lops; each Lop has an ob & Op part: the ob looks at t. problem & selects out a suitable op (if any). — However this is not t. search process, in which (even in β) we generate t. candidate Gops (& Lops) w.o. first looking at t. probs. Even in this "NMTM large corpus" we don't look at t. corpus when we are devising new CPM's (& Rems). (in my present idea of how to do it)

.12 5.13.82 12:36A So, actually, none of t. search methods I've proposed

.13 before [28.01] looks at t. corpus before constructing trials. A search method that does so (& I'm sure humans do it) certainly could save much cc., by not constructing trials that have zero chance of success: also looking at t. problem often suggests methods of solving it — including methods that devise new concepts. (This could also ↑ t. pc of soln a great deal! ↑ maybe not: see 131.01.

→ So .12 really is a general search methodology that I must eventually go into! into!

Rec: 128.01 ff I think what I found so attractive about t. idea was that th. heuristic. I was considering simply (seemed) to do nothing but change t. pc's of various obss. — & yet it yielded a rather common, useful heuristic search method. [In fact, it may have done much more: it had 2 characteristics that would put it outside at least t. kinds of searches I was considering i.e. 129.30 & .35. It is probable that eventually I will use a search method that doesn't have these 2 limitations & so t. idea of 128.01 ff will be o.k.! But, as of now its not a directly usable idea.

.32 SM ~ 330A

.33 8PM N.B. In all of t. discn. of search from ~40.01 on, & trials Gop's were constructed w.o. looking at t. probs. Hvr. in this [illegible] discn. of 40.01 ff, there was no reason why t. Gop's couldn't look at t. probs! As far as I can see, all of t. discn. of 40.01 would continue to be true if t. Gop generation method was able to

{look at t. problem. One impt. effect of this will be to ↑ t. pc of t. soln. tremendously (see .13 ff) e.g. certain trials could be assigned pc. of zero.} — hmm, maybe not — see 131.01

.% Defs call these "blind" v.s. "sighted" search algms.

.01 On second thot, "sighted" search may not change pc at all. i.e. perhaps by defn. pc is apriori (Blind) probability — so Only t. cc of t. search c(l)d be changed in a sighted search. Hvr., this could change effective normzd pc., since for a gn. CB. we could find many more Gops that would solve t. prob.. "Normzn" in this context is over t. consi concepts of CB ≤ A.

Note: Proby distribns assoc. w. a gn. CB (≡ A, say) do give a real CPM. — This is one of Willis' v.g. ideas. such CPM's has simple problistic. properties. The machine assoc. w. a gn. CB. is a FOR (perhaps ---- tho I'm not sure now.... read that report I wrote. ---- it proby has a defn. of a FOR.)

So, during an L srch, one simply uses a seq. of machines w. progressively greater C.B. A, 2A, 4A ....

Willis dealt w. sequencial predn. hvr.: I'm not sure this result is necly applicable to extrapoln. over a set of unorderd objects. (Tho it may well be).

Another point of difrnce betw. Willis result & present problems: I use t. CB as a limit on search cc. I'M not sure Willis "CB" is a genzn. of this (i.e. that this is a special case of Willis "CB").

Oat: $120/kft.
pine $60/kft.

AH: In Organic evoln, all proposed solns are "Blind", but there are ways to give something like "sighted search": e.g. t. Ability to adapt rapidly for certain envt. types., is an example.

— So t. moral is, it is possl. to solve very dift. probs. using Blind Search.

Also, Org evoln. can be usd. for suggestions on how to improve this search.

One difrnce w. Org. Evoln: T. problems are not "solved" or "not solvd" types, but a numerical Gore is gn. for each proposd soln (≡ reprodn. rate). Also, there is much noise in t. system.

There is **No Backtracking** in Org. Ev.

——— SM 2:50A

Some interesting things about org evoln: (1) The nature of t. code for t. operator is (perhaps) a sequence of enzymes, w. perhaps some ordering/info of when they are supposd to work.

(2) The use of sexual mating to produce one kind of mix of gene parts.

(3) Grouping together of synergistic genes in sexual mating, due to t. "crossover mechanism".

(4) " " " genes via t. chromosome topology.

(2), (3) & (4) amount to a kind of conditional probability/inter dependancy among genes, w.o. mutation: its a shuffling & selection out of t. gene pool of sequences of hi geni expected Gors.

(5) Mutations: One kind of noise in t. system. (Another kind of noise is in t. Gore itself).

→ 133.01 spec.

5.13.82 TS

132

Review: This is a review of ~ 40.01 to 131.40.

First a kind of outline:

1) T. nature of t. T.S. being solved: (A Q.A.TM; MTM-type T.S.)
Define all of t. terms: How does such a T.S. differ from other poss. T.S.'s;
All of T.S. is simultly available to TM.

2) Some restrictions on T.S.'s that make them Solvable.
53.25 is hyly restricted T.S. that is easily solvable.
Some kind of related restrictions that will permit them to be solved.

a) 53.35

5.14.82 TAM

b) Allowing some misordering of concepts in t. T.S. (11.6.38 for Bibliography on "8")

c) The desired concept is sometimes not t. "Best" one w.r.t. t. Gore used.

d) T. theorem needed in intermediate concept (for concept, 3 for concepts) is not presented in t. T.S. by probs. of all, so if a direct search is used, t. Needed CB. becomes too large.

β elem. of Gop
"sited" is exclusion of search thg on.

NMTM:
96.08
96.10
85.03
42.01
43.10

Defns:
Gop, Lop
SP
PTT

The kinds of Gops used in t. large T.S. types can be elzd to β.

The kinds of search for concepts allowed can be either "Blind" or (in more advanced TM's), sighted. (i.e. TM can look at t. prob. before searching for new concepts needed to solve it).

β = 69.35 ff

Backtrack:
114.07

The major diffys (107.01-.40) & how some may be overcome.

