4TM.

Last page = End of 4TM

(special optzn Methods for induction: Some generaln of Boon Seq.)
(Doing Phase 2 first (perhaps never do Phase 1!))

0:324.40 : This list is a kind of "Hill climbing" peculiar to induction modeling (≡ compression).
Note that 324.38-.40 is quite compatible w. 324.31-.37 ; These Optzn Methods can recognize induction problems & kinds of induction problems as part of the hierarchy "context" (or "external context") of those problems.

PSM's

t: 323.19 : @Plan: (cont)! Phase 2 has (expandable) set of (prob. Solving Methods).

35 An alternative Plan is (324.31-.37) to do Phase 2 to start off! — i.e. no phase 1 at all!

D29.04 So: List various approaches w/ little detail, then go back & put in more info: (possibly more "BRANCHES")

37 [SN] Consider "TRUE Lsrch" for ($\frac{d}{s}$)-induction in "Phase 1" : This would essentially be same as starting in Phase 2 (324.31-.37). This "Covers" all possl. ways that I could try to do "Phase 1"!

0 Hvr, while starting off in "Phase 2" does look promising, it is a new direction, & I want to list the normal, "regular" ways for the IAP talk. Actually, listing many good ways to do Phase 1 is a good initial approach to "Initial Phase 2",

My impression: That one of Major Interesting problems in "Phase 1" is doing TSQ's — easier for TSQs in d-induction; harder & more interesting for S-induction.

Perhaps I've been thinking about S-induction in a too abstract way! Considering special problems would probly suggest specific solns. & Genzn of those specific Solns.

0 (SN) d-induction & INV probs conclude everyday diffy: Can some/all S-ind probs be expressed as d-induction &/or INV probs? The 3 long representation seems to be better — which is one of foremost pol claims (cf. order is University)

1.2.05 One item 3 long is that the shortest R code for a Ai is usually not very distant in pc from the A1 pc of that Ai. So S-induction is a lot like d-induction.

We might say that in d-induction we need a single O^j w. min pc that works for all Q's.
In S-induction, we want a O^j ∋ its pc plus the sum of its "R;bc's for the QA;s is a min.

For this S-induction form, the search routine has to be worked out. We can get suggestions from the d-induction routine. We should consider both regular Lsrch TSQ's & the "problem pool" TSQ: Also consider BU search as well as Lsrch.

→ So first outline d-induction for the various TSQ & Srch types;

31 T. corpus for O^j induction (in PPMS) can be the set of successful O^j's of past, weighted by
.32 the no. of probs solved by each (or total "importance" of the problems solved).

In "simple" Lsrch, we just hunt for a O^j that will solve all probs up to Qn. Using the corpus of .31-.32.
As each problem comes in, we do a new srch w. a slightly different O^j corpus.

If the s-ind Lsrch corresponding to .30 ff : The problem is, how to weight the past corpus of past O^j's. ... How has been known the entire set up to now (≡ "n" probs)
We could try various past O^j's on complete QA corpus up to n, to give them wts. I imagine that most of them would have very small wts, since they couldn't work many of the problems w. any reasonable pc.!

→ 5TM 1.00

4TM

"MCT"

.10 : spec 323.40 ! The basis of t. pd in Normal OZ Lsrch, is not clear. T. great breakthru of t. Mixd Corpus Form" was that we would do well by trying to find a d.f. for (OZ method, problem dcrn pairs) : giving t. P.D. of solns (cc or "t" was included in t. problem dcrn). Next breakthru was t. WON soln — T. simple form.

In Idsia report: [struck out] T. idea of Phase 1 was to be good enuf at s-induction to Go to phase 2. This early Goal was to be obtaind by any means posbl. — i.e. no need to stick to Lsrch. — Tho I think I considered (mainly?) "≈ Lsrch" over part of a s-funct dcrn, & something "≈" like "Edit dist" for t. rest of t. pc evaln. "Augmented GP" using "PPM" was another direction.

Perhaps my "error" in ~ 3TM 137,138 ff. (→319.00–323), was not to realize that Lsrch was not to be applied "exactly" to finding s-induction solns! — That is, in general, searching for solns to s-induction problems was dif'lt, & perhaps had few features to make it easier

.15 than General OZ problems. (.155) → ONE impt feature: If solns of hy pc exist for

.16 entire TSQ, then listing trial cand O^j's in pc order is an acceptable technique.

It becomes ≈ an INV problem (But INV probs aren't Lsearchd in that way either!).

Consider .155–.16 .... is it true? What is expected cc. of soln? Consider O^j to be a

.20 3 Time. Consider Phase 1, s-ind. tsqs. If we do a complete backtrack from t. n B QA, we search for O^j & th n R(end)'s — which can be a very small total pc! Hvr, what is t. actual cc of finding soln?

I also considered PPM for t. "Adaptive" part of adaptive Lsrch — for both INV & IND problems. At this point t. main difference betw Lsrch & G.P. was that G.P. used Bsrch & Lsrch used Lsrch. (G.P. was Monte Carlo; Lsrch was not (well, Lsrch could be Mt Carlo, but when it is, it is quite difrnt from MtCarlo in G.P.). Both are guided by a P.d. that uses PM.

→ "Lsrch" lists O^j's in pc order (pc's of O^j's) & tsts them on corpus.

→ G.P. srch Creates a subcorpus of O^j's W. GS > X_0 (X_0 ≈ $\frac{O_c}{\text{G.....}}$ is Max). Then Mt Carlo

.30 Selection from that subcorpus. Corpus is continuously (or periodicly) updated wrt PPM & X_0.

Lsrch works well if there is a good O^j of very hy pc that solves t. TSQ.

.31 SN I had been thinking of Doing Phase 2, when Phase 1 had [struck out] done well on problems that Phase II's meta controller ("TM₂") would find familiar. i.e. when TM was open working on Phase 2's Meta problem.

Would it be possi. to start TM out by having it in params of a program that generated a bunch of OZ solvers that were known to be ok? (I would give these solvers to TM

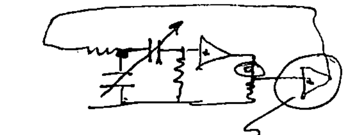
.37 in "factored" form. [This is a REALY SEPARATE Approach to TM] One BCFQ! How much "Logical Reasoning" would be need for "Phase 2" problems,

.38 [struck out] SN In t. forgoing pp. I haven't really mentioned some optzn methods that are very special for induction; eg. ① finding contexts that have short dcrns & are good for induction.

.40 ② looking for rangs in a corpus, then recoding it (& rewriting it) wrt that new rang [= "Coding & recoding"]

D.26.04

4 TM



(See 322.40) Must ask what the pc. of the corpus is, if we use a certain m & a certain width of dist. We want to make pc of "code" of corpus. — So the big Q is: Do we get more compression than w. "straight" PPM? Remember we must be coding only useful Lisp-like pgms. This method may work for pgms like those but not for English Text.

There is another possy!: That it may not compress (is) well, but that this is a good way to generate cands for my $\frac{\delta_c}{G_{max} - \bar{G} \in \Sigma \mu_c}$. ← So this should perhaps be criterion for choosing m & distribn. width as a function of m. If the criterion is, indeed identical to Max corpus compression — Then fine!

So: There are 2 directions of improvement of the PPM. Trick(s)!

1) Elaboration of "Context" to be sub-trees rather than just RPN or PN strings of symbols. This is not easy (& apparently) not cheap.

2) Using partial subtrees or entire branches or just a few symbols of subtree — for "Prediction" (or more exactly Monte Carlo cand generation). — Also what assoc Prob distribn "width" to use. The concept of "width" may have to be modified if we use contexts that are not simple strings.

OK: List various paths to TM: Give difftys in each, refs to attempts to fix them.

0) Plan: phase 1 is d-induction then s-induction: This is closely subgoal to enable phase 2, which concerns the probs for phase 1 & phase 2 (325.04 →) (325.04-05 →)

1) Lsrch on a d-Ind TSQ in Math. →
The R(ecn) Functs may not be needed. Checking on Past Corpus can be: "Hardest probs first"

~~[illegible]~~

Writing usable TSQ's may be major problem. — But may be Easy if large CDS is allowed.

≈ Lisp can be used (≈ AZ).

Possy of TSQ based on Lrg to do part of what Maple/Mathematica do (But note)

> See 320.28: It may be that Lsrch on d-Ind for Math TSQ will not be much good w.o. ability to do Logical Reasoning.. Also Note (320.28-30)

2) "A TSQ for S-Inducting Functs" is a major jump from (1): Some difftys:

a) Choice of s-funct. representations: (Probly use several).

b) How to use d-function capabilities to help w. s-funct lrng. (The "transition" from d to s lrng).

One way is "EDIT Distance". Its not "Univl" tho.

Also note: d-Induction is ≈ INV Lsrch. s-ind is a OZ Lsrch, which is really difft!

In an Lsrch OZ problem, one searches over Optzn. Methods, rather than trial solns. — Much of my recent discussion of Lsrch for OZ problems is wrong! (But not really irrelevant!)

Anyway: This makes true Lsrch over Optzn. routines look like a strictly Phase 2 problem!

My recent (& older) discussions of GP for TSQs had little to do w. Lsrch! It uses a srch method related to Lsrch.

In the list of trials of Optzn Methods: ≈ "Lsrch" would be one simple method.

Perhaps GP would be another, ANN another — But there are really an enormous no. of very heterogeneous Optzn techniques.

OZ

Actually, Phase 2 doesn't use Lsrch. Lsrch simply lists OZ methods in their pc order — The arguments are not considered.

324.00 →

4 TM

0: 321.40 Way to do 321.36 ff: From M, chose random pts, not merely on defent corpi. Keep list of contexts found at each point. These contexts can be of varied complexity: They can be simply Suffixes (as in PPM) or can contain branch, subtree contexts of various sizes. Anyway, when one has generated lots of contexts this way — find way to tell if 2 contexts are "the same" or "the same to degree d" (d = 1,2,3,....∞). Then exchange branches at the 2 pts w. same contexts.

8 **The** PPM version of [illegible] .00 ff: Say we are using PPM to generate prob. dist. Cands w. assoc. pc's at a certain point, we have a known set of contexts (suffixes). For each suffix, we have a pd over the single symbol that will follow — Hvr. by looking at the various cands that have been indexed, one can also get a pd over the complete branches that might follow. Well "pd" the term is perhaps not so appropriate — since each branch will usually occur but once. Hvr, each cand w. a particular context, will have a complete branch at that point — & one could just chose from at random w. a likelyhood. Having done so, we may have finished our cand devn. or not. If we are not finished, we simply continue as before the branch was selected. Selection of a branch amounts to selection of a Terminal Symbol.

15 To be determined in (08-.15): ① How much context to use? One way: In PPM one can end up w. a specific context to use: (partly on basis of consistency of predns). One can then use that same context & simply randomly pick a branch from past cand pts w. that context. This is by no means the only possl. way to do this, hvr.

② How frequently to chose branches in this way? One extreme is to make all choices this way. Another possy. is to make the decision random w. a pc to be adjusted so as to maximize $\left[\frac{\sigma_G}{G_{max} - M_G}\right]$ of Generated Cands. How to do this is unclear!

0: 0.26.04 A Trick related to (.08 ... to .08): In PPM we have this set of strings in alphabetical "suffix" "context" "postfix" order. For a "normal" Mt Carlo PPM "pseudo", in generating a Cand., we go to the point in the Bw xfm having the largest prefix common to the "Cand thus far", Then using a prob distribution of a certain width, we chose a point (corresponding to a pt. in the entire corpus) up or down stream, & we use the simple character following the context at the corpus member at that pt. Instead of selecting a single character, we could (as .08 ff), select the branch of the corresp cand., using a certain pd. width. or, we could select a seq. of m characters, & use a distribution width that was a fnct of m, so for larger m, we use a narrower distribution.

Remember that the goal of the Mc Carlo technique is nominally prediction — so we always (see 323.00)

4TM

31.29 - .40
31.50 ....

?? - It can almost always get relevant context by using a shorter context.

0: gets "off track" if becomes very likely that it will have to use the contexts of cands of lower pc than $\tilde{O}^j$. — Since $\tilde{O}^j$ will not have these contexts. So the usual cands will tend to be from $\tilde{O}^j$'s cands allow $\tilde{O}^j$ correctly, until it "misses" a symbol - ... & then begins to use pc's from the rest of the corpus

3 of cands ..... → (Note 5TM.3.00 - 01) → 5TM3.00

0 : On PPM ... or S. Gramm based GP, v.s. regular G.P. : The usual argument for S-Gramms that is supposed to be v.s. G.P., is that while GP Crossovers transport useful subtrees to offsprings, ~~Many~~ fast Crossovers just as often, Destroy useful subtrees. I don't see this: It might be true of one-always destroyed parents: If might be done in a tournament selection scheme, in which the 2 best cands became parents & are replaced by their 2 children: it would be better to replace 2 worst cands by children of 2 Best Cands. (which may be done).

In general, if 2 parents are much good, they should not be discarded!

Of course, my Method of using PPM does generate a S-Grammar There is still much Q as to whether a good (mut(trees) often in G.P. may not be better than PPM in certain impt ways. Hvr. while GP can ~~not~~ generate useful subtrees in cands, it cannot use them in context-all. Also, GP does appear to generate excessively long cands (??) ← Apparently not!

0 3 If M is the average length of a branch (averaged over the entire corpus of cands), Then replacing a branch of mean length M by a branch of mean length M, leaves the expected branch length invariant.

5 Hvr, one normally restricts the choice of parents' branch: the usual way is to use a subcorpus (say cands of $G \to X_0$) ... in which case the above .23 - .25 applies & mean branch length is invariant.

---

0 Actually in G.P. Chosing a branch at random may be regarded as "sort of" choosing a smaller branch w.r.t. basis of some context! i.e.:

Chosing branch ABCDEF w.o. context is equiv to choosing branch BCDEF with context "A".

But still PPM doesn't work that way! A context occurs first then the p.d. on what will follow is altered.

---

36 A way to get context in GP that suggests a better way to use context in PPM! To do crossover with any subpopulation, M. Pick a cand from M (a random one say). Pick a point on C_0. look at context of this pt on C_0. Pick random pts on random members of M, until one finds same context as the chosen point on C_0. When the same context is found, exchange branches below the selected cands in pts of same context. This produces 2 offsprings. → Amore efficient → 32.00

(Spac 0:319.40) : In Lsrch and Searchfor solns (both d and S induction): 2 kinds of Difltis: (1) Finding ind funcs that will satisfy all of (or as much as poss (of) t. corpus "in Problem Pool") as poss. (2) Not (wanting to (having to)) check a new cand on t. entire corpus [crossed out] for each Cand.