SM ~ 3 AM

5.16.82 12:01A

The f(cc/pc) concept
p 55

First a discn. of t. T.S of 53.25 & how it is solvd: — which is defnd in ~ 40.01 ff; 39.35-40.08 is t. first strong step. next 43.08 - .40, 45.03 - .15 ... The Phaser early algms were discarded when 53.25 was proposed for t. T.S.

T. T.S. of 53.25: This is t. simplest corpus type treated. T. T.S. is a sequence of problems of t. Q,A type. T.M. is gn. t. entire T.S. which is an ordered set of Q,A pairs. We want to construct an Operator (= GOP ≡ Ground Operator) of maximum pc that can operate on each of t. Q's to obtain t. corresp A.

134.01 spec

5·14·82 TS

.01: space 131.40 : (org evoln) The mechanisms 2,3,4, for shuffling genes, are for small improvements & small risks. The mechanism (5) (Mutation) is for larger jumps & is much more risky. The way t. mutations are ordered so as to ↑ their expected Gorc, is unclear to me at present. Hwr., t. fact that t. genes mutate individually is impt.

.05 Mutation & var. Each gene a problistic distribution in dern. space about its own original dern.

One way that evoln. can accelerate, would be for it to get better & better distribution functs in .05.

5·15·82 2:30A As evolution continues, t. derptn of t. organism gets larger & larger

5PM 3:10A

One of t. characteristics of t. genetic coding method: that each gene site can be varied over a certain set of genes, & almost always t. resultant organism will be rather viable.

We can regard t. adaptation method as having a large jump mechanism in dern space (mutation) & a small jump mechanism ("variation"?) about t. mutation (or lack of mutation) point.

.20 → 141.30

Left ~ 2PM Back ~ 2½ to 3

11:40P Sat Dogs start

12.14A
12.25
12.50
1.14A
1:27A
2:38A

Did I mean diffy ≥ (135.35) ? I think so !

.01 : Th. Diffy 1 (135.32) can be dealt w. by choosing sampling out of t. past

a fixed no. of problems. These would be ~~[illegible]~~

≈ (only 1 / not > 1) prob. from each concept used. Say ~~[illegible]~~ ≤ a

.04 constant, k concepts into t. past. While this method is

not infallible, it will be O.K. for large enough k! One could

select k so that t. cc spent checking t. past was ≈ t.

cc of discovering t. proper Gops.

From 106.01 -.17 ! For solving k probs in t. past,

.12 cc spent solving probs for each concept is ~~[illegible]~~ $(m+k) \cdot w \cdot \bar{N}$

.13 " " or Gop search (per concept) is $A \cdot \bar{N}$

so set $(m+k)\,w = A$ now. $w \leq A p c_i$

so $(m+k)\,w \leq A p c_i \cdot (m+k)$

.19 $\therefore A \leq A p c_i (m+k)$ · or $m+k \geq \frac{1}{pc_i}$ · (m = no. of probs. solved ≡ N of 135.23)

See 137.17 -.25 for a clearer, more correct Analysis.

since we expect $\frac{1}{pc_i}$ to be usually > 1000 or even 10000,

and usually m ~~[illegible]~~ 1000, we expect that k = 1000 or 10,000 would

be quite adequate.

—— So chosing k = 1000 or 10000 may be fine!

Hvr. .19 : $m+k \geq \frac{1}{pc_i}$ seems wrong! I guess what it

means, is that to have $(m+k)w = A$ ~~[illegible]~~, if $w = A p c_i$

then we need $m+k = \frac{1}{pc_i}$.

If $w < A p c_i$ then $m+k$ is $> \frac{1}{pc_i}$ to get

.30 At any rate, if $(m+k) < \frac{1}{pc_i}$, then certainly, t. ~~[illegible]~~ cc

of checking problem solns in t. past is < t. cc of

finding new concepts

SM 320A

5·16·82
7:20 P bowling starts
7:30 P wed!
Summer day: visit
one of Jeff's.

7:30 P ~~[illegible]~~ Re: .30 : If we want to use this

~~[illegible]~~ Algm. for induction of next problem, t. fact that t. present

Gop has worked for t. last 10k problems may be an adequate

assurance of inductive validity SM probly

Note also, that if ... (A rudely interrupted & neat idea!) ∴ we use the A←2A

loop to find an adequate A for t. T.S., Then t. A we use will be t. A that is

adequate for t. concept of largest $\frac{cc}{pc}$. — So in .30 ~~[illegible]~~ $m+k < \frac{1}{pc_i}$

.01 is for ■ pc_i being t. pc of t. concept of max $\frac{cc}{pc}$ in t. entire T.S.

Some confusion here! If we set $m+k = \frac{1}{pc_i}$ · then $w \le A\,pc_i$: $\frac{w}{pc_i} \le A$

so $w(m+k) = \frac{w}{pc_i} \le A$ · ∴ $(M+k)\cdot W \cdot N < A\,N$ ■ 9.V.136.12,13

So cc of checking probs < cc of discovering Gop.

More exactly, if we consider a single Gop: Then for it, (m probs)
$(m+k)\cdot W < A$; I.e. cc to check its past (& present (m probs)
is < cc needed to discover that Gop ($\equiv A$). No!
A is/t. an upper bnd on cc needed to find all of t. Gops at that point.

Gen'l notes:
1) Gop complexity A_i total or Given A.
2) Then $A \leftarrow 2A$ loop.

.17 More exactly! Say pc_i is t. pc for t. i^{th} Gop$_i$ generated at gen. pt.
N is t. no. of Gop's we're forming, that all have t. same max m (m≡no. of probs worlds)
k_i = no. of probs checked into past : so set k_i so that $\overline{N}(m+k_i) = \frac{1}{pc_i}$ ∴ $(m+k_i) = \frac{1}{\overline{N} pc_i}$

Total ~~time~~ cc. checking = $\sum_{i=1}^{\overline{N}} (m+k_i) w_i \le \sum_{i=1}^{\overline{N}} (m+k_i)\, pc_i \cdot A$

$w_i \le A\,pc_i$.

$A \sum_{i=1}^{\overline{N}} \frac{1}{\overline{N} pc_i} \cdot pc_i$

$= A$

So $\sum_{i=1}^{\overline{N}} (m+k_i) w_i \le A$.

Still, $\frac{1}{pc_i}$ can be rather small for many Gops : (say 10 or so:) cc$_i$, hvr, can be very large.

{ say $\frac{1}{pc_i} = 10$, $\overline{N} = 10$ Then $m + k_i = 1$ so $\wedge k_i$ is probly <0 !

.25 So k_i would be ~ 0 (or even less!)!

→ Another possy is to choose $k_i \Rightarrow$ total cc to work $m+k_i$ probs is $\frac{A}{N}$

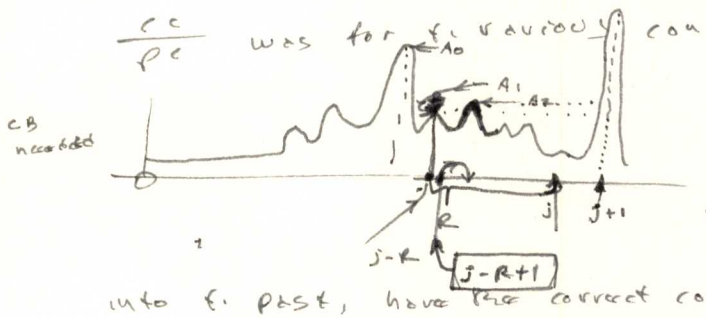
So total to work all probs for t. $\overline{N}$ difunct Gops, is $\overline{N} \cdot \frac{A}{N} = A$.

An impt thing to note is that the $\sum c_i$; A used for almost all Gops is >> they need. T. A used is for t. most intractable Concept in t. **entire T.S.** This means that t cc many to work a prob is usually << $pc_i A$. In addition, t. T.S. is designed so that for t. correct concepts, $\frac{cc_i}{pc_i} <$ some A, but this cc includes both t. construction of t. Gop : t. soln. of t. problem — so t. soln. of t. prob only will usually need much less than $pc_i \cdot A$ to solve it.

(Hvr, In general, I suspect there are prob types where t. cc to solve a prob. is > t. cc needed to construct t. Gop.
— as well as probs in which t. Gop construction takes more cc. →140.19

.01 : : 5.18.82 | 1:15A : A difficult prob! In the T.S. of 53.25: How to do T.S. so that all Gops don't have to have the same A is the most intractable concepts. ("Most intractable" means "hyest least").

Say we have worked up to Gop$_j$ in the T.S. & our needed CB has been A (after various ~~backings up~~ failures, ~~restarts~~ doublings of A & restarts). We can then look back & see how big the cc/pc was for the various concepts that were critical in getting up to Gop$_j$.

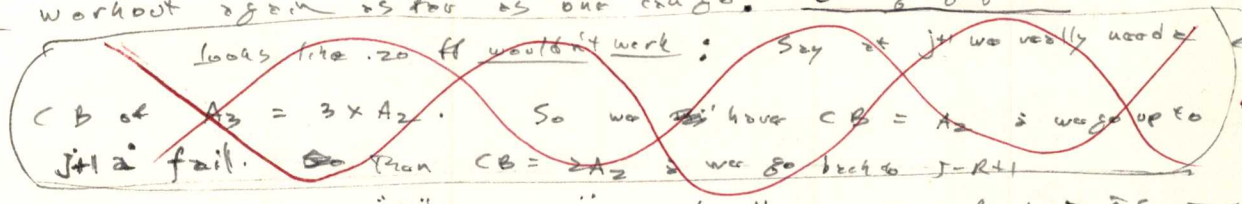


A possl. assumption: That for some value of R, we can safely assume that all problems more than R problems into the past, have the correct concepts.

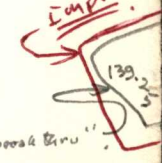
So we assume that either if we've solved up to j, that all the concepts from 0 to j-R are correct. So, if we fail at j w. CB = A, then we do A ⇐ 2A & only go back to j-R.

~~Say for ability at j we had A = A$_0$ & by peak. Then~~

.20 After each peak (like A$_0$) we keep CB at A$_0$ until we get R past it. If there have been no failure, we then reduce the CB to the peak value of the last R problems. We continue this way: value recorded for any problem we initially set CB to the peak over the last R problems. When a failure occurs, we double A, then go back R & workout again as far as one can go. The "re-doubling & going back R" continues until we "break thru".

.26 

on second that, it does seem to work!



I think of "R" as an "interaction" parameter of the T.S. — That concepts that are > R apart don't "interact" — but what I don't know how this applies to .01 ff, & I'm not at all sure it's correct!

.35 This defines what R means

.38 more exactly

The idea may be: That there are 2 (or reasons that a Gop trial can only fail at j. (1) That CB at 0 is too small or (2) That the CB used on searches from j-R to j were too small (i.e. CB at j needed to be much greater than it was)

→ The idea of .35—.38 may perhaps be tested on the ($\mathbb{R}^{12}$ v.s. loop) problem.

SM ~ SAM

.01 : : Another way to think about "R" (cf. 138.01 (-.38). That a newly introduced concept has < R probs to "become necessary". It "becomes necy" when an advanced concept must use it as a subconcept in order to solve a problem.

This (.01) idea of "R" becomes impt. in a related situation; i.e. we may want to drop all concepts from memy that are "entirely" not used within R probs after creation. (* ⇒ they may be ≡ w. .01).