1984

[SN] I don't seem to have a clear picture of t. state of affairs! — Reminds me of my state at t. start of my "Month at IDSIA"! Recently, it seemed quite clear that I had started pgmg PPM. That's a nice Detail that I think I understand somewhat, but t. rest seems to be a middle (muddle)!

How to Improve GP seems clear, but getting GP to do TSQ's is still unclear!

BU Search is done by GP. What I have is a v.g. GP, but getting it to do TSQ's is not clear.

FOR a problem space for which a TSQ is available, it would seem that Lsrch should be vastly superior to BUsrch...

.12 So, [The Argument / t. argument is:] Say one has solved upto Q_n in t. TSQ: It should be not so difflt to refrm all t. info and use it to srch for soln to (t. TSQ augmented by Q_{n+1})

[crossed out] .12 would seem to be true indip of whether one uses Lsrch or BUsrch.

Also, [This Argt.] indip of whether one was doing d or S induction

T. main thing that .12 doesn't go along w. is (perhaps) t. "Problem Pool" approach — and even then, it can be modified to apply. — T. "Problem Pool" is as close as we got to a TSQ, in t. "post TSQ" part of TM's Education. Its close to t. Activity of t. "Free Lance Scientist".

[SN] Re: RPN vs. PN notation! It would seem that we would translate by reading code in reverse order.... that's all. RPN has "advantage" for Cand funcs, in that one can start of w. Q_i as one of t. args. This is not so easily facilitated in PN. — Tho one can simply put Q_i in as one of t. "primitives" — as I did in t. Analysis of AZ: which used PN. Superficially, it would seem that (other than because of (context) considerations) pc's of cands using PN and RPN would be identical — But if context is considered, they could be much different!

.18 [SN] On "Logical Reasoning"! A real, reasonable motivation for my not considering it in my models of induction (yet), is that others have done lots of work on this. — I mainly work in areas in which I feel that insufficient work has been done. → 323.20R

316.03 315.15 314.30.... 311.40 [D25.04] [SN] A poss. Difty in "T. Problem Pool" approach: Int. corpus of O^j's, O^j's will tend to be t. most adequate for "easy" problems. So Cones to deal w. these problems will be much more frequent than those usable for hard problems. This difty could be partially alleviated if O^j's that solved many probs were given Much wt — (which would tend to be true, it t. "default" solns (of essentially "unsolved" problems) are given "default pc's" of very low value. → On t. other hand in "Boosting" we want to try harder to find Cands that work t. probs that have, so far, been unsolved. STM 2.00 ← note complete discussion of BOOSTING.

If we use PPM, this may not be so bad! Say $\tilde{O}^j$ is an O^j containing more real "solns" than any other previous trial. New trials will tend to be like $\tilde{O}^j$ to start, but there will always be a reasonable chance that t. new Cand will not follow $\tilde{O}^j$ exactly. As soon as it

4TM

.0 : (spec 317.40) We may have to modify the R's so they accept/reject the new Q in the proper ways.
Or we may simply used the compressed sets assoc w. the R's as the effective R categorization functions.
The "compressed sets" constitute a "soft" (Probabilistic) R function. I've written a bit (not much) on soft R functs ----
I don't know where, hyr.

Back to 3TM 319-323: Just how far did I get on the "S-function representation problem"?
ibid 320.17 ff discusses 3 Issues. ~~Extrapolation~~ also ibid 319.00-.05
323.11-.18: Bibl. summary (ibid 327.00-.30 said to be "good" but reason the bundle it here
321.00-.09 is on "How to use parts of previously known to be good Oi's to make new trials" — "PPM" probably doesn't better.

.08 Ibid 138.23 ... about ~~searching~~ Lsrch based on $|O_i| + |R_1| + |R_2| \cdots$ order is I guess, ie expanded. ——
It certainly seems right, but looks like it would take enormous time! (Even for Lsrch, this is the unnecessary
time needed. The only Alternative I can think of is BU ("non-optimum") srch.
One should do Lsrch's BU srch in ll — the Lsrch soln is best (if one has enough to find it) ... if none does it
then BU srch is one alternative: is it the only alternative?
Re: the Lsrch order of .08: If Oi fits several Q's well, we would want to retain it for more
a/o (over R_j) trials. Ibid 138.28-.30 discusses this a bit. also ibid 319.00-.05
Ibid 318 may hold much interest (but is not in present "Page Packet")
The "Lsrch" of .08 may not be as simple as it sounds!

.0 : (D.29.04) : So the Basic Problem is: How far have I gotten in planning Phase 1 (or "Alpha" term)?
317.30-.40; 319.00-.19 are certainly on this problem.
In Phase I, I will (very probly) want to use a TSQ: Whether I will use pure Lsrch;
pure BU srch or a Mix — Most (likely) is uncertain. Also uncertain is exactly how the
the 3 would work.
Sub-problems are: Forms Representation of S-functs.
How to go from D. induction to S-induction in TSQ. — What would be some good problem
examples?

.0 Another View of TSQ is: Learning Math "lang" & "probsolving" from examples in
Maple, Mathematica etc. — In those cases, for Not-So-Hard problems, Phase I may be
adequate & d-induction may be adequate. Ideally, we next go on to S-induction.
The mechanics of this Genza are unclear! Depends, in part, on what kind of induction we're
doing .... eg numerical or string. (315.29; .16-.19) is one way to go from D-induction to "3 inv mc s-induction"
—— the details are unclear & it's not certain that it could be workable.

The next BIG Problem is Search over the S-funct. space for solns to the TSQ.
I've considered 2 kinds of TSQ's for this: ① Normal TSQ for Lsrch ② "Problem Pool"
— TM selects probs to solve from a reordered "pool" of problems. → (spec. 320.00)

: Topics of interest: Working Backward:

1) Roll of Defns in Lisp, Forth, AZ, Z144 ... That we may be able to [illegible] not searching or re parsing past. 316.04 ... 317.30.

→ 2) (316.31 .16-.19) The only way to MAX Efficy of: Construction of (D) con[illegible] Lngs for TM(Z) Methods of finding regys in Corpus. So far we've considered Langs like Lisp, Forth, AZ, Z141, Machine langs.

Regyosearchers: PPM (mainly): Subtree Searchers: "Context generators".

3) Search for Solns to TSQ's: The "problem pool" approach 311.29, 314.30ff
← Singlets, then Triplettes Then ... 315.09 a new tech using EDIT DISTANCE. (310.21 / 312.00) ← on Lsrch v.s. Bu srch.
→ [315.22] → T. need (again) etc; "R" ~~[illegible]~~ input ~~[illegible]~~ of EDSIA.
→ Edit dist and i. "R" input of 3 iomc. Not 315.34

315.29 : a epist. fam from D-funct. induction & S-funct production — Pseudo more general S-functs → 316.16-.21 more (like truely universal); 3 iomc.

313.25 - 314.24 another Go at "HMC" · try to map $d^2 \frac{n+m}{n-m}$ into $c + m \cdot \ln n$ of sd 78.

313.10 Proof that f. usual normzn defn is best for normalizing semi measures.

312.00-.01; .30ff: On Lsrch v.s. Bsrch. (maybe no real dicotomy!).

312.02 : ANt Colony System (ACS): What it is & how it is similar to ALP, ANN, Bz(?)pop, "Wisdom of Crowds"

18 → This ties several prob solving methods together. ∴ useful in writing PSM grammar.

you
807 ws lltr called

[redacted] all notes on website.
Also, Discuss: ask Q's via email, if you didn't ask them in class.
Also discuss my belief on "Cosmic God".
Mention it in Lecture Series.

10 : 316.31 is a very impt. point!

Th. regys (i.e. subtrees) in t. Lisp-ish notation seem to be t. most common types of regyt recognized by humans (other than Direct PPM type regys). Since all Universal langs are inter-transferable perhaps every universal lang will be internally "coded as a Lisp.

→ See what other kinds of regys (that are useful for compression) exist. Look at various kinds of problems, various "Breakthrus" in science as compressions: — try to design search schemes that would have made them more likely. Look at Polya & other books on prob. solving, creativity.

SN Dragon: Has a micro phone testing routine. [Also, try difft Mike/earphone comb. to see if that's it.] Also, make sure there is no other Mike input & chesses. Is Switch on Mike Busted? SWITCH

11 : 316.07 Re: Definitions: Consider, say a Forth/Oops: when we make a defn, say γ; and γ occurs m times in t. corpus our p.c. for t. corpus is $\approx P_{D\alpha}\cdot(P_\alpha)^m$

$P_{D\alpha}$ = pc of Defining α --- depends "on "leng th of α; P_α is the probability of one occurrence of α in t. corpus.

If, on t. other hand, we scan t. corpus & γ occurs m times in t. corpus, we post-hoc define γ, & its total pc in t corpus becomes $P_{D\alpha}\cdot(P_\alpha)^m$: same as before, but w. additional cc of discovering γ & its compression advantages in t. text. T. main apparent difference is. 316.20; reparsing previous corpus. This would appear to make a Big difrnce: Big Advantage:

20 — Hvr., this may correspond to Old, "wiser" Scientists [illegible] reusing parts of old theories, V.S. students, not knowing the older models, try more genuinely radical, "creative"

22 Solns. to problems that older, wiser Heads would discard as "Unlikely".

Q: Is .20-.22 a correct analysis of Older, Wiser vs. naive beginners??

It may be that t "Older Wiser" do not evaluate pc's properly: That they consider only very hy pc theories because they are too disturbed when they are wrong.

Any reasonably good srch routine need not make that mistake - - - -

29 — (It could make other, equally bad errors for other reasons?)

30 (Modern) Remarks on [3TM 319-323]: "On S-fncts & how to srch over them"

≈ "Final opinion" ≥ 323.00-.03: That it's t. problem of extrapolating a G funct. from a corpus of (O^i, G) pairs.

At that time, I hadn't considered PPM to help w. this & MEJ ($\frac{d_G}{G_{max}-G}$) to help work on PPM!

T. present problem is (1) How to assign (comparable) G's to O^i's that have diff difrnt sub. Corpi (315.09 is one approach)

(2) T. problem of not wanting to test each new O^j over t. entire each corpus. (The "Recognition functions of IDSIA Report, were a strong step in that direction.

36 One ≈ approach to (2) After we have a set of sets of Q's which R_i has recognized (so a set of sets) for each R_i we compress its Q. set. When a new Q comes in we see which compressed set it belongs to (we get a PD). We try to solve t. new Q using t. O^i's of t. R_i most likely to be assoc. w. it. If no success, try 2nd — O^i assoc w t. second most likely R. — (see ≥ 19.00)

4TM+ ....

"Problem Pool"

.0 : 315.40 : [scribbled out] If we use t. "problem pool" ≈ 314.30 : say we get O's Res can work small subsets of probs w.r.t. TSQ. These functions could correspond to difnt R values — as (315.34 – .40) w. R choices suitable for each problem. [Well! not so Obvious as to how to do this! — or if it would work at all!] One way is 317.36 : (compression for categorizn) → 320.31
[like L: Vitanyi USB Compression to Categorize Music.

3TM 425.04 Discn of Models for TSQs. Also other ideas of ...

04 → [SN] On ["definitions"]: T. way OOPS & Koza & AZ do defns looks about t. same!
— Difnt from Z141; in which defns are made after t. pgms look useful! ← Actually, one could do Both!
Analyse t. difference of t. systems: There may not be much quant. difference. If so,

07 I can save lots of time! → 317.11

[T. Symbol that initiates a defn. will have a certain pc; In PPM, This symbol will give its own/sub corpus of successful defns of t. past. Hvr, symbols before t. defint. symbol will influence t. subseq. symbol probys. I could disallow that, but it's a legit statistical dependance, perhaps I shouldn't complain?

Compare t. details of t. way OOPS, Koza & AZ make defns. Also, see how recursive defns occur. They can be very inefficient in execution! If so, have a special Operator that finds them & (compiles / converts) them to more efficient code. — T. PC of t. cond, hvr, is that of t. original, uncompiled code.

.16 I Guess that one of t. objections I had to OOPS using Forth was that t. regularities in code were not apparent, that (perhaps) [They were unlikely to be detected by PPM]

19 A "good Forth writer" is supposed to put all (?) of t. Regys into t. Definitions. D D

.20 Consider: In constructing a Cand, say, we chose t. defn symbol D. This automatically changes non D as a prefix — (i.e. things that have not been defined before). Any hy pc sequence following D will be a good cand for "compression by definition". Once defined, this defn could be used to reparse t. previous corpus! This would (probably) screw up PPM's WB xfm data object.

There may be some point (advantage) to having Defns not apply backwards. It's a way of "forgetting the errors of the past" & starting out on a new Life" ☺.
— Tho it does seem to lrn more slowly. We could do evaln of t. pc of a new defn. W.O. reparsing t. past!

30 O.K. : So .04 – .30 does give a new perspective on Definitions ≡

31 Very impt: A very impt Q is .16 – .19: what Lang. Conventions/ Definitions make it possl. for PPM to recognize useful regys? [More Generally, [what lang ≡ Regy detection system] works best?]
Lisp + PPM seems not bad. What about Forth + PPM or Lisp w. OOPS-type defn scheme? + PPM. Also Mechanical Lang & PPM

Well, a Lispish lang w. subtree discovery seems fine (but expensive).

.35 Well, what about being able to recognize nearby/sub trees? Consider a certain subtree, say; Could it be possl. to index corpus so that all instances of that subtree could be cheaply found? Maybe some kind of ..., "Holographic" Memory?

Perhaps Study TRIE data structure to get ideas on .35 — also to better understand discussions of PPM.

spec
0:314.40: [I assume here, it's meaningful to say "solve" a problem --- somehow we have to decide on criterion for adequacy of a soln — i.e. what pc for soln is "adequate"]

So in the "Solution pool", we are pushing toward getting solns containing a large no. of problems.

Say we have 10 problem corpus. If we have a soln O^j for probs 4 thru 10, then we will try very hard to get a soln for 4 thru 10 plus one or more of probs 1, 2, 3.

We will remove from the PPM corpus solns that are much worse than more recent solns. (But maybe not; we may still want some of them also.

Anyway. Q is how much wt in PPM corpus to give to various solns.