[SN] In a more advanced T.M., (say human), it may keep a concept much longer than "R" if it has reason to believe that it will be useful in the future. The mechanics of how this might be done is unclear to me. Maybe looking at future probs. & seeing that certain concepts would probly be used/in solving them.

[SN] I don't think that "R" here is related to the "k" of 136.04ff — which tells how far into the past we want to check. a/ new Gop.

Perhaps: → Try Algm. 138.01 — .40 w. some/ initial R value/ R0 as necessary & as much as is reasonable. If the T.S. isn't solved, Double R & start again, loop to . Still, we would need a 2 dim. A, R scan — unless we had a aprid for R., then we could do a simpler search. (cf. the 117.26ff using the apripd h(x,y).)

115.18 !

on hierarchy for "theory" & concept; 113.28 – 114.03

.25 – .26 138.26 5.19.82 1:12A After a "BreakThru" the initial CB for the next Gop trial will be the Max Lcost over the last R Gops. (Note that the CB of the "Break Thru Gop" may not be particularly large ... the essential concept needed may have occurred a few Gops earlier & it had large Lcost.).

.25 – .26 completes the decn. of the algm of 138.20 – .26.

~~Next~~ How does the Algm of 138.20 affect the cc of solving a T.S.? If the Algm is not used, the old way gave a total cc of search for concepts of $C \cdot A_{max}$; where C is the no. of concepts in the T.S. & A_{max} is the Lcost of the concept of max Lcost.

My first impression of 138.20 is that this: say we consider A_i [i=1 | c] as the seq. of Lcosts of the seq. of concepts needed to solve the T.S. Let R_{A_i} be a sequence derived from A_i seq, in which each A_i is replaced by the max value of A_j (j = i-R | i).

Then ∴ cc of working t. T.S. is $\sum_{i=1}^{c} {}^{R}A_i'$.

Hm., I think this is wrong. ~~Another~~ ~~an upper~~ My present impression is that an upper bnd for C.C. is $\sum_{i=1}^{c} {}^{2R}A_i'$ & t. true cc is between Phases 2 .

Well! Continue on Review: I really shouldn't spend much time now

perhaps very important idea!

cont. Diffys of 107.01 – .40 . / After Review, Better first "expand" into β , Then look into non-Blind modes of prob. soln — & go back to T. TSPA & algebraic T. S. & see how things go. I suspect that many of t. apparently "impt." diffys shown ~~in~~ don't occur in Real T. S.'s.

5M ~ 3:30 AM

.19 : 137.40 : 10:40P First a brief note: Most reasonable ways to probe into the past & M_i probes into t. future for Gop_i . R is indep of i . T. Q is: How much cc will this take relative to Lsrch for correct concepts?

w_i is mean cc for probs for Gop_i . We retain cont. average, ~~R~~ $\bar{N}$ Gop's per "stopping pt."

So: compare $\underbrace{(k+m_i)\, w_i \cdot \bar{N}}_{\text{cc used for prob solving.}}$ with $\underbrace{A}_{\text{cc used for Lsrch}}$

~~We are really interested in~~ say $k >> \overline{m_i}$ ← average. &

So look at $k\, \bar{w} \cdot \bar{N}$ v.s. A : $w_i < p_{c_i} A$.

.30 w_i is average "in both the" i direction & t. "$\bar{N}$" direction, to obtain $\bar{w}$

$\sum_{\bar{N}\text{ direction}} w_i < A$ $= \sum_{\bar{N}\text{ direction}} p_{c_i} \cdot A < 1 \cdot A$ so Average of w_i in $\bar{N}$ direction is $< \frac{A}{\bar{N}}$. So & $\bar{w}\,\bar{N} < A$

& $k * (\bar{w}\,\bar{N})$ v.s. A is not a very interesting comparison!

<1

is. I suspect that $\bar{p_c}$ is <<1. Just how small, I have no clear idea. (I probably meant "1"

(i.e. This $\bar{p_c}$ is averaged on both $\bar{R}$: (i directions) did I mean "A" here? — or "i"? "i": see .30 ↑ Hm. I think I meant $\bar{N}$ here — But I'm not sure — see 139.01 for ref. to def'n of R.

$\bar{w} < \bar{p_c} A$. Before, I was worried because some p_{c_i}'s were to be larger (i.e. not far from 1) — but clearly, I'm interested only in $\bar{w}$ & averages & ∴ prob'ly $\bar{p_c}$, & averages.

Actually, as & present time, I have no good idea how large $\bar{p_c}$ is. Perhaps Also, I don't know how much of A·pc is used to construct & Gop & how much is used for prob. solving — (when,

.01: in deed, it is possl. to make such a division!).

.02 Another approach (which seems V.G. rite now) is to design t. TM so that this prob. is not so impt. — e.g. in β & related TM types, when one creates a new LOP, it is usually not necy to check it on very many probs of t. past, because usually it is not relevant to many probs in t. past — it's designed mainly to deal w. a new situation.

.09 Similarly w. t. problem of 107.38 of having too many ll codes for t. same Gop, I had hopd to get around this by using suitable languages for TM — so this prob. is less likely to occure... It does seem to occure infreqly in human concious prob. solving.

8:45

Bank account } Motor Vihicle.
Security } Mutual funds ← secc.

SM ← 3 30A

(one day missed!)

.20 5.2■.82 | 1AM : If one starts out w. a language in which .09 is o.k., then the "selection mechanism" mite prevent it from defining concepts that would make many ll codes a problem!

Could not some such similar device work for .02 ?

A possl. example of .20: In my usual way of thinking about linear regn. & Maxm, There are all these ll codes for t. corpus. Hor, there are other ways of dealing w. t. same coding methods that don't use all these ll codes — in fact t. usual conceptual methods of conventional statistics, don't use ll codes at all.

→ Note: In org. evoln. there is no explicite testing of new trials on t. past.
Again, it may well be that for certain kinds of languages & proper selection, this may be adequate for V.G. induction.