39 A slightly Different tack: Say we have n probs. in TSQ: We run cands on all 10 problems & using "Edit distance" we obtain for each cand a pc for the entire TSQ. At first, most probs will have very hy bcosts, but we would put them into the PPM corpus anyway. We will then weight them depending on their bcosts for the entire TSQ corpus. Essentially, we only have one problem. Its an OZ problem. We can, say, elim from PPM corpus all solns w. bcost > x: then we select x to maximize $\frac{\sigma}{G_{nzx} - M_G}$.

5 We can assign a simpler (default) pc to solns that take too long to find Edit Distance (.22) → 320.31

6 [SN] A (Gauzy trim) on the "Edit distance" idea: Use a 3iumc, try a few short "R" inputs, then use R w. smallest Edit distance, or use all 4 or 8 or 16 ... R values & pick "closest" one to the "True" A_i; Or, use the set of R values w. [illegible] outputs "close" to A_i. Training of this sort can move toward a useful "normal 3iumc" w. R that gives A_i exactly.

19 — 310mc 310.12

0;310.21 → (SN) On edit distance: If 2 strings of length n & m are very close, it should be often easy to find a short edit distance — in time much << m·n (<< m·n)

21

22 (.15) Clearly we can't use .09 — .15 when the TSQ corpus is very large. Checking all previous QA's on a cand would take too long.

An idea related to (.22)! We may notice that a certain subset of problems gets hy PC w. O^j_1; Another subset of probs get hy pc w. O^j_2. We then try to find a way to Recognize ("R" system) which subset a problem is in, & use the appropriate O^j..............

What I'm noticing: that using R input = Λ (i.e. 2 input umc) w. Edit distance is essentially = using D-functions! So this is a PATH from D funct to s-funct. The next expansion is

29

30 (.36 – .38) — A simplified version is to have each d-function have 4 or 8 ... outputs: correspond to "R" having 2 s4 – 8 possible values. Next: perhaps have R w. values [illegible] w. different wts corresponding to the "Length" of R.

34 The "R" device is equivl. to having a bunch of O^j d-functions & for each problem, picking them at random (w. each O^j funct having its assoc PC (≡ wt.) w.t. ∝ $2^{-(\text{no. of bits in } R)}$). I discarded a model much like this, some time ago, but I think the R choices for all QA's had to be the same. If the R choices are (more) indep for each QA, then its OK, & ≡ normal 3iumc. → 5TM4.23

Go Over this idea Carefully & It seems very impt! Also Note 317.36 on a soft (probabilistic) "R" function → see ST 4.23–.26 (spec .316.00)

30 : (313.40 spec) : If σ^2 is true varc., t. model will have varc $\sigma^2 \frac{m+n}{n}$. Go thru calcns carefully using this fact.

My model pc $\prod_{i=1}^{n} \frac{1}{\sqrt{\pi}\,\sigma} e^{-\frac{1}{2}\frac{(x_i-\mu_M)^2}{2\sigma^2}\left(\frac{i}{i+m}\right)} = \prod_{i=1}^{n} \frac{1}{\sqrt{\pi}\,\sigma\sqrt{\frac{i+m}{i}}} e^{-\frac{1}{2}\frac{n}{}}$

} Assuming m values of prior corpus are "Given" in both cases.

True model pc: $\prod_{i=1}^{n} \frac{1}{\sqrt{\pi}\,\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} = \prod_{i=1}^{n} \frac{1}{\sqrt{\pi}\,\sigma} e^{-\frac{n}{2}}$

$\frac{m}{\text{true}} = \sqrt{\prod_{i=1}^{n} \frac{i}{i+m}} = \sqrt{\frac{n!\,m!}{(n+m)!}}$ | $\sqrt{\frac{n^n m^m}{(n+m)^n (n+m)^m}} = \left(\frac{n}{n+m}\right)^n \left(\frac{m}{n+m}\right)^m \sqrt{\frac{n\cdot m}{2\pi(n+m)}}$

for $m \ll n$: $\left(\frac{n}{n+m}\right)^n = \left(\frac{1}{1+\frac{m}{n}}\right)^n \approx e^{-m}$ | $\left(\frac{m}{n+m}\right)^m =$ $\left(\frac{\frac{m}{n}}{1+\frac{m}{n}}\right)^m = \left(\frac{m}{n}\right)^m \cdot e^{-\frac{m^2}{n}}$ ≈ 1.

e^{-m} could be part of the Akaike factor,

But $\left(\frac{m}{n}\right)^m \cdot e^{-\frac{m^2}{n}} \sqrt{\frac{n\cdot m}{2\pi(n+m)}}$

?

$\left(\frac{ne}{m}\right)^m$ looks like a correction factor for $Z(q)$!
(Perhaps : check details!)

Its ln is $m\,(1+\ln n - \ln m) = \boxed{m + m \ln m} + m \ln n$

313.25R : $C + A\, m \ln n$ — $m + m\ln m$ could be part of C : in which case, we get $A = 1$, not $\frac{1}{2}$.

OH! I forgot the fact square root of t. whole expression so $\ln \to \frac{1}{2}\ln$. or $\frac{m}{2} + \frac{m}{2}\ln m + \frac{m}{2}\ln n$

plus $\frac{1}{2}\ln\sqrt{\frac{n\cdot m}{2\pi(n+m)}} = \frac{1}{4}(\ln n + \ln m - \ln(n+m) - \ln 2\pi)$

$\frac{n\cdot m}{2\pi\,(n+m)} = \left(e^{-\frac{m}{n}} \cdot \frac{m}{2\pi}\right)$ $\frac{1}{4}\ln$ of $= -\frac{m}{4n} + \left(\frac{\ln m}{4} - \frac{\ln 2\pi}{4}\right)$ part of "C"

⊙ so the "C" part is ok. but we have $\frac{m}{2}\ln n - \frac{m}{4n} = \boxed{\frac{m}{2}\left(\ln n - \frac{1}{2n}\right)}$ ← for the part that depends on n. The m dependent part goes into "C".

I did assume $m \ll n$ in t. approxn. — so maybe this is acceptable for the "$A\, m \ln n$" part.

rather than $-\frac{m}{4n}$ | $\frac{1}{4}\ln\left(\frac{n\cdot m}{2\pi(n+m)}\right)$ | T. part involving n is $\frac{1}{4}\ln\left(\frac{n}{n+m}\right) \approx \frac{1}{4}\ln\left(1-\frac{m}{n}\right) \approx -\frac{m}{4n}$

So, its "in t. rite ball park", but I'm really not so sure its rite, because my algebra is so Buggy !

24 — The t. derivation of .00 ff is not so complicated !

30 : 311.40 : Consider 311.29 ff (t. "problem pool approach"). Suppose we had a way to list S-functs in something like pc order (Note 312.00-.01; .30ff on a possible way to deal with). T. Q is, how do we learn a bunch of QA's ? In we find solns to single probs, then pairs, then triplets, etc. What pc goals do we accept for singlets, pairs, triplets, etc ? What wts do we give to solns in t. say "$\approx$ PPM corpus?

. If we had one (QA singlet) only in t. TSQ, we'd just work on it until CB was reached.

If we had 2 in pool : Do one, Do t. other then do both together. w. 3 in pool ! do all pairs first, then all 3.

37 ⊙→ Note Bene: Working any subset of t. corpus must be easier than solving t. entire corpus

For a larger corpus, we start by trying to do as many probs as possl. w. one O^i :

NIPS 3TM 216.00 Read; Also [illegible] STACK of ID 611.00f

00:312.40 : We do Lsrch using pc'_0 as pc index ∴ we use $\frac{cc'_0 + cc^2}{pc'_0}$ as a "stopping criterion" in $\Sigma T \le 2T$ Lsrch; In ‖ Lsrch we work on 2 cand so as to keep $\frac{cc' + cc^2}{pc'}$ ~~[illegible]~~ the same for all cands.

T. result is that we are doing Lsrch on an OZ problem, w. $G_{orc} = pc' \cdot pc^2$.

I think that for d-induction, we can use (INV problem / simple Lsrch), but for s-induction, we have to treat it as a OZ problem. So ~~[illegible]~~ in OZ Lsrch we try to find best Gorc w. available CC. If a certain Gorc, G_0 is known (BY t. TRAINER) to be able to [illegible] w. a cand whose $pc = p_0$ ∴ cc of soln is cc_0 then we will find that Gorc $= G_0$ ~~[illegible]~~ w. $CB = \frac{cc_0}{pc_0}$.

T. idea of t. TSQ is to make p_{c_0} big. T. TSQ's "Adapting" to search PD will not modify t. cc_0 — (tho ~~[illegible]~~ TM can affect t. cc_0 in "Phase II" ~~[illegible]~~ TM's.).

0

HA! I found that proof that t. best normz function for Sami measures is $\tilde{x} \to \frac{x}{x+y}$; $\tilde{y} \to \frac{y}{x+y}$ $\tilde{x} + \tilde{y} = 1$. It's on (NIPS 3TM 207.10–.19)

T. problem is lines .12–.13 and line .18

↓ p.c. produced so far to nth term of corpus of univ. d.f.

Workson $E \sum_{i=1}^{n} (e r i)^2_i \le$ ~~[illegible]~~ $\frac{P_{sol}(n)}{\mu_B(n)}$

for finite derbable μ, ~~[illegible]~~ $\frac{P_{sol}(n)}{\mu(n)}$ will be bounded by t. pc of t. deriv of μ.

Hvr, t. minimum value of t. normz constant occurs when one chooses a sequence of bits in t. sequence

⇒ ~~[illegible]~~ $f_1(x,) \div X$ is minimal for each of t. ~~[illegible]~~ or 1 choices in t. sequence.

This min is maximized when ~~[illegible]~~ $\frac{f_1(x)}{x} = \frac{f_2(y)}{y}$ for all of the choices

T. only functions for which $\frac{f_1(x)}{x} = \frac{f_2(y)}{y}$ and $f_1(x) + f_2(y) = 1$ is ~~[illegible]~~ Normal Normen.

0 Proof: $f_1(x) = \frac{x}{y} f_2(y)$ so $\frac{x}{y} f_2(x) + f_2(x) = 1$; ~~[illegible]~~ $f_2(x)(1 + \frac{x}{y}) = 1$

21 So $f_2 = \frac{1}{1 + \frac{x}{y}} = \frac{y}{x+y}$, **The Proof looks pretty reasonable!**

HMC (how many coefs) In Sol 78 p 421 last ¶ of col 1; I say t. ~~[illegible]~~ ln ~~[illegible]~~ of

25 t. ratio of model pc's is not $2^{-k(n)}$; but $k(n)$ is replaced by $C + A b \ln n$. [derbs. — structure of model; no. of symbols; b — no. of params (continuous); "characteristic of accuracy of model"]

Consider linear preds w. m coefs, n data pts.

Using $\hat{\sigma}^2 \frac{n+m}{n-m}$ formula! (↑ observed σ^2.) True model will have $pc = \prod_{n=1}^{M} \frac{1}{\sqrt{\pi}\sigma} e^{-\frac{(x_i - \hat{x})^2}{2\sigma^2} \frac{(n-m)}{m}}$

My " " " " " $\prod_{n=1} \frac{1}{\sqrt{\pi}\sigma} \exp - \frac{(x_i - \hat{x}_{my})^2 (n-m)}{2\sigma^2 (n+m)}$

~~[illegible]~~

0 $pc_{True} = \prod_{n=1} \frac{1}{\sqrt{\pi}\sigma} e^{-\frac{n}{2}\frac{(n-m)}{n}}$ — those σ's vary w. n also! $\left(\prod \frac{n}{n+m}\right)^{\frac{1}{2}}$ most (n cancel)

$pc_{my} = \prod_{n=1} \frac{1}{\sqrt{\pi}\sigma} e^{-\frac{n}{2}\frac{(n-m)}{n+m}}$

ratio $= \exp\left(-\frac{1}{2}(n-m)\left(1 - \frac{n}{n+m}\right)\right) = \exp -\frac{1}{2}(n-m)\left(\frac{m}{n+m}\right) = \exp\left(-\frac{1}{2}\frac{n-m}{n+m} \cdot m\right)$

$\frac{1 \cdot 2 \cdot 3 \cdots m}{m+1 \cdot (m+2) \cdots (n+m)}$

m corresponds to b, t. no of coefs.

Woops! In the $\prod_{i=1}^{n}$, I didn't really do product of n=1 to M

$\frac{1 \cdot 2 \cdots m}{(n+1) \cdots (n+m)}$

It ought be $\exp\left(\frac{M}{2}\right) \cdot \sum_{i=1}^{n} \frac{i-m}{i+m}$) start sum at $i = M$ $\frac{1}{1+m} + \frac{2}{2+m}$ It may give a $\ln(n)$ term.

$\frac{1}{1+1} \frac{2}{1+2} \cdots \frac{n}{n+m}$

$\approx \sum_{i=1}^{M} \frac{i+2m}{i} = n + 2m \sum \frac{1}{i} = n + 2m(\ln M + \gamma)$?

got signs in exponent right! $\frac{?}{?}$!!

This approach may eventually be correct, but t. alg. has to be gone over carefully!

well known function! $\frac{n! \; m!}{(n+m)!}$ → 314.00

21:32 Wed.
β

0 : 310.23 : N.B. In t.³ Ed. to's idea (its Genzus) of 310.21 : This really doesn't do trials in pc order at all: its more like BU srch! — Not nearly ! (30) Isea

.02

ACS (Ant Colony System : first issue of j. of Ev. Computing : 1997

Full PDF in PS: 12/15/04 (I also have hard copy of journal)

Authors say it was better than Sim. Anneal & Ev. Comput. — (both non-crossover using) — for perhaps Trav. Sls prob (TSP).

My impression of how it may work: We run some ants thru a maze; note distance run by each (d_i) after it finishes. Then each path is given (total) Pheromone wt. $\propto \sum \frac{1}{d_i}$ of the ants that traversed that path. Henceforth, ants having a choice will make them $\propto$ to total pheromone of that choice. I think t. paper is readable, but disregard part that gives (intuitive) motivational applic to behavior of Biological Ants!

This is similar to ALP: we have a set of alphabet letters. We have a set of cands that are strings of these letters — w. empirical Score (G_i). If we regard each string as a corpus code & Score a pc of that trial, then we want to assign wts to t. alphabet ($\equiv$ pc's) $\ni$ t. total parallel pc of t. solns is min. Say L_k is k^{th} letter in alphabet. m is t. trial no. If $N_{k,m}$ is t. no of times (L_k) k^{th} letter is used in m^{th} trial ($N_{k,m}$ often $=0$) then rel. wt. of L_k is obtained by maximizing $\sum$ parallel pc of code by suitable wt. assignment to L_k. It can be solved iteratively, by assigning initial wts & modifying them on basis of apparent pc's of individual trials. — (This is a standard approx method for certain kinds of pc maximiz. probs.)