SM ← trip to Cambridge for ~3 days.

.30: 133.20 | 52482 | 11:30PM | Another thing in Org Evoln.: The selection mechanism may work in such a way as to select for "hyer order" devices in t. system. I'm not so clear on this — i.e. what it means.

One possl. meaning would be to modify t. L search algm. for Gops. Another mite modify t. method of "solving" t. entire T.S. (An example of a "Soln. method" would be 135.01 – .28). An example of a modfn. of t. L-srch Algs. would be to change t. condl. aprip'd for t. new set of trial Gop's wrt. t. Gop(s) that just faild. Another modfn. mite be to change t. params

Org. Evoln.

5.25.82 | 12.05A of t. Lsrch — or to do it more randomly.

.02 Modifns of Hyer order things in Org. evoln. would be changes in or introduction of t. chromosome mechanism.

.03 — changes in or introdn. of t. crossover mechanism. Hyer order Hyer yet would be

.04 changes in (criteria ways) for which animals select mates. ...

.02 & .03 change t. conditional dependencies of gene selection (from t. gene pool) (rather than modifns. of t. gene pool itself by mutation).

This "hyer order evoln" amounts to Learning in TM_2 (I don't see any mechanism for $TM_1 = TM_2$, hur).

At t. "hyest level" in org. evol., there is t. selection itself, & this does not evolve. (Tho perhaps .04 can be regarded as something like this).

Another "hyer order" kind of Evoln. is t. evoln. of t. genetic code ... ie. which nucleotides were to be used & what strings of them corresponded to which Amino Acids.

→ 8.28.82: Marvin Sugg. that there might be ways for org. evoln. to "Think" or do computations that would speed up t. org. evoln. process. This is, of course, much contrary to most Biological thinking.

I have always considered this possy, but I never really worked out any way in which it might come about.

Some possl. examples of this kind of Meta evolution:

→ 1) Evolution of t. genetic code! ie. which nucleotide strings signify what amino acids.

2) Development of sexual reproduction.

3) Evolution of societies (in social organisms).

— both of these are impt. changes in t. evolutionary mechanism itself. T. second one had (org. evolution itself.

Evoln. aims for a certain "Top goal" (≈ Max. fertile reprodn. rate)

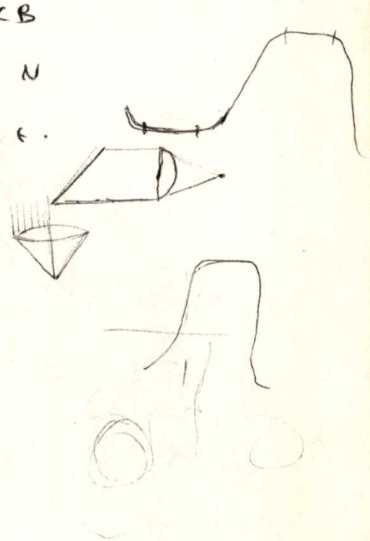
Any way sub-goals can be introduced? — When Man breeds cows, we have coevoln. betw cows & Man, so hy Milk production in cows becomes a sub-goal for t. survival of cows,

One way of looking at t. T.S. problem: Gn. a particular T.S., what is t. smallest cc ost with which it can be solved? It becomes a probe in computational complexity. How do simple changes in t. T.S. (degradations of its "perfection") modify t. necy cc to solve it?

A "perfect" T.S. — takes cc of its length. Also to / total cc is easy to compute — just t. Σ of t. c_{ij}'s of each of its problem sets. ← Don't think so!

A (perhaps) related problem: to devise a T.S. to teach an animal a particular thing w. minimal cc. Here cc may be many times only. (perhaps add in costs of reenforcers? — probly not).

.20 This would be true if TM knew enuf to stop searching after t. desired soln. was obtaind. However, in t. search methods i've been suggesting, this usually doesn't occure, & TM usually searches well past t. "minimal" point. In one search method, TM uses t. same CB for each problem (or each "problem set") — so if there are N $\binom{\text{probsets}}{\text{probs}}$'s, it will take a total cc of ~ N·C ; where C is t. cost of t. most intractable $\binom{\text{prob. set}}{\text{prob}}$.



8.3.82 TS: Bochner: comments on his proposal.

I had been suggesting t. pc of t. subject, S, as being a measure of its degree of order (v.s. Disorder).

Actually not such a good measure. Something more to the point would be "t. shortest code for S", but "t. shortest code that S has found thus far for t. data that it has organized."

Similarly, instead of conditional entropy, we say we know t. code(s) that S has now for t. data it has been subject to. Given a problem, P, and a certain soln. to that problem, X_i, & a certain decrn. of that soln., $D_j(X_i)$. — Then, using S's codes up to now, we can determine t. cost of $D_j(X_i)$ w.r.t. those codes, & also t. cost of testing $D_j(X_i)$.

$D_j(X_i)$ will be t. Decrn. of X_i w.r.t. t. codes now within S.

So $2^{-l(D_j)}$ will be ~ t. pc of that soln. Decrn., and so $w(D_j) \cdot 2^{l(D_j)}$ will be t. cost of that soln.

There may be other decrns of X_i w. diff. $w(D_k)$'s $w(D_k)$ — Hvr. L search will list us those solns in order of $w(D_k) \cdot 2^{(D_k)}$.

Another look at things: Consider t. set of solns. that S has found, for all of t. problems $[P_i]$, that it has worked on thus far. Each of those solns has its assoc. search — & a cc & pc for that search & an L cost for that search. (We include as part of t. search for a soln., "introspection" that occurs after S has stated his soln. as output) that can result in better solns. — also t. "introspection" can used later abstractions on earlier problems, to obtain better solns. ("Better" means uniformly lower pc for t. same or smaller cc. or lower cc for t. same or lower pc.).

Somehow, this set of solns., (including any regularities found in this set) can be used to make a decrn, D_j of a possl. soln X_i of a gen. problem, P.

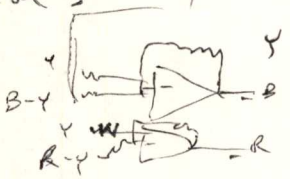
N.B., hvr., solns found in this way are "Blind" — apparently

obtained w.o. "looking at" the problem, P. [redacted] for this reason, the results obtained will be usually poorer than were its obtained when the soln. algm. is allowed to look at the problem P.

(This is like/ the Blindness of systolic evolution algms. — The system can look at the past but can't "see" the present problem)

Alternatively, at a higher level, TM searches for an algm. that is able to look at a problem & find (often by some kind of search) a soln. to that problem.

Is this a TM_2 ?



B-Y+R=G
B-Y+Y=B
R-Y+Y=R
(Y-B)-R
(R-(Y-B))
B-Y+R

On the other hand, I'm not so sure that all of the TM's that I'd been considering recently were unable to "look at the present problem". I remember using an ob, op formalism, in which the obs would look at the problem, & each would decide whether it could solve that problem. The GOP (≡ Ground Op) consisted of the totality of these Little op (LOPS) in U.

At the end of the last attack on TS (5.25.82) I had a general plan for a simple TM! The concept of a "perfect" T, then corruptions of this perfection (by too few (or omitted) examples of certain kinds &/o partial misordering of the examples)) & the problem of how much the total cc needed by TM is ↑ by each kind of corruption.

This gives a useable kind of T.M. I could then approach the problems of improving it in various ways! Eg. use of a TM_2 & ultimately, $TM_1 = TM_2$. Also, the Q of how to allow "TM to" look at the prob. it has to solve" rather than simply make the some rote search for a soln.

Review of t. review: <This is to give me an understanding of what has been done thus far>

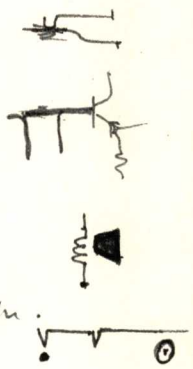
P. 134 (Def of t. T.S. of §3.35)

P. 135 [redacted] (A discn of t. problem of solving this T.S.)

A discn of t. soln. I've been contemplating. Also an impt. apparent Bugs in that "soln".

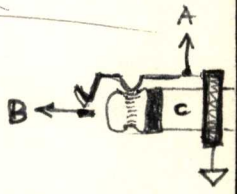
135.32 & .35 list & diffys of this soln. 121.25 ff. discusses at some length some possl. approaches to t. diffy of 135.32.

Also 136.01 ff discusses it's difficulty ways: also 146.30 discusses a possl. soln.



A "trouble": Whenever I discrib. any diffy & so that I really understand t. diffy, I begin to get ideas on how to solve it ... so I end up working on t. problems & not getting [redacted] on understanding of t. [redacted] state of t. entire T.S. problem.

SN It may well be that using t. shortest code for a corpus is ≈ using Pm & t. "Σ" of all possl. codes of t. corpus: [redacted] ([redacted] a proof of ≈ this by [redacted] P. Gacs) — but it was don't used t. shortest code but a/ fairly short code instead, then I suspect that using many codes in ll is [redacted] quantity better than using t. best of many short codes.



The truth of this would [redacted] depend on t. distribn. of code lengths. Gac's theorem tells us somethng about this distribn. — Does it tell us enuf? Also, maybe Gac's proof has more info about t. distribution.

Anyway: Re: t. Problems of 135.32, .35; exponential storage problem.

..30 135.32: (see 121.25 & .30 for 2 aspects of it). The exponential storage prob. might be dealt w. this way! Consider t. last 3 "stoping pts". T. first yielded R_i different solns. The i^{th} of these yielded R_{ij} / accepted solns at t. next stopping pt. The j^{th} of these yielded R_{ijj} accepted solns at t. 3rd stopping pt.

Tl. total no. of solns. obtained for t. last 3 stopping pts is "Like" $\sum_{i=1}^{R_1} \sum_{j=1}^{R_i} R_{ij}^3$. We simply retain those of these that U = 1000, the means, say, of these at max pt. We use whatever value of U that is reasonable.

More simply, say, at a certain stopping pt., we have → 147.17 spec.

The problem of 121.25 (excessively low expected pc's) somehow doesn't seem so impt. The pc's of solns ↓ by a fair factor of what they should be, but we automatically compensate (to target (?) extent) for it by using larger values of "A". "A" (of 135.10) is doubled if one starts over at the beginning, if one runs into an example that one can't solve w.r.t. current value of A. This particular "doubling A" algm. is discussed in various places & modifns. of it were considered.

The problem of 135.35 (the cc → c² difficulty) doesn't seem to be bad: we could consider only a finite no. of examples in the past — algorithmically selected or randomly selected, or some other way. Other solns. have been suggested.

.17 : 146.40: R_0 distinct solns w.r.t. same (max) N_1 value. For the next stopping point, we "sprout" these R_0 solns. (each w. its own pc_i) and we obtain R_1 "Next" all of equal max N_2 values new solns. Say the i'th first soln. has n_i second solns of pc pc_{ij}. Then we retain the U solns. of max $pc_i \cdot pc_{ij}$ — we retain the top R_0 of them, say.

We may want to retain data of sequences of 3 stopping pts, rather than just 2. or even more stopping pts.

N.B. One of the reasons for using the particular kind of TS, & particular kind of soln. that I have here, is to solve the problem I ran into w. "Eval": I had a choice of 2 kinds of Algm:

(1) Do substitution xfm N times; T. pcost was $\propto N$, but I could start out w. small N & ↑ N incrementally as the probs got harder.