(I don't think Ants would survive using t. rules they describe!)

So next approx for L_i : $\propto$ total wt. of L_i in successful trials — each trial is given wt. $\propto$ product of pc's of its letters.

In t. TSP: the letters have "context" (i.e. each city has only a few roads coming out of it). For each context (= city), we have a separate alphabet of letters.

That (ACS) they should do so well (& indeed they did!) is encouraging for ALP!

Another possibly positive encouraging to ALP is "The Wisdom of Crowds".

*T files
552/12/10/04
260 SP/12/10/04*

Possibly Similar is Back Prop ANN: Averaging wts of various cand. solns.

.10 .(01) Actually MDL has some diffy: — w. t. "2 part code" — we are able to construct Models (part 1) in pc order, but evaln of Models (part 2) can be anything! While this is true of MDL, it is also true when we average our codes in ‖. In fact, it is true for many, if not most (or even almost all) S-funct evaln schemes!

If it is how do we compute t. Lcost ($\equiv c_i$) of a particular, known. soln? Well, say we know part 1 of a MDL code has $pc = p_0^1$ & $cc = cc_0^1$; Part 2 has $pc = p_0^2$ & $cc = cc_0^2$

To this soln, we assign $pc = p_0^1$ & $cc = cc_0^1 + cc_0^2$. p_0^2 is not regarded as a pc in this part of Lsrch — we regard it as part of t. Score ; t. Score is $(p_0^1 \cdot p_0^2)$. T. time to find it using ‖ Lsrch is $\frac{cc}{pc} = \frac{cc_0^1 + cc_0^2}{pc_0^1}$ (313.00)

S-funcs (cont)

00:310.40 : finite no. of x_i, one unit to randomly sample them — but ① There are already a deep set of x_i.
② Even w. [illegible] of (x2), One has no idea if t. ones have been included in t. sum.

It [illegible] $[x_i]$ are simple scalar [illegible], we may be able to norm. them w. randomly constructed $f(x_i)$. For vector x_i, not so easy.

Hvr, t. attractive feature [illegible] $f(x_i)=p$ is that it may be easy/fast to compute x_i — which would save lots of time.

Soln of time varying OZ probs 3TM 310.12–.25

How many Certs, 3TM ≡ NIPS 293–297; 303, 304

3TM 266 on Paul's stack lang.

Curv fitter TM 2:778.00

1TM 65.30 Proof that Lsch [illegible] only worse than levels in [illegible] order!

Conditional P.D.

So: Dropping $f(x_i)=p$ for t. time being: Consider other forms of S-funcs.

Consider t. 3 inputs: Starting w. t. UMC: One input defines O^j.
Another input is for t. arg x_i (string or no.).
Another input is t. random component, R.

The output = $U(O^j, x_i, R)$. Ideally t. machine reads all of O^j, then all of x_i, then read all (or part of) R and then prints t. output. If it is allowed to print out put after looking at only part of R, then t. meaningful parts of legal R's will form a prefix set, so their Σ pc ≤ 1. If they are a complete prefix set their Σ pc = 1.

R need not form a prefix set! R can be arby finite strings of symbols w. an "end" symbol.
T. pc assigned to t. end symbol will be determined by average lengths of solns to problems in "t. present context". We assume symbols of R are uncorrelated. If they are binary (0, 1, end) then they will have pc $=(\frac{1}{2}-\frac{\epsilon}{2}, \frac{1}{2}-\frac{\epsilon}{2}, \epsilon)$; $\frac{1}{2}(1-\epsilon); \frac{1}{2}(1-\epsilon); \epsilon$. pc of a string R string of n binary + end will be $(\frac{1}{2}-\frac{\epsilon}{2})^n \cdot \epsilon$. ϵ is chosen to make pc of set of known R's $\epsilon \approx (\bar{n})^{-1}$.

We can actually make $1-\epsilon$

Hvr, I don't think I ever found a good way to search for solns: That older note in ~Nov 2003? 1/10/10 [illegible]
It was on a yellow sheet of paper! first at Page 33 then → 133.

That's it!! 3TM 133.00 ff / → ibid 137 — 138 (poss) Bug on ibid 138.285

3TM 319.10 ... 323.11, 323.21 supposed to be more serious approach — That there were Bugs in this

<Tho it may be that w. recent discoveries, I could fix those "BUGS"!>

A different tack I'd considered for TSQ's is (a) t. "problem pool approach": N.B. it not exactly t. "problem pool" approach! We have an unordered batch of QA's: TM tries to find individual solns, then pairs: then ... as many as possible by same O^j. This builds up t. set of cores in t. corpus in a useful way. We make pro dist. of n the O^j that can solve n of t. QA's: #

Hvr, t. f... is for d-induction. I had a previous discussion of this "pool" approach, but I don't know if it holds for S-predn. Well, even d-predn has a pc assoc w. each O^j.
Say we find a bunch of O^j_k! each assigns a pc to some subset of t. QA corpus. How to Wt. this corpus of O^j's?

see 314.30

$-Functs Rev.

: start w. discus. of thm $P = f(x)$ $x = \begin{pmatrix}\text{number}\\ \text{string}\end{pmatrix}$ is not such a good form

We'd like formulation where pc would always be >0 & normed. (maybe not exactly >0 ?),

Basically, T. problem is to get a form $\in L$ s.t. (or BU such) is least expensive — In particular — we are interested in problem of minimizing Pc of a OJ wrt "corpus up to now" Since OJ is usually an $-funct (& we are more interested in OJ's than in $-functs because they enable phase 2)

[Actually, even when OJ's are d. funcs, we will learn by appropriate their to get pc's of solns. ABDEFG 'abcdefg. ABcdefg.]

We want $functs in form that can be rapidly generated & tested to find f. pc of f. corpus wrt. Prem.

For Zrchy, we'd like to be able to Generate $functs in ~ pc order — (.50)

Re: 3 input unc.: I think I have a better understanding of it now so that f. ∈ AZ (say) might not be bad (if do want to find way to do recursion or just Do loops (to enable Prim. Rec. funcs).

One way to do 3 iunc's: We use AZ to construct a function that has Q_i as one possl. input — whose output is $\in A_i$. Actually, it's $f_j(Q_i)$ & f. pc assigned to TG's outputs is its "distance" from A_i (f. "correct" output). We need a good measure of distance (this is a clustering approach). In many Numerical problems, the simple $|x-y|$ norm — abs value or squares or modulus squared would be OK. — But for strings it's not clear what metric to use. Ideally, we would say: "How much info" we need to alter $f(Q_i)$ into A_i. This usually takes too long to Compute. One approx is "Edit distance". See Google; also (PS\editdistance 12/11/04)! Also see Bookmarks on Editdist. This paper says "using Dynamic Prog it takes time $\propto$ mn to compute edit dist between strings of lengths m & n respl. The Edit dist can have a diff "cost" for Additions, Deletions or Substitutions. (see 315-16a, 315-21) → 312.00

(11) Consider $functs of f. form $f(x) = \begin{pmatrix}\text{string}\\ \text{or no.}\end{pmatrix}$ Ruby : would like f to be >0 for all x in, if possl. normed ∈ so we can compare various reasonable final "f" candidates.

One way to ∃ Normz: we want normed f(Xi)'s: Just try k randomly chosen X_i then $\left(\frac{1}{k}\sum_{i=1}^{k} f(X_i)\right)^{-1}$ will be ∃ normz const. If want start w. small k; if cand looks good, use larger k to get more precision. No! I. true norm const is $\left(\sum_{i=0}^{\infty} f(X_i)\right)^{-1}$.

By studying μ_k & σ_k^2 of $g(k) = \sum_{i=0}^{k} f(X_i)$ could we estimate $\mu_{k=\infty}$? — No so easy! We have no idea as to whether large $f(X_i)$'s have been included in f. series. If Prem says → 311.00

0: 308.40 : Even in BU Mode we have to deal w. inventing "R" functs or equivts. We can use the idea of 297.25: we test a cand on a fair no. of hard problems only. When we find a fair no. of cands that work on that set, we begin testing on random "easier" problems. This may be good cut of both Lsrch & BU probs.
— The Getting "R" functs is probly desirable — TM may be able to generate t. equivts of "R" functs by itself, w.o. trainer telling it directly about them.

1: (308.40 space) : T. idea of 308.37 is impt. [redacted] The technique dscrbd would be fine for Lsrch of S-functs.
Somehow it reminds me of BU search — But BU srch starts w. low G cand & tries to improve it —
12 classically by mutation (simulated G.A.); less classically by crossover (GA, G.P.) & even less classically by MEJ w. assoc S-grammars (Fitting S-trings to data).
4 So maybe t. stuff I have now is enuf to do an acceptable "phase 1" (16) → Like 302.20
5 SN mm A possl. Source of ideas for forms, representations of S-functs is t. Duda, Hart, Stork Book!
6: (.14) T. 2 Main problems recently "Solvd" (see 295.04 for some refs: 302½.00-.40.): ① How to represent a variety of S-functs to enable
7 Lsrch & perhaps BU srch. ② How to deal w. problem of verifying O^j (in either Lsrch or BU srch)
18 On t. whole previous corpus, when we are doing a large t.s.q.
I should perhaps write good summary of these 2 "Solns". (Bibl. review)
For ② (.17) : Verifying, evaluating a O^j cand. (306.24-.27); Quick Abort (297.25-.40); (309.00-.05) (294.32-.40) is early devn of "Quick abort"
The "R" function (Recognition function) Discussd in Early part of IDSIA Report (pp. 8, 9; §1). || 272.37 a somewhat different Approach
(A possl recent idea related to R method: 294.20-.28 || Also Note 250.11 ff on "problem pool" approach
For ① Representation of S-functs: (.15) 274.04-.40 ; from ≈ 265 on : (266-267: on Universal Analog Machines)
270.13-.19; 271.00-.07
302½.00-.40 : 308.00-.40 (.30 space -.40); 309.00-.12

0

→ (.30) is a good way! we do Lsrch as for d-Predn, but each trial is assigned a "distance" from t. correct Az.

0: 302½.40 ; Originally I thot that I'd start w. d-predn; then "[illegible]" to Spredn, but I didn't have clear ideas on how to do this. I write 302½.60 to do it now ---- but what seems easier is:

[illegible] To start w. s-funcs: to first learn sum/sub/mul/div (8, 37) :

T. machine learns that soln. is 8, 37 + .25 ; − .25 ; × .25 ; / .25 : If then tries to correlate t. empirically correct functional form, w. t. correct soln. This is easy to do, because t. functional forms & t. solns. differ in only one place & those "ones & zeros" correlate 100%.

So ≥ sub 3,4 → − (3,4) → −1 ; mul 3,4 → × (3,4 → 12 | We then need a way for TM to implement these discovery: If (sub) → − is one way, If (mul) → × etc.

0 Here We have broken t. lrng task into 2 parts: T. first part is bcost by a lot & t. second is bcost even more.

[T. problem of [illegible] to evaluate complicated [Lisp] functions, [illegible]. T. reason its annoying is because I think I know how to do it, yet I find it difft. to get TM to lrn it —— I don't know what primitives or observations to put into TM so it would get t. right idea of "sub object", "t. evaln of sub objects".

.17 T. idea of associating a number (or Value not nec a number) w. [illegible] certain subtrees. (.27)

Some ways to deal w. this: ① find "acceptable" soln (by me) & put it into TM. So TM has t. soln & t. sub. absts. Go on to other problems. T. experience will perhaps enable me to get a usable soln. to this problem.

0 ② in "SAARB TM" I had a stack machine & I think it solved this problem easily .... but this was because notation for problem was match by "Stack" structure of reference Machine.

[(NB) It may be necy for TM to use t. same methods that I use, to solve problems if I am to use Lsrch, which (usually?) means that solns. will be in t. form that I solve t. probs.

.27 (.17) Possibly relevant! T. subtrees are "objects" w. "property lists" (= "slots"). "Value" is one possl. Property. A difty here is that we'd perhaps have "too many objects" for us to have a property list for each!

0: 302½.23 While 302½.00 – .20 gives P.D.'s on continuous variables, & at discrete mathical pts., finite in no. or infinite, It doesn't give P.D's on strings (finite or infinite). ALP certainly does that! — But usually in a very hy cc way. 302½.27 – .37 Does P.D's on strings.

Actually, if we use a "proper" P.D. on code length ("3 inputime." "3.iume" isn't so bad!) For each (QA, say) trial $\tilde{A}$ generated, [illegible] we compute $|\tilde{A} - A_{correct}|$ & add it to t. codelength of $\tilde{A}$. T. defn. of Norm can vary; we can start w. some simple, default, Norm, & modify it for each Q or Qclass, when we get large enuf SSZ — (See discussion of "clusters" on 302½.31 – .37 its perhaps t. same thing)

37 → Hvr, as derbd, these "Clusters" aren't much good for Lsrch ! (WOOPS!) Unless we do an Lsrch in listing cluster "centers", & [illegible] our Gor is pc of cluster × pc of $\tilde{A}$ wrt. that cluster. This differs from Standard Lsrch [illegible] ... → 309.00 → (309.10) Spec

D6.04
4TM

MEJX.00
Moy (xwindo)

.00 : Variation on MEJ method $\left(\frac{\sigma_G}{G_{max}-\bar{G}}\right)$: (Mej xwindo)
We have this complete population : In Bins for various G ranges. So say 100 bins of = width.
(or chose bins for ≈ population.) Then for each bin, we keep track of its population (≡ wt)
.33 its $\bar{G}$ & σ_G^2. Then, If we make a PD on the Cands where $pc \propto e^{-(\text{Order of G from top})/\lambda}$
for each λ we have a different S-lang (≡ PD). For each λ we can obtain its
σ_G^2 & $\bar{G}$ since they are obtained by wtd means of the bins: wts are population
times something like $e^{-\frac{x}{\lambda}}$ or whatever. So we can calculate $\frac{\sigma_{G_\lambda}}{G_{max}-M_{G_\lambda}}$ ← same for all λ
for each λ & pick the one that is hyest.