(2) Use a stop rule to tell when to stop substituting. This is a "loop soln". It has lower pc than the rules of (1), for small N, but it's by lower pc, if N is large.

However. The incremental pc of (1) is always very small for a suitable T.S., so one would, in such a case, never devr. (2).

The T.S. of 53.25 & the "soln" of 135.01 does, I think, & give one the soln. (2), & this soln does so in a way that [illegible] in a desirable way.

.01 An impt. Q is: From t. simple Q-A machine of 134.01–135.40 (w. its particular soln), how does one get to a machine that can work diffl. probs?

Some immediate refinements are: ability to deal w. "low" "few" examples of various concepts & misordered problems. (see 107.01–.40 for various diffys & enhancements of t. system). → see 133 The Bmethod (64.35 ff) & 107.24 are in t. directions of strong enhancement.

Another possy. is in t. direction of TM2 now! Here we give as "Q" an entire problem, & t. corresp. "A" is a method for solving that prob. (& presumably) similar probs. One way of looking at it would regard its present prob. solving [redacted] algm as looking for a "method" — since it looks for an operator that relates Ai to Qi.

However, what I'm thinking of seems signifly different. What I want to do is collect a bunch of prob. solving methods — then arrange them in some sort of order so they are like a T.S., with successive methods building on previous methods. Then I'll use this as a T.S. for a suitable T.M.

.27 IMPT → Another Motivation for [redacted] working on & perfecting t. models of 134.01–135.40: I feel that this idea of trying algms. "close" to [redacted] previously successful algms is a good idea & that somehow, it's t. only way to solve any problem. For this reason, I want to use this as a "study problem" - to familiarize myself w. t. various concepts needed, before I use this [redacted] idea to solve more general problem types. (→ Also 149.29) .32 → 149.01

This seems very like [redacted] is it Lenat's Conjecture

.33 Another good direction to take 134.01–135.40 in: Dealing w. TS's with poorly ordered probs in t. latter part of t. sequence. This approaches the human situation in which one looks for probs. that one can solve that are likely to lead to new, useful concepts — in particular, new concepts that one feels are likely to be needed in other parts of t. T.S.

.01: 148.32: Any probistic info about prob. solving previous probs. — on methodology, etc — should be used to help solve t. present problem — & if it's properly used, contributes to t. ↑ in pc of t. soln. — & presumably to t. ↓ of cc of t. soln. → Hvr., I'm not really on very firm ground on t. "↓ of cc" part — tho, presumably, it's very imp! [∴ ↓ of cc of soln.]

.08 What would be worth while: To consider various Prob. Solving methods & show how they are examples of .01. ← This is a rather general case!!

.10 e.g. Say one has found that a certain kind of search procedure, β is v.g. for problems of "type α". So one has, as part of a "plan" — "whenever a prob of type α occurs, one trys β on it." (or, more exactly, the probistic version of this). (Note that any "plan" is a special kind of <very common> Coding method),

Viewed in this (.10) way, any search procedure that one uses can be justified on the basis of "code compaction". If one thinks, say β is good for probs. of type α, one thinks this because of statistical (analysis studies) or logical analysis on a mixture of t. 2.

Now "logical analysis" is, I think, a special case of "statistical study" — at any rate, it will result in legitimate code compaction. And of course statistical (analysis studies) does lead to proper ↑ in pc of result. Partial ↑ in t. direction of a Well! This looks v.g.! It's a sort of vindication and "proof" of Levin's conjecture: i.e. that Lsrch is about t. best way one can go about solving math probs.

.29 The weakest part is how cc is involved. Perhaps by I'm really not very certain about just how it works out how good any of my systems are in this respect.

studying t. simple ≈ "Perfect T.S.", I can get some ideas on this.

— This is in line w. 148.27 ... That t. work of 134 – 135 is a "study problem".

.32

8.13.82 There is a somewhat different way of looking at t. problem of 148.01, (but I'm not sure that this is a good way!): Say we have an algm, α, that we have evolved & it's successful in solving many probs.

.33 We apply it to a new prob. & it doesn't seem to work ("seem" because it need not have a "hard" failure ... it may simply find no soln. w/ acceptable cc.). When this occurs, we try to solve t. problem using "nearby" modifns of α (modifns that

are "close" to α in description space. → see 015

A way of looking at 148.27.32; 149.01 — .32 that I like more, is to observe any problem solving session of a human & try to express it as a simple search among t. * possl. solns. in order of pc. (or inorder of Least). As soon as a soln. is found, we then try to update t. pc's of all old data. that have been affected, & we devise whatever new data. that seem useful

One of t. big modifications for t. present T.S. work was to see if this could be worked out in a reasonable way. Perhaps if I am able to do this O.K., then CBI is, indeed, an adequate soln. to t. "knowledge representation problem".

.15 One feature of t. search relevant to 149.30: That we divide our cc betw. ① spending time on an existing prob.solving algorithm that haven't yet converged & ② Searching for new PS. Algms. This can be done in t usual Lisch way by stopping work on a prob.solving method or an attempt to improve one, when $\frac{cc}{pc} > A$ ..... then double A. when one is done w. that "Round".

8.1582 TS

P. Bieree is concerned w. "How Data is ~~modified~~ transformed into knowledge."

My impressn. of "what knowledge" is:

It's data that has been put into more usable form. I.e.

~~many of its regs~~ (1) some of its regularities are indicated (or coded out) by, say, compressive coding (2) The form of t. ~~data~~ coding makes it easy to use: for instance, part of it might be put into the form of a table (e.g. a log of sine table) if that form is less expensive to use than the compact pgms for calculating log or sine.

This idea seems important. One function of a heuristic is to convert data into ~~as~~ maximally usable info.

A simple kind of illustration: we have a pgm. for $\ln$. we can make it more usable by making a table (if we're interested in speed). The no. of table entries (= accuracy) depends on the expected use of the table. If we expect use of such a table for several problems, we put the table in RAM, solve the problems, then use the RAM space for something else.