.6 Using this "xwindo" type of d.f. gives us benefit of occasionally low trials —
which is not true for sharp cut-off of normal MEJ [Call this MEJxwindo]
[or MEJX]

HA! Some trims on Olde MEJ! Say we have 6 "Bins" of .00-.03. We Say "x" is the
G level of a Bin. Consider $\left(\frac{\sigma_G}{G_{max}-\bar{G}}\right)$. If we add a new bin's data to the existing P.D. (≡ S-lang.)
σ_G will change & $\bar{G}$ will change. If they both change in same proportion, we are at
a MEJ max, Say our candidate bin has a certain σ^2 & M. The change in denom is easy to compute.
The % change in Numerator is harder. $\text{new}\,\bar{G} = wt\bar{G}_{old} + wt\,\bar{G}_\Delta$
$\sigma_G \to \sqrt{\sigma_G^2+\sigma^2}/($ $)$ $\text{new}\,\sigma_G^2 = wt_1\sigma^2 + wt_2\bar{\sigma}_\Delta^2 + \left(wt^{?}(\bar{G}_{old}-\bar{G}_\Delta)^2 ?\right)$
.0 So I'm not clear on this! At first I thot it would be easy to tell if numerator & denom
changed by same %; so I'd know when I got MEJ: But maybe not!

.22 I also had that earlier "Complete Model of TM" (IP 510.09-.10 ; 509.10-.18 ; 509.36-510.08, 509.19ff ; 2002)
This was the idea of a Grand PD used for TM_1 & TM_2 also. It was described in introduction of (DSIA Report.
So: I want to (List, characterize, Dsrb) all of these ~ "Complete" Models of TM. I guess these Model
had some 2 diffys that I may have solved ... reference 309.16-.18

.0

4TH Bibl. :00

30 : : A MSPress (2002) on improvements to PPM. Seems to have good discussions of various issues:

P:\PS\ PPM improvements - readable - Raccoon Proofs.ppt 11/18/04 date in "PS" file.

0 : 305.31 (5) A fifth kind of lang is machine lang. — Say a set of RISC instructions: The objections to FORTH (Rest it's not easy to find definitions, comments) apply even stronger to RISC insts.

Typically we have a small set of insts that get parts of precomputed (a) a large no of Registers, (b) several stacks.

(c) RAM (assumed). i.e. Conditional Jump instructions. FORTH advantage is extreme speed. (Pro [redacted] (slightly compiled) FORTH can do a book as well).

20 : Next: Suppose we have chosen a Language: Several choices of action:

1) We can do LSRCH over to find a candidate ⇒ they fit the corpus $[Q_i A_i]_{i=1}^{n}$ as well as possible. As guiding P.O. we use a 5-long w. corpus of $[Q_i A_i]_{i=1}^{n+1}$. We do tests in PC order.

24 2) If as in (1) we use LSRCH, this means each Candidate must be tested on entire previous corpus. As corpus grows, this can become very expensive. One "solution" is to try worst (most diff) QA's first. [QUICK ABORT! see 297.25 ff.] see 297.35 for objections to this. — Suggests that

27 "R" method of IDSia reports might be OK, or in write direction.

3) We can choose not to do LSRCH (see 303.10 for dizzen of LSRCH v.s. $B_{\text{actual}} U_p$ srch.)

30 so in BU srch, we start w/ a not very good ("Default") code for the corpus $[Q_i A_i]_{i=1}^{n}$ and do ... Wee loop mutations on other modification methods to try to improve it. GA/GP is one way to do this: we still have the "Test whole corpus" problem of (.24 - .27), hmm. For search we can use 1. PPM and probably the MEJ $\left(\frac{\alpha_c}{\text{[redacted]}\, G_{max}\, i_c}\right)$ method (or modifications of it),

D 4.04

4TM

128
11 × 12 = 132
11 × 11 = 121.

.20 :302½.27 [SN] in Every TSQ... S. Punch: A early problem type would be to find good "contexts" for predn. This is an OZ problem. → 302½27

.01 See. 294.20ff ["T. problem of A.I. has been solved!"]

.02 302.40 [A (Naive) (not so naive) Main Path] : Make list of various Approaches to TM, a givet. difficulties, Bottlenecks in Each. || some refs: 294.20, 272.25-.30 (276.68: 3ways)

[311.10 253.84] I had been worrying about [Choice of a lang.] for Lsrch: Lisp like langs had apparent difficulty that they normally weren't able to ask for more bits in a defn. This can be dealt w. in Several ways: (1) I did have an elaborate system to get them to ask for more bits. (2) (new) One primitive asks to ask for another bit (3) Just use regular Lisp form, in which every finite string is a legal pgm. — T. pc of the primitives can be very context dependent (All possible numeric sizes of "contexts"...) — this is similar to (3). Assign pc's to pgms using some (post hoc optimized) function of pgm length — Not t. universal pc of t. integers! N.B. 294.00 a/o Note 294.14ff For t. "primordial TM" one quite used t. univl. d.f. on integers. [This may be about Same as .065] (.065)

(4) Using Forth as lang: One problem was construction of defns. Forth (& Lisp) have definition facilities but They are created as part of the devns of cands. A more useful kind of defn. is "post hoc": i.e., We just code cands & look for subtrees that have been useful — so that defining them compresses code. Superficially, it seems that this process would be much more useful in Lisp than in Forth. This is because Any subtree is a legit sub function. In Forth, I don't know what a legit subfunction looks like! (.30)

[SN] A ppl. weak pt. ("Needs work") BUG in my View of Functional langs: There can be a function S x,y ("STORE"): which stores x at address y. R, y ("retrieve") Then Retrieves content of memory y. The function S x,y produces no immediate output; Also if "R, 3", say appears in a formula, it doesn't necly mean same thing as "R, 3" appearing elsewhere in t. formula! S,x,y is a fcn w. "side effects".

IN Forth, we can use stack as a memory (tho its not such a good idea to!). So "3" puts 3 on t. stack. In OOPS What sort of Memory commands are there?

In OOPS t. closest thing to ["Memory" RAM]: Instr that exchanges m[th] & n[th] objects on stack; On instr that copys from a place on one stack to t. n[th] place on another stack.

On p 29: "B. A stack of integer arrays ... (not used in this paper)": This mite be used as ordinary RAM

25 Anyway: There are functions that use RAM in simple ways & I should try to characterize those functions; See if there is a way to break them down into "sub functions" (or sub...objects!).

26

I can treat this memory stuff .20-.26 as I'm treating "logical reasoning": Try to write TSQ's that don't use it, Then when I need it, work it out. (.25-.26 has an idea on how to do memory)

N.B. function defns, viz in 1 Series don't use > 3 or 4 registers: They don't use RAM to Generate t. functions?

29

.0 (.17) **Re Forth:** Try writing some reasonable functions in Forthish lang: See what kind of "Contexts" are useful for predn. See if (≈) PPM would be useful for predn. in these pgms. → (306.10)

.12 [SN] ON DEFINITIONS! T. advantage of Defns over PPM a its "improvements", is that when we compress using Defns, T. compressed Code is in a form that is likely to suggest new defns (compressions). I don't think this is true (or "as true") for ≈ "PPM". W. defns. We normally have parsing ambiguity, so we
(a) may want to reparse corpus after each (or after several) new definition(s)
(b) A/O use several hy pc parsings in ll & look for new defns in them.

4 TOM

20 : 303.40 : Normally in Lsrch, we have "Updating" in which the Guiding PD is updated in view of recent Problems Solns. Updating is usually the Most Difficult of TOM's activities. Just how can PPM MEJ (.015) be used in Updating the Guiding P.D. ? MEJ = Max Expected Jump

$\left[\frac{\sigma_G}{G_{max} - \mu_G}\right]$ ← MaxJump Expected

Rotsats: Seek D:\SS2 but mostly D:\PS Also special filter in D\PS in "Rotsats". Is this same as my characteristics on "Approx. Loops"

Well, we can use previously "successful" P.D.'s (OJ's ?) as corpus ... (possibly wtd):

PPM (either regular or using more complex contexts) can induce a P.D. on OJ cands & we can Lsrch them in PC order.

If we want a PD that involves Gore, we can use ... Max Expected Jump MEJ.

MEJ is our standard way to solve open problems — certainly not the Best, but "not Bad"

A better way would give a better PC of G as a funct. of dcm of cand, or would give ways to construct by Grandy cands that had hy pc of hy G. Or w. input Data of a set of cands & their G's: to make an s Grammar that assigns PC's to (cand, G) pairs. If we ask for a certain G, it will give a PD over cands w. that G.

Hvr, perhaps there is a more direct "object" or "system" to find optimus rapidly using past experience.

---

16 In MEJ: instead of making a Lang for each G-level, I do an interpretation & a Lang for all cands w. G's > X. So each X has a Lang. But This MEJ was Designed for G-A: It had to do w. μ & σ of offspring set pairs to learn from to Lang. — well, that was one applicn. There was a more direct applicn (I Think). I did work out that Applicn: I did not use PPM

19 I've forgotten how I did it! This uses PPM on a Grammar.

20 Say we use as corpus all cands w. G > X. This set has a $\bar{G} = \mu_G$ & a σ_G & a G_{max} — all as funcs of X: we pick an $X = X_0$ s.t. $\sigma_G/(G_{max} - \mu_G)$ is max.

As we evaluate cands, we keep track add cands between μ_G to X_0 to the Lang, & it's G_{max}, $\mu_G = \bar{G}$ changes. We also keep track of Those pruns for the "near by" Langs w $X = X_0 - \Delta$. & at $X = X_0 + \Delta$: When one of These get better MEJ's we change X_0 to $X_0 + \Delta$ or $X_0 - \Delta$.

25 So .16 - .19 was for normal GA w. Crossover mating.

20 - 25 was for "GA" w. PPM & no Mating or Mutation.

One Diffy w. Improving / Lsrch Updating: If we use set of previously "Successful" OJ for our corpus: We'd like to wt. the different cands, but it's not clear how! I'd like to associate each cand w. some empirical Gore .... but unclear as to how (inf. Initial QH, TSQ,) If I give wt. of 1 to all accepted cands, Then I can't do MEJ (.015), since all cands have same "Gore".

30 What we want is a good compressn method for the seq $O^j_1, O^j_2 \ldots O^j_n \ldots$ the seq. of successful cands. Actually, whatever must be optimum compression we "side info" — The seq of Q's is "side info".

D.1.04
4TM

0:302.13 | On "How many Coefs.?": $\sigma^2 \frac{n+m}{n-m}$ v.s. $\sigma^2 \cdot e^{\frac{2m}{n}}$ v.s. $\sigma^2 \frac{n+}{n-m} \cdot \frac{n+1}{n+1-m}$ ← Akaike: 298.00

One way to decide/check on my algebra would be to simulate $x_n = \sum_{i=1}^{m} x_{n-i} \cdot a_i$ + noise, and measure the observed errors.

An easy way would be to use "Consul compiler": it has matrix inversion; (The inverting matrix is faster: n^2 rather than n^3 ... it's still easier to use pre-programmed matrix inverter (?)).

Also Look at my formula for k coefs in Sol 78. — Also look at TM notes of that time to find out how I got those ideas: The formula in Sol 78 says $\sum(\text{error})^2 < c + \frac{1}{2} k \ln n$ — k = no. coefs, n is no. data pts. c = ln pc of derv of expression containing k params. I got $\frac{1}{2}$ from Rissanen, I think. The idea in S78 was that in turn 2 (cbits) k → $c + \frac{1}{2} k \ln n$ (in ln domain rather than binary logs). || Did I get this backwards maybe 2K ln n ← But proposed desc is not same as Akaike result of $\sigma^2 \cdot e^{\frac{2K}{n}}$ so pc of result is $\sim e^{\frac{n}{2\sigma^2(1+\frac{2K}{n})}}$ " $e^{-\frac{n^2}{2\sigma^2(1+2K)}}$

(a different "n" ☺)

$\frac{P_n(M)}{P_n(G)} > e^c$ $n^{-\frac{k}{2}}$

(p.2) **SN** on Lsrch v.s. **BU** search: Lsch is good when the trainer knows (or suspects) that there is a short code for the soln. that is within acceptable CJS. When this is not true, then BU is preferred. It is very impt that the cones used for both methods be accessible to one another as much as possl. In Lsrch, if TM gets a shorter soln. to problem than Trainer expects, TM should continue srch to find Trainer's expected solns. This is so we are sure that TM has acquired all of the concs that Trainer expects it to have, so Trainer can continue writing TSQ's w.o. trying to understand TM's "breakthrus". TM will always be able to solve Trainer's problems in time that Trainer expects, however. If TM is consistently finding signifly faster solns. that Trainer, it is clear that TM has "gotten ahead of trainer". Ideally, trainer should learn to understand TM's (better) solns — but this may be impractical & too time consuming.

The point of this "Trainerized" phase of TM's education is to get to the point where the trainer can devise training problems based on approximate models of TM's mind — Just as the trainer does w. human students.

Can we teach TM BU search using Lsrch? I am thinking of BU srch for OZ or IND problems as starting with a soln that is not very good, — but it has a G > −∞ (e.g. in the case of IND., a pc > 0). [or maybe a set of "not v.g." solns.] What TM has to lrn, is an Algm. for BU search, & this can be of acceptable CJs. One trouble w. this may be that such algm. tends (cands) are expensive to test.

Another (perhaps most serious) objection is that such algms are usually invented via logical analysis rather than simply doing algm. trials in sq. order. My unease at this pt. is because I haven't worked out details of how TM is taught to do logical reasoning. I had this idea that TM first lrns academic, abstract logic, — like it lrns Algebra. Later we teach it the relevance of the concepts in logic to problems in math in which "reasoning" is needed. So TM is eventually able to "reason logically" about anything in which such reasoning might be relevant.

N29-04 302½

4TM

Also 4TM 20.00 ! → **S·Functs :00** :38-40 is imp't.
274.07-.40

10 : on **S-functs: One good** way to get s-functs: Take a common "primitive" set of s-functs & take d functs of them: eg. say one primitive s.funct is uniform d.f. from 0 to 1.

Take T. Gaussian D.F. — integrate it so it goes from 0 to 1. Then, if we have a uniform D.f. from 0 to 1, the inverse of this will give us t. usual Gaussian D.F.

Any D.F. from −∞ to +∞ can be expressed in t. forgg. way ... via its integral.

What about discrete d.f.'s? If we have a discrete d.f. of k pts uniform betw 0 & 1, we can still use t. inverse of t. ∫ of any d.f. from −∞ to +∞. — I'm not sure that this is quite what I want!

I'd like a way to combine P.D's so I always get normed P.D's — consider how P.D.'s are combined, interact in RW. ! !

Given a uniform D.F. on k discrete pts, uniformly placed betw 0 & 1. — we can map them onto any other interval & preserve normality, since each of t. k pts will always have $pc = \frac{1}{k}$.

4 How can I get a non-uniform d.f. on k pts? Perhaps via Σ form, so each pt. is Σ of pc; of all pts of lesser i. So last $p_i = 1$. Any monotonic funct betw 0 & 1 will do this.

5 Then we map these k pts onto any other set of k pts. that we like. — So this gives all possl. finite discrete D.F.'s.

We can have an ∞ of pts betw. 0 & 1, & have a D.F. on them viz .14–.15

0 This ∞ of pts. can then be maped onto any finite or infinite interval !.

(.14R) or, Betw. 0 & 1 place k or ∞ pts. assoc w. spaces betw pts are intervals corresponding to pc of "higher" pt. so Σ pc = 1. We then map t. interval 0,1 onto any other interval — finite or infinite & we get assoc probs to the corresp. pt.

T. forgg. seems Nice! I want to Develope it more, — more examples! Make sure its adequate ... can represent all I want/need ... The "semi measure of ALP shouldn't give trouble ... it is normed if we allow t. "U" symbol. → 308.30

23 Another source of pc's is Tom Etter's "Join" function in EXCEL. See my files on Etters stuff; I have written a bit on it, a bit in "Notes on Etter": But much more elsewhere ... Maybe TM files??

27: 305.01 Also note 305.00–.01 ! ← This is Markov [sp?] Born, seq. prediction, looking for useful contexts; Making a lang. to generate contexts: Lang consists of "parts" of various types" and "Combination rules" of various kinds. Both are expandable. PPM is a first order fast way to do this. So a TM working problems of 0 this kind could learn to improve PPM! — A PPM is a component of TM_2.

31 Another common type of s.funct is a set of clusters: T. output of domain into be a single cluster center & a distance function from that center. The center could be a string (or a no. or a vector). The distance function could be at first certain standard types, then more tuned to the functions ssz & a/o search results "makers". We can also have >1 center: each with own pc. The most general case has very many centers (w. assoc pc's) & sharply pc's of non-centers that rapidly ↓ w. distance from centers. This is in direction of "3 up of univ" → Note 308.30ff

37

38) A Methodology for S funct invention: Give TM probs in TSQ. For each s-problem type, devise an s-funct form that works for that kind of problem. After many s-funct form types have been used, try to genz t. set to make it universal. → (Spec 308.00)

N 29:04
4 PM

00: 301.29: In my old Maxm writings, I wrote $\frac{n}{n-m}\sigma^2 \to \frac{n}{n-m}\frac{n+1}{n+1-m}\sigma^2$
Unclear if this is meant to be for R windo or X windo;
If it's for R windo, I expect the expression to be ∞ for m=n (which it does); but
it is also ∞ for m = n+1 — which is not so reasonable! — Tho perhaps one should
not consider m ≥ n at all!
Anyway, it would be relatively easy to get this experimentally, using μ (the true pdm)
as $\mu(x_n) = \sum_{i=1}^{m} a_i x_{n-i}$ + Gaussian noise.
I could test the pgm. w. noise = 0. Since the Matrix is Toeplitz, it should take only
10 ~ m² rather than m³ steps. Num Recipes has pgm for Toplitz inversion.
However, I may want a more general Matrix Inversion pgm: Partly "in general";
Partly for regression other than simple non-linear.
Perhaps write pgm in Cansol compiler (which I want to learn, anyway) — small cost
13 has a matrix inversion (of unknown Goodness!). → 303.00

Also Note 258.06-.22 on "Grand Plan" for Using GP for an i.e. TM
Spec.
20: 297.40: So the perhaps impt problems for a quick (maybe Phase 1) TM!
1) T 3+1 areas of diffys of GP on 272.25-.29
22 1) Faster methods of finding good solns: Use of PPM s/o Lsrch: Better Models: as results of TM looking at problems suggesting model classes. Also Attempts to approximate Gorc as a funct of cand params, when True, exact Gorc is very expensive.
In general, try to do this last, any way (even if Gorc is cheap), so, with a direct form of Gorc, we may be able to more quickly find peak! || Use of $\frac{\sigma_G}{G_{max}-\mu_G}$ as Gorc for a "language to search over". ↑ .225
2) Profit from solns to previous problems; be able to do tsq's: Also note .225
3) Beyond Gen. Alg. Be able to watch Gorc in action so as to understand why a cand was good/bad.
4) Get good, useful, searchable forms for s-functs.

Spec. 305.02

10 : from G & R S's p337 :

$(2n-1)!! = (2n-1)(2n-3)(2n-5)\cdots = 1\cdot3\cdot5\cdot(2n-1)$; $(-1)!! = (+1)!! = 1$.

$$\int_{-\infty}^{+\infty} x^{2n} e^{-\frac{x^2}{2\sigma^2}} = (2n-1)!!\, \sigma^{2n+1}\sqrt{2\pi} \;\Big\|\; \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{+\infty} x^{2n} e^{-\frac{x^2}{2\sigma^2}} = (2n-1)!!\cdot\sigma^{2n}$$

$x^0 \to \sigma\sqrt{2\pi}$

$\frac{\int x^2 e^{-\frac{x^2}{2\sigma^2}}}{2} \to \frac{\sigma^3}{2}\sqrt{2\pi}$

$\frac{\int x^4 e^{\cdots}}{4} \to \frac{3}{4}\cdot\sigma^5\sqrt{2\pi}$

$\frac{\int e^{\cdots} x^6}{6} \to \frac{3\cdot5}{6}\sigma^7\sqrt{2\pi}$

$\frac{\int e^{\cdots} x^8}{8} \to \frac{3\cdot5\cdot7}{8}\sigma^9\sqrt{2\pi}$

$\exp - \left(\int e^{-\frac{x^2}{2\sigma^2}}\left(\frac{x^2}{2} + \frac{x^4}{4}\right)\cdots\right)$

$\exp - \left(\frac{\sigma^3}{2}\sqrt{2\pi} + \frac{3\sigma^5}{4}\sqrt{2\pi}\cdots\right)$

$\approx \boxed{e^{-\sigma^3\sqrt{\frac{\pi}{2}}}}$ — only $\frac{x^2}{2}$ term used.

$e^{-\frac{\sigma^3}{2}\sqrt{2\pi}}$

[redacted] Second Moment of numerator $(\vec{x}\cdot\vec{y})$ is $A_2\cdot A_2 = \sigma^6\cdot 2\pi\cdot n$ (299.35)

first mom. of denom = $\vec{x}\cdot\vec{x} = A_2\cdot n = \sigma^3\sqrt{2\pi}\cdot n$,

Second " " " $= \vec{x}\cdot\vec{x} = A_4\cdot n = \sigma^5\cdot 3\sqrt{2\pi}\; n$

First Mom of Num = 0

Sec Mom of Num = $A_2^2 n = \sigma^6\cdot 2\pi$

Var of denom = Sec denom − (first mom)^2 — **Woops**: (divide by zeroth moment!) ← ? [$= \sigma\sqrt{2\pi}$]

Another woops! — in .01 if I want a normed d.f. for Gaussn. $\int_{-\infty}^{+\infty} e^{-\frac{x^2}{2\sigma^2}} = \sigma\sqrt{2\pi}$

So I should divide many nos. by $\sigma\sqrt{2\pi}$.

$x^2 \to \sigma^2$
$x^4 \to 3\sigma^4$
$x^6 \to 3\cdot5\cdot\sigma^6$
$x^8 \to 3\cdot5\cdot7\cdot\sigma^8$

first mom. of denom = ($\vec{x}\cdot\vec{x}$) $\sigma^2 n$ | ~~first Mom of~~ Sec Mom of Num = $\sigma^4 n$ | (here σ is the S.D. of the Mother Gaussian Noise)

Sec " " denom = $3\sigma^4 n$

Sec ~~...~~ (sec moment of denom) = $(3-1)\sigma^4 n = 2\sigma^4 n$.

20 $$\frac{1}{\sqrt{2\pi}\cdot\sqrt{2\sigma^4}}\int_{-\infty}^{+\infty} e^{-\frac{x^2}{2\cdot2\sigma^4 n}}\cdot\frac{x^2}{2} = \frac{3\sigma^4}{2} = \sigma^4$$

instead of the complicated multiplicative args of 300.-30 ff, consider the mean value of

$\frac{1}{1+x}$, in which x has the d.f. $\frac{1}{2\pi\sigma} e^{-\frac{x^2}{2\sigma^2}}$. $= \frac{1}{2\pi\sigma}\int_{-\infty}^{+\infty} \frac{e^{-\frac{x^2}{2\sigma^2}}}{1+x}\,dx$

$= \frac{1}{2\pi\sigma}\int e^{-\frac{x^2}{2\sigma^2}}\left(1 + x + x^2 + x^3\cdots\right)$ odd terms = 0.

$= 1 + \sigma^2 + 3\sigma^4 + 3\cdot5\sigma^6 + 3\cdot5\cdot7\sigma^8$. (see .01 R)

Some troubles w. forgg: 1) $\frac{1}{1+PP(x)}$: when mult by its numerator, its statistical nature mite have an additional

29 Effect... is like adding a PDS — adds more err's; adds more σ^2, 2) The Num & denom. are hily correlated.

30 : 2 impt ideas : ① How TM problem is already solved: (294.20) | ② 294.00 ff (which has worked well in past + Gauzn to → 302.00

N 23.04

4T M

~~24~~ 300

20: 299.40: $\int_0^\infty x^m e^{-\frac{x^2}{2}} dx = 2^{\frac{m-1}{2}} \cdot \left(\frac{m-1}{2}\right)!$ so for $m=2$: $2^{\frac{1}{2}} \cdot \frac{1}{2}!$ / $m=4$: $\frac{2^{\frac{3}{2}} \cdot \frac{5}{2}!}{}$

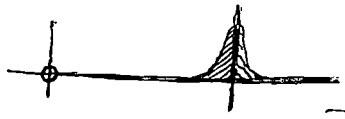
$\frac{m=4}{m=2} = 2 \cdot \frac{3}{2} = 3$. Yes!

I'm not so sure of (00)↓1, for $2n=m$! $m=1$ $\frac{1}{2 \cdot 2} \sqrt{\pi} = \frac{1}{2} A_2$

$m=2$ $\frac{3}{2 \cdot 2^2} \sqrt{\pi} = \frac{1}{2} A_4$

$\frac{A_4}{A_2} = \frac{3}{2}$. ← seems true! There was an error I made: top of p237 G & R table of S's

My representation of a Gaussian D.f. by 2 pts at ±σ may not be good!



$\int_{-\infty}^{\infty} \frac{dx}{1 + e^{\frac{-x^2}{2\sigma^2}}}$ ~ $\frac{1}{(1+f)^2}$ ~ $1 - f + f^2 - f^3 \ldots$

Integrs of even terms cancel.

$\int_{-\infty}^{+\infty} \frac{dx}{1 + f(x)}$ = $\int_{-\infty}^{+\infty} 1 + f^2 + f^4 \ldots$

12 $\int x^m e^{\frac{x^2}{2\sigma^2}}$ → ~ $\sigma^{2\left(\frac{m}{2}\right)+\frac{1}{2}}$ σ^{m+1} .30

$\int_{-\infty}^{\infty} e^{-p^2x^2} dx = \frac{\sqrt{\pi}}{p}$

$p^2 = \frac{1}{2\sigma^2}$

$\int_{-\infty}^{+\infty} e^{\frac{-x^2}{2\sigma^2}} dx = \sigma\sqrt{2\pi}$

Perhaps I try actually (linear regression w. known σ^2 of generator) to see if σ^2 is correctly predicted & "next case" is accurately predicted.

Try to see if Merkle's formula is better or worse than $\sigma^2 \frac{n+m}{n-m}$, $\sigma^2 \frac{1+\frac{m}{n}}{1-\frac{m}{n}}$.

i.e. compare $\sigma^2 \frac{1+\frac{m}{n}}{1-\frac{m}{n}}$ w. $\sigma^2 \cdot e^{\frac{2m}{n}}$

$\ln \sigma^2 + \frac{2m}{n}$

$\frac{n}{n+m}$ $\frac{1+\epsilon}{1-\epsilon}$ v.s $e^{2\epsilon}$ $\quad \epsilon\,\sigma^2 \cdot e^{\frac{2m}{n}}$

$\frac{1+\epsilon}{1-\epsilon} = (1+\epsilon)(1+\epsilon+\epsilon^2 \ldots$

$= 1 + 2\epsilon + 2\epsilon^2 + 2\epsilon^3 + 2\epsilon^4$

$e^{2\epsilon} = 1 + 2\epsilon + \frac{4\epsilon^2}{2!} + \frac{8\epsilon^3}{3!} \cdot$

$1 + 2\epsilon + 2\epsilon^2 + \frac{8}{6}\epsilon^3 + \frac{16\epsilon^4}{24}$

$1\frac{1}{3}$ $\frac{2}{3}$

So they differ at 3rd order! & much at 4th order.

Say $\epsilon = \frac{1}{2}, \frac{1}{3}, \frac{1}{4}$

ϵ^3 $\frac{1}{8}$ $\frac{1}{27}$ $\frac{1}{64}$

$\frac{2}{3}\epsilon^3$ $\frac{1}{12}$ ~$\frac{1}{40}$ ~$\frac{1}{100}$

Hmm, m can be close to n so ϵ ~ .8 Say, and $\frac{1+.8}{1-.8} = 9$ $e^{1.8} = 6$ (not that off)

So actually $\frac{1+\epsilon}{1-\epsilon}$ is a good approx of $e^{2\epsilon}$

$\epsilon = .5$ $\frac{1.5}{.5} = 3$ $e^1 = 2.7$

$\epsilon \doteq .9$ $\frac{1.9}{.1} = 19$, $e^{1.8} = 6$ for ϵ close to 1, these 2 methods give very different results.

.12 try $\prod_{i=-\infty}^{+\infty} \frac{1}{1 + f(i\Delta \cdot i)} \left(f(i) e^{\frac{-x^2}{2\sigma^2}} \right)$ = $\exp \sum -\ln(1 + f(\Delta \cdot i))$

$\ln(1+\epsilon) = \epsilon - \frac{\epsilon^2}{2} + \frac{\epsilon^3}{3} \ldots$ ≈ yes.

$\frac{d}{dx}(\ln x) = \frac{1}{x}$, $\frac{d^2}{dx^2} = -\frac{1}{x^2}$ at $x=1$ $x'=1$, $x''=-1$

No! it's $\prod \frac{f(i\Delta)}{1 + i\Delta} = \exp\left(\sum_{i=-\infty}^{+\infty} \ln f(i\Delta) - \ln(1 + i\Delta) \right)$

No! to get a value ≥ 2 small wts in a product, mult by $\geq \epsilon$, w. small ϵ.

So maybe $\exp \sum_{-\infty}^{+\infty} f(i\Delta) \cdot \ln(1 + i\Delta)$ = $\sum$ $e^{-\frac{i^2\Delta^2}{2\sigma^2}}$ $\left(i\Delta - \frac{i^2\Delta^2}{2} + \frac{i^3\Delta^3}{3} \ldots \right)$ rationing & counting.

$\exp \int_{-\infty}^{+\infty} e^{\frac{-x^2}{2\sigma^2}} \left(-\ln(1+x) \right) \doteq \left(-\frac{x^2}{2} - \frac{x^4}{4} - \frac{x^6}{6} \ldots \right)$

N22.04

4TM

2.90

10:289.40 : Consider 289.30 : given Y_i & model $z_i = M$ constant to find first $M \in \sum (Y_i - z_i)^2 = \min$.

This gives $M = \pm \frac{n}{n} \sum_{i=1}^{n} Y_i$: If we assume $[Y_i]$ ... was generated by a model of mean M_0 & Var $= \sigma_0^2$.

What is good estimate of σ_0^2? Working Backward: If we have a generator w. $\mu = 0$ & Var $= \sigma^2$,

Whatis most-likely (mean) σ^2_{obs} observed for n-cases? For true $\mu = 0$, M_{obs} will be $\approx \phi \pm \frac{\sigma}{\sqrt{n}}$

σ^2_{obs} will be $\approx \sum \frac{x_i^2}{n} - \left(\frac{\sum x_i}{n}\right)^2$: T. first term is $\sim \sigma^2$, T. second $\frac{\sigma^2}{\sqrt{n}}$ so $\sigma^2_{obs} \simeq \sigma^2\left(1 - \frac{1}{n}\right) = \sigma^2 \frac{n-1}{n}$

so $\sigma^2 \approx \sigma^2_{obs} \frac{n}{n-1}$.

Try for foreg. dgt for T. model of 289.32 : i.e. True model is $x\alpha = y \pm \left(\sigma^2_{\text{noise about zero}}\right)$.

First try simplified version of (033) (032) say $\alpha = 0$. What is now α_{obs} & σ^2_{obs}?

Find $\alpha \ni \sum (\alpha x_i - y_i)^2 = \min$. its min. $\sum (\alpha x_i - y_i) = 0$ for $\alpha = \sum y_i \Big/ \sum x_i = \frac{\bar{y}}{\bar{x}}$ no!

$\frac{\partial}{\partial \alpha} = \sum 2(\alpha x_i - y_i) x_i$

α tends to center around zero.

$\alpha \sum x_i^2 - \sum y_i x_i = 0$ so $\alpha = \frac{\sum x_i y_i}{\sum x_i x_i}$

$\bar{x}$ " " some value : say $\bar{x}$ is far

So, for normalized $\vec{x}^N$, $\alpha = \vec{y} \cdot \vec{x}^N$ = projection of ($\vec{y}$ on $\vec{x}^N$ or $\vec{x}$ on $\vec{y}$) ← No! $\vec{x}^N = \frac{\vec{x}}{\sqrt{\vec{x} \cdot \vec{x}}}$

$\alpha_{obs} = \frac{\vec{x} \cdot \vec{y}}{\vec{x} \cdot \vec{x}}$.

To get estimates of α_{ob} & of σ^2_{ob} we may need more Info about $\vec{x}$.

One kind of constraint: y_i & x_i are from a stationary Time Series, and say T. seq. is "stable" so $|\alpha| < 1$.

In .1IR α_{obs} would be ratio of 2 autocorrelations, $\Delta t = 0$ for numerator; $\Delta t = 1$ for denominator.

Say $\alpha = \phi$ & seq. is stationary. $\vec{x} \cdot \vec{x}$ will not be $\propto$ proportional to n, T. sum, since $\vec{x}$ is random Y_i is random wrt T. Y_i (distribution will be symmetric about ϕ & random wrt T. Y_i will be a random Gaussian if T. seq. is stationary).

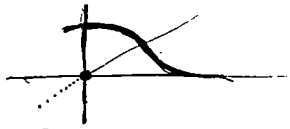
wrt. X_i, $\vec{x}\,\vec{y}$ will be a product of 2 random Gaussians; so $\vec{x} \cdot \vec{y}$ will be a normal D.F. w. $\sigma^2 \propto n$.

Woops! x & y will both be Gaussian d.f.'s of $\sigma^2_{y_i} = \sigma^2_y = \sigma^2$ of T. noise. So we know what T. d.f. of $\vec{x} \cdot \vec{x}$ and d.f. of $\vec{x} \cdot \vec{y}$ are: for .1IR, we can (easily?) find first & sec. moments of num & denom.

& denom, $\vec{x} \cdot \vec{x}$ is first mom $\propto n$; say $\alpha = \int_0^\infty z^2 e^{-\frac{z^2}{2}} \cdot z \, dz$ then $\vec{x} \cdot \vec{x} = n\alpha$.

ed. Fredkin

Clausing / Boon Chestnut Hill → tower → route 9 - Hyslop 166 :



All that I $\vec{x} \cdot \vec{x}$ need is first and second moments of $x \cdot x$ & $x \cdot y$ for $n = 1$.

first mom for $x \cdot x$: $\int_{-\infty}^{+\infty} z^2 e^{-\frac{z^2}{2}} dz$ (A_2) Second moment : $\int_{-\infty}^{+\infty} z^4 e^{-\frac{z^2}{2}} dz$ (A_4).

$n = 1$: $x \cdot y$ $\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (xy)^\alpha e^{-\frac{x^2}{2}} e^{-y^2} dx\,dy$ = [illegible] moment.

from 301.01:

$A_2 = \sigma^3 \sqrt{2\pi}$

$A_4 = \frac{3}{4} \sigma^5 \sqrt{2\pi}$

for $\alpha = 1$ first moment, $\int x = 0$ for $\alpha = 2$ i.e. $\int x^2 e^{-\frac{x^2}{2}} \cdot \int y^2 e^{-\frac{y^2}{2}}$ (A_2 · A_2)

so first mom of $x \cdot x = A_2 \cdot n$; second moment $x \cdot x$ is $A_4 \cdot n$

" " " $x \cdot y = 0$ " " $y \cdot y = A_2^2 \cdot n$.

Rough approxn : T. second moment of numerator is divided by $1 \pm \frac{\text{some constant}}{\sqrt{n}}$ world-wide Batter.

If z has var v, then kz has var $k^2 v$

The numerator denom is $x \cdot x = A_2 n \pm \sqrt{A_4^2 \cdot n} = A_2 n \pm A_4 \sqrt{n}$

$= A_2 n \left(1 \pm \frac{A_4}{A_2 \sqrt{n}}\right)$ (doesn't work for $n = 1$, or perhaps ≥ 2 ? maybe it does work ... but a bit informal.)

$\frac{x \cdot y}{x \cdot x}$ has var $\frac{A_2^2 \cdot n}{A_2^2 n^2 \left(1 \pm \frac{A_4}{A_2 \sqrt{n}}\right)^2}$

$= \frac{1}{n\left(1 \pm \frac{A_4}{A_2 \sqrt{n}}\right)^2} = \frac{1}{n - \left(\frac{A_4}{A_2}\right)^2}$

If would be nice if $A_4 = A_2$!

$\left| := \sqrt{(1 + \frac{\varepsilon}{n})(1 - \frac{\varepsilon}{n})} \right.$

$= \frac{1}{\sqrt{1 - \left(\frac{\varepsilon}{n}\right)^2}} \approx \frac{1}{1 + \frac{\varepsilon^2}{n}}$

— 3.00

11.24.04
4TH

Akaike.00

PS: Refs of int: Occ Razor in Physics 97.PS. 19/06/04: For discussions of "Source of Apriori"

Akaike's AIC 8/28/04 $AIC = \min(\ln \sigma_m^2 + 2m/T)$ or $\min T(\ln \sigma_m^2 + 2m)$ || $(\sigma^2)^T \cdot e^{2m}$

T = no cases = n; m = no. of coefs in (linear) regression. $\approx \min \sigma_m^2 (1 + \frac{2m}{T})$ ← Total sq. error

~~Since T is constant over it~~

On the other hand, I was using $\sigma_m^2 \frac{T+m}{T-m}$ ~~[illegible]~~ $= \sigma_m^2 \frac{1+\frac{m}{T}}{1-\frac{m}{T}}$

CRITICISM of Akaike formula! It does not "blow up" for m = T. $\sigma^2 \frac{T+m}{T-m}$ does blow up as it should.

taking ln we get: $\ln \sigma_m^2 + \ln\left(\frac{1+\frac{m}{T}}{1-\frac{m}{T}}\right)$ for small $\frac{m}{T}$ this is $\approx \ln(\sigma_m^2) + 2\frac{m}{T}$.

So AK's AIC is ≈ using k ∋ the expected error for next prediction is min.

So: How to reconsider $\left(\left(\frac{n+m}{n-m}\right) \cdot \sigma^2\right)^n$ with $\left(\frac{1}{\sigma}\right)^m$. (hvolume in m+1 space of coeffs plus σ)

or AIC → $\sigma^n \cdot e^m$

→ $(\sigma^2)^n \cdot e^{2m}$ ≈ $\sim \sigma^{-1} - m - 1$ ← anyway since it occurs in all choices to be compared. $(\sigma^{-1} \cdot e^{-\frac{m}{n}})^n$

---

could one get this e^{-m} at the "same place" I got ~~[illegible]~~ "spurious" e^{-1} in Z141 analysis? NB

Perhaps more derivation of $\frac{n+m}{n-m}\sigma^2$, we make some assumptions about the apriori's?

This result is σ^2 / a function of ~~[illegible]~~ time $\frac{m}{n}$, a dimensionless constant.

Actually, both m & n may be "dimensionless constants."

Consider coefs: for m coefs, there is a region in m space where σ^2 is within σ^2_{max} & σ^2_{max} × (1+δ). For linear regression, the coefs are all dimensionless. — So, in theory, one could compare the volumes of different dimensional spaces. Also note that coefs tend to be "~1" (or maybe $\frac{1}{m}$?) for linear regression.

~~[illegible]~~ This may be ok. for linear regression, in which all coefs are of about the same size. Hvr in linear Cum fit, this need not be so. The we can normalize the basis vectors

so perhaps coefs tend to be $\sim \frac{1}{m}$.

A Big Q is: Does this hvolume in m dim space approach give same results as $\sigma^2 \frac{n+m}{n-m}$?

The way it's done: for given m, there will be a vector $\vec{v}_0$ that give min σ^2. Other nearby values of $\vec{v}$ will give smaller σ^2.

A single codes is a set of coefs plus a σ^2. For each set of coefs there will be a σ^2 that gives max pc of corpus. We can assume a uniform apriori for σ or for σ^2 (I don't know if it makes much difference). We have n(+1) data pts + extra pt is the one predicted. To get the pdf for the new pt., integrate over all m+1 space.

Another way to do the prediction is to consider the pc of corpus assoc. w. all pts in m+1 space. Each point gets a certain wt. or the pc it gives to corpus. ~~Each~~ ~~Each set of coefs & σ^2 gives a prediction for~~ ~~the next pt~~ Each pt. gives a prediction for every possible value of the next pt.; to get the predn. of the entire set we average their ~~[illegible]~~ points predns. w. their wts.

We can also use sums of pts to ~~[illegible]~~ get wts & give predns. For each set of coefs, we can sum over wtd Σ of all σ^2 values in predicting n past data pts, & get a M, σ^2 & a wt. for that set of coefs.... which can then be used to get wtd. predns.

N1904

4TH

297

QUICK ABORT: .25

10 : In Sol 78 : I had proof that Cover's method of induction was worse than ALP by an ∞ factor (a "very small" ∞, hvr ∞) — This was for a version of Cover's method that used Goel's method of Norman.

02 I recently showed that for ALP, t. Norman constant is always bounded by $2^{\text{dc only and the true model of data}}$.

If .02 is true for Cover's method, then ALP must be better than Cover by this "small ∞" factor. (?)

Check this out!

On my web page, I could write "Comments" on Sol 64 v4: Levin's analysis: on "nonconvergence". Also his "discovery" that prefix sets must be used to get good pc values.

Also remarks on Sol 85 : That equation for when Singularity occurs! [illegible]

— how eq. was derived & what its exact closed form soln is. And more recent stuff on "Moore's" law. & Von Neuman's Remarks about "experts". → Explain what I mean by Moore's law i.e. halving of cc per bit every 1 yr or 1½ yr. is not limited by size or neccly by heat.

0 Also how singularity occurs if amt of money invested each yr. is bounded.

0 .294.40: pc's of ≈ "Best" cand thus far (when they had gotten as far as that particular problem)

Lets look at conventional QATM! Say it has "solved" 10 probs "Satisfactorily"! so it has at least one funct that can do all 10 probs w. "reasonable pc" for t. entire set of 10. We give 11^{th} Q.A.

23 We want a O^j for all 11 problems. We first try to get soln for QA_{11} alone & this is of satisfactory pc. (.25)

24 SN for each QA problem in a young TM: Trainer gives a pc level that it must achieve .. a minimum.

25 (.23) After solving $(QA)_{11}$ we test t. algm on previous problems: t. lowest geometric (scan) mean problem first & and then we test it on all other past probs in order of diffy of problem. We fail it if product

27 of prob's of solns thus far is "too low". ← (this "too low" level may be set by

28 trainer also, or be derived from thresholds Given by trainer (in .24)).

So t. large. is an Lsrch Soln, We mite also use the R (recognition function)

0 mechanism of IDSIA report.

My impressn is that t. "Quick abort" idea of .25 - .27 would quickly reject most cands that would eventually "fail". Hvr, in General, we don't want to have to test a new O^j on a t. entire past corpus. Usually t. modifns in t O^j will only effect recent problems. - - - -

35 We should be able to take advantage of this "fact" (if, indeed, it is true!). One way would be The "R" method. Could we somehow arrange for it to Discover t. "R" method? ... Suitable TSQ.

Also note that usually Lsrch will not devt. Quick Abort, because modfn of t. P.D. does not (normally) enable Quick abort. A Phase II Machine will look for good Optzn Methods & these must be able to consider all over such time & ∴ not be im poss. to discover. QuickAborts, → 302.20

Note 309.00

space. notsamp → 302½.20

293.00
10: [crossed out] : Hvr. one could get second derivative (Array/Matrix/Tensor) & a solve [crossed out] eq. to get best jump size. T. coefs of Tensor can be obtained partly by chain rule, partly by making several trials in various directions. Also, one mite assume all off diagonal elements are zero, or, that only t. elements within 1, 2, 3 . . . indices of diag. are non-zero; Or other assumptions, models, to reduce no. of elements needed. In a standard Non-linear Curve fitting techniques (Marquardt ... method) I think they assume off diag. elements are zero.

0 : (SM) Median Filters : use of "Min Median error" should be saved for SM: Perhaps start w. linear Models; Then use ANN. "Linear models" are not really linear because $e^{-|x|}$ error criterion (or $|x|$ error criterion), but we can start w. Regression coefs to fit for Mean sq. error criterion as first approxn.

In ANN, using Back prop, or — Marquardt method, t. $e^{-|x|}$ error produces no big change in Methodology.

— For "x window" a "median" of xpress $x_n, x_{n-1}, x_{n-2} \cdots x_{n-\infty}$ would find $\overset{M}{\bar{x}}$ ∋ $\sum_{i=0}^{\infty} |x_{n-i} - \overset{M}{\bar{x}}| e^{-i/\lambda}$ was minimized.

to obtain this $\overset{M}{\bar{x}}$, use successive approxns starting w. $\bar{x}$. It may be possible to show that in successive approxn, t. errors always ↓, so t. arith mean has largest error. this would enable one to know which of past data one had to refer (remember) in order to make corrections (over!).

0 Try using $\frac{1+\frac{m}{n}}{1-\frac{m}{n}}$ · (measured error) v.s. $e^{\frac{2m}{n}}$ (measured error) [A trials] for "Median prediction." — Empirical runs. Use generator of data in which t. "noise" part has a $e^{-|x|}$ distribn.

0

4TM Rev:

: This is a preliminary backward review from ~ 288 backwards.

277.20 to 288.20 & revisited on 291.00-.18

275.00 - 277.19 : Is a "Go" at the Z-141 paradox; Not carefully done, but may be essentially correct. I want over it more carefully.

274.07 - .40 Mainly on Problem of finding Good ways to represent S-functs: Some good Suggestions: Also 274.25: Links this problem to TSQ writing in a very useful way! This seems very imp. (particularly the link to TSQ's. It is an imp. part of the General problem of the form of TM & its TSQ.

273.26: Lists 3 Big Problems: (1) How to use coins to pass probs to help solve present problem.
(2) Search v.s. BU (Bottom Up) Search: If we do both simultly, how to best find conjs that may help each use.
(3) how to represent S-funcs: (See (.09) & 274.07 - .40 for good ideas on this.

272.30: If I had 2^k wks to decrypt TM, for k = 1, 2, 3 . . . . how would I do it . . . Say for small k?
I think this research started off the stuff that followed thru "Z-141" to 288.20
The "3 Big Probs" of (.09) were an imp part of that thread. - But I guess I forgot that
I was trying to solve 272.30! I eventually formed the "Search v.s. BU search" Question.

Note 258.06-.22 A "Grand Plan" for TM using GP.

Good! The stuff leading to 272.30 is nice!
272.21 review's pt. 6 needed improvements of GA's - Note "3 areas" of 272.29½

From ~265 ans, There was discn of ways to represent S-funcs:
Before ~265, Discussn of Search v.s. BU (Bottom-up = GA) search.
(266-267) is on Universal Analog Machines

"What has worked Well in Past" : .00 – .19.

4TM

.00 : "What has Worked Well in Past" This is a general method of getting & updating one's aprips. We have a particular algm, say, that tells how to get aprips for present problem based on success/failure of that assignment a priori method in previous problems. (Getting α of α_i of -1.36 would be an example). Each possible algm. will ↓ bc of corpus, until one adds in bc of } t. dcrn. of t. "How it worked well in past algm". So one must first measure just how much use of that algm ↓ bc of corpus.

Particular important uses of .00 ff: 1) Geometric Probability 2) How to compare n volumes in diff. nos. of dimensions of param. space (e.g. -1.34)

I want to make sure that I have a general, unambiguous, exact, effective, formal, practical

09 way to do this. (.14)

0 Perhaps Give Paper on (In California) .00 ff! "T. Univ. Dist and Aprips". : "Aprip in Theory and Practice"

The Computation of Aprips ~~Computing Aprips in~~ — Theory and Practice. In Search of A Priori — Theory and Practice

. Also Discuss Occ razor: When it works & when it doesn't work (reference papers) to when it doesn't work).

3 . See PS: 18/06/04: "Occ Raz in Phys":

4. (.09) T. problem of .00 is closely related to problem of creating trials for t. "latest task", using concs that "have worked well in t. past". T. usual problem in .00 is that we have forgotten just where & how we created t. models we are using & we don't remember t. param assignment algms that should have been "bundled" w. these models. Normally when a model is created, it's tested on t. past to see if it's a good model. If it's a good model & we decide to accept it, then we must have had techniques within t. model definition that tell how to test it — which is equiv. to knowing how to use that algm for prediction — either on old data (verifiable preda), or on new

14 data (not yet verifiable preda)

0 : A quick "take" on "T. problem of A.I. has been solved":

Consider t. problem of relating new GA/GP probs. to old: T. dcrn of t. latest problem is compared w. dcrns of previous probs that have been solved. T. problems that are "close" to present problem (all past problems have their terminal ≈ final population stored on disc) have their terminal populations mixed to form t. initial population for t. new problem. There are certainly other, probably better ways to do this. Weighting them & assigning params to them should be doable by (.00 – .10).

.26 For a "closeness" metric we can use PPM compression, or better (usually slower) compression

.28 methods.

Main Big Q's First Note: 272.25 – .29 (6 "improvables" in modern GP. — More like 3 general areas of improvement → see 272.29½)

→ Also, can it do TM w.o trouble? ← (This Q is mainly: "Can it do S-pred well?")

→ { I think a main Bottleneck problem was representation of S funcs: As is my models were not good enuf

0 to do Lsrch for QA problems or do much (BU) search in GP.

32 → Say I had a way to list S-funcs in ≈ pc order (writing a toolbar of a "say" of correct order): Key to goal on whether we're doing Lsrch of BU srch or BoB!

Still, I'd have problem of testing new cands on entire corpus. {Testing is not so hard for d-funcs because we can order cases in order of pc of failure, so cands fail quickly (cf. at all). ← Could we do anything analogous for S-funcs? — Well, YES! Put problems being tried in pc order (or L cost order?) w/ problem w. lowest pc first. "pc ordering" will be from mean pc over past cands. Reject or stop search on a cand when t. pc for entire corpus (thus far along list of probs by pc order) become < current threshold (Perhaps do p ← p/2 for pc's ... like in T ← 2T Lsrch). ← Actually, this doesn't seem to be such a hot way to do it! —

A perhaps, better way: Discard cand when pc for problems thus far is sufficiently less than → 297.20

Normalzn :00 || ANN.20

0 (Spec 290.40) 4 : Paper: "Notes On Normalization".

Gives results on 1) convergence of $\Sigma^2 P(U)$ to ? z·ln2·k.

2) That Normz constant converges. See 290.20-.40

3) That Normzd P.d. is always better than unnormed : $\Sigma (\ldots)^2$ is < by min of ln Normzn const.

4) That "usual" normzn function is Best

5) Discuss "by constant factor" as important in practical induction because the choice of reference UMC is not arby. (This Mis is dis cussd (to some extent) in my Kol 'talk. (paper).

.0 (267.06) ANN w.o. cross val. ① Pick random set of initial conds for a net.

② Use back prop. for k "seconds" (or other "count"). Do prediction w. resultant state.

③ Repeat (① - ②) l times:

23 ④ Take average of ~~resultsu.~~ predn. results: perhaps wtd average.

Q's. { Total cc ≈ k·l. Given a total cc (≈ k·l) available! How to allocate k & l. ??

or perhaps: to spend more cc on certain sub-runs?

Other ~~possible~~ Q's: how many params (wts) (Nodes) to use in net? what topology -- how many hidden layers, what population of hidden layers? perhaps in (.20) (= ①), item no. chose topology at random also. No. of params could be random, but

28 ~~small no. of par~~ in (.23) (averaging) small no. of params get more wt. (32)

These params, as well as various Coeffs that enable one to correlate the params w. initial problem data, can be solved by "what has worked well in past" (See 294.00)

30 T. advantage of the foregoing method over "Max flatness" is that larger space of "hypoths" is available. One could ↑ no of params in maxflat, but not as "wildly" (numerous. Also use several "maxflat" Models in parallel.

32 (.28) Another wting trick. make wt. funct of no. params, ms error on known Corpus,

— if error = 0, use "ratio" of hvol in params space to hvol in predn. space of covering "correct" predn. A Big Q is how to compare hvols of difrnt nos. of dimensions!

34 One way: To make sure sum of hvols converges! each extra dimension has factor, α of loss wt. ("hvol")

36 α is found by optimization of past predns. → see 298.01 Conceivable one can use a difrnt. α_i for each successive dimension, if one has a lot of "past data".

On ANN: (unrelated to immediate preceding stuff): Normally in "Backprop" they use various tricks ("momentum", "lrng rate") to decide on step size. T. direction of jump is known viz derivatives of wts wrt. Error. (Spec → 296.00)

N1404

4TM

: Notes on 2c4c :

1) One way to explain the problem: We have this corpus: (say Alphabet is a, b, c): we have two 2-grammars: Which is ~~more likely to be~~ true of pc of each generating the corpus? A sort of Bayesian problem: Th. 2-grammars: () is simple Bern w. a, b, c.

2) is ~~⇒~~ simple Bern w. abc_γ; ~~⇐~~ γ ≡ aabca. ~~⇒~~

Th. 2-grammars do not get here pc's for each of their choices. We assume uniform pb's ∀ ∈ over all possibs. Certain pc values will be more likely — but we are mainly interested in how pc's of the 2-grammars influence their pc's.

Essentially, we want to know the pc of deriving γ.

Re: Re "$\frac{1}{e}$ factor": 277.20 — 279.02 is a simple way to get pc of ~~the~~ γ, but it doesn't have $\frac{1}{e}$ factor! This treatment is incomplete because it does not consider the ↓ in pc of the Par (≠γ) symbols when γ acquires pc > 0. As soon as we introduce the ↓ of pc, we get the $\frac{1}{e}$ factor. 278.26 — 279.04 explains how $\frac{1}{e}$ arises.

4TM REV

: List of Problems: unsolved &/o "inadequately solved"

1) Z141: The 2 versions of behavior of system wrt. varying n! 285.31, 32
Also, my general feeling that Z^R is not so reasonable.

$\left[Z \equiv \left(\alpha e^{\frac{1}{\alpha}-1}\right) : \alpha \equiv \frac{R}{en} \right.$ ends to constant as n grows. $\left.\right]$

Note 99TM 1141
2V.g. review of work on Z141.

But in my letter to Wolff I said that I had some understanding of why Z should have that form: (i.e. $\frac{1}{e}$ factor)
See 292.10–.13 for discussn of how $\frac{1}{e}$ factor arises
I. main recent problem was that 285.31 & 32 differed by a factor of $\sqrt{n}$ a serious discrepcy!

It seems unlikely that the approxn. I used in 283.37 should be off by a serious factor!

2) "How Many Coeffs": 286.00; 287.00–288.20 I got close to a soln. to this problem but it does need work. The stuff I did in 1962 in Mex City is prob'ly relevant/helpful.

3) the problems of .01 arose when I felt that "coding w. definitions" did have some serious advantages over the method used in PPM & its genzns. 278.20 it was an attempt to put in more exact form the ideas of 278.23–.25 i.e. that 277.20–279.22 (278.00–22 in particular) Solved the "$\frac{1}{e}$ factor" problem in Z141 (apparently it didn't) → see 292.10–.13 for how $\frac{1}{e}$ arises. & the problem of making defns of ngms w. unusually low (or zero) frequency. I think this second idea didn't work very well either! → But it did explain the $\frac{1}{e}$ factor: see 292.10–.13 for discussn. See 279.10 for the beginning of this. Continued to 280.19 — 280.15–.19 is impt.! < (SN) on this: would parallel coding of "unusually low freq ngms" contribute further to the diffy w. this idea? >
Then unusually hi or low freq. of any ngm must lead to a shorter code; 275.15–.16

4) In my [illegible] Summing over all possl parses of γ (in Z141), I used the factor $\frac{R!}{(R-m)!\,m!}$ as the no of possys. Should it be $\frac{R!}{(R-m)!}$? (R is the no. of times γ occurs in corpus; m is the no of times it is considered to be γ, $0 \le m \le R$.
Also, I only summed codes w. m befo

MEM (Max Entropy) .00   Normalization : 20

.10 : T. 2 versions of MEM { simpler Max Ent. & Max Cross Ent. may both be [illegible] in terms of ALP.

For simple Max Ent. : Statement: we have a seq. w. $P(x_i)$ being unknown funct of past of x. Also constraints on $P(\cdot)$. MEM ~~[illegible]~~ in this situation, most likely $P(x_i)$ is specified by

.04 is that for which $\sum \ln P(x_i)$ is max. [.045] → This is seq. that is ~~[illegible]~~ least info. & best t. seq.

Hvr. .04 is not same as .045 (for [illegible]!). While min of .04 is correct, & same as Max likely hood; .045 is ≈ ALP, but is less clearly stated. — The ALP implies .04 is [illegible].

At any rate, if ALP trivially implies .045, then if $P(x_i)$ is a funct of both t. past of x_i & t. past of y_i (a II seq) then .045 ≡ most. Max Cross Ent. version of MEM.

.0 For both versions of MEM ⊂ T. norm (MEM is ~~[illegible]~~ special case of cross entropy) in which y_i = constant or null) if .045 implies [illegible] of .04, then perhaps t. corollary of t. ALP Convergence thm (sol 76) holds! I.e. thm - all dof. & [illegible] facts are known = min MS error via these methods! — I don't know if these facts are known. If they are known, then it may perhaps give a new proof of t. conv. thm.

.0 : ~~[illegible]~~ Corollary of Conv. Thm (sol 78): If $\frac{P(x(n))}{\mu(x(n))} > \alpha(n)$ ← for all sequences of length n. (P: string)

Then ~~[illegible]~~ (Expected value ~~[illegible