What we have to consider is the ~~computing~~ computing cost of the entire operation — i.e. cost of computing table + cost of renting RAM for table for the time that it's used.

Contrast this with the cost of storing the pgm. to compute $\ln$, & the cost of computing each value as it is used. In some cases, the raw computing time (not necessarily cost) is prohibitive, so we have to use something like a table.

I guess what a TM should do is put the data in a form that is v.g. (in minimal cc sense) for solving the problems of the past (i.e. expected future probs).

35 Say P_0 is a pgm (= Machine) that has been used to solve probs in the past. Then we use $M'(P_0, X_1) \to P_1$ to make a trial modifn of P_0. M' is a universal machine $\to$ $\forall P_0$, $M'(P_0, \Lambda) = P_0$ — the null modifn leaves P_0 invariant.

This, however, is a "blind search" X_1 is not in any way

sensitive to how P_0 was inadequate.

In the case of Perfect Training Seqs., I did consider modifns of t. kind search method; — I think t. "B" (64.35ff) method is one possy.

One thing I have to keep in mind ... a thing that makes t. searches 151.35ff more likely to be reasonable: that in deciding what is "close" to P_0 (using $M'(P_0, x_1)$) one uses info about what sorts of modifns. have, in t. past, helped. — Such info ↑ t. p.c. of things a human, say, would ordinarily try. (This is t. idea of 149.01 .... but that is for a slightly narrower kind of problem).

Also note 149.08: consider prob. solving methods as heuristics — (i.e. consider pc as well as pc,) and ... show how these methods can be made special cases of Lsrch.
Note that we are applying Lsrch to t. problem of finding better prob. solving methods. α_i

---

.22 So t. main thing is this: To develop Perf. Trng Seqs & modifns of them, so I'm familiar w. t. technical quantitative details.

Then, try to put 2 kinds of problems into Lsrch form:

.30 1) Ordinary yes/no search problems à many Math probs w. Black/white answers.

.32 2) Hill climbing (optzn) probs.

The idea of (2) of course is $TM_1 = TM_2$.

The main idea of .30, .32 is that anything a human uses to guide his search in either (1) or 2), can be regarded as part of a "Plan" w. probabilistic values for various alternatives, & as such, is expressable in t. form of an Lsrch.

Perhaps t. idea of 152.22 ff can be summarized by Levin's conjecture — ⟨that Lsrch is ≈ t. best way to solve any problem⟩.

So: In working a T.S., we have 2 problems: ① How to go about t. search for a soln. to a new prob., given t. updated state of knowledge obtained by solving all previous problems ② How one updates t. knowl. base, after solving (or not solving) a problem.

Clearly t. soln. to ② depends on what one uses as a soln. to ① — so, to start off assume ① is ~~[illegible]~~ Lsrch. So ~~how~~ for ② t. Q becomes, how does one summarize t. info obtained from a soln. so that all of t. info is used by a standard Lsrch?

Note that info obtained in searching for a soln. can be info about CC as well as PC. — i.e. tricks to save cc are ≈ as impt as tricks to ~~f~~ find new short codes.

.21 Introduction of Heuristics (cc reduction devices) into solns of T.S's: ~~[illegible]~~ If we have a Kusp machine (or better yet, a Levin — Gacs ~~[illegible]~~ ized Kusp machine), t. soln. to a problem could contain a ~~[illegible]~~ set of specs for how to reorganize t. hardware of t. machine so it could solve t. problem w. lower cc.

.28 An ~~apparent~~ difty: ~~TO~~ ~~get~~ an increase in speed of a factor of 32, one couldn't use > 5 bits! Or putting it another way, if t. dscn. of t. device to ↑ speed is of length 5, then it must be for a speed ↑ of > 32. ~~[illegible]~~ — If this is t. whole story, then it seems wholly inadequate, & unlikely that such improvements would occur!

— So t. Q. is, how heurs. of the desired sort could be introduced into t. soln. to t. T.S.

.38 A possl. way out of .28: that instructions on how to ↑/purge speed of system must be of low b cost — e.g. 1 bit or less. — maybe even zero bits. This can

8.19.82 JS

occurs if t. proposed improvement can be deduced logically or almost logically.

Hvr. 153.28 ff seems not to be t. way people figure out ways to speed up t. system.

On t. other hand, it would seem that ~~writing~~ computational explorations costing avg fixed cc = Co that result in ~~[redacted]~~ a general halving of past & future cc's, is worth while! So, in view of 153.28 we have a serious paradox!

8.23.82 : So: How to devise good (or even acceptable) search methods for techniques of cc reduction?

A cc reduction method could take t. form of a "PLAN".

Is there a way in which [redacted] t. search would automatically go thru L-true operations resulting in a "plan" that must (logically) work at ≈ zero cost? Para "logical reasoning" of any kind should have ≈ zero cost.

.18

SN Title of Paper: 3 aspects of the Unconscious: or Zen & the Art of Scientific discovery.

→ 155.01

SN Use of perfect Training Sequs. for Radio Communication w. extra terrestrials. (The "Lincos" problem) "Lincos"

Look at Lincos: Is it a PTS? Also [Hans Freudenthal] assumes that t. E.T. would end up w. models ~ to those of human "R.W." .... unlikely in view of Bierre's "Grand hypoth" or what "Primary Theorem": Hvr. perhaps it doesn't make any difrnce — as long as communication is via radio, and t. E.T.'s use good induction methods, their replies will tend to be correct. T. Q is ~ to that of 2 people conversing: The models in t. 2 minds may be difrnt., but their verbalized conclusions will tend to be t. same if both have good inductive models.

I think Freudenthal that that t. problem became "only Academic" if pictures were allowed. I don't see why.

8·30·62 PS

Note that TM_2 is a H.C. problem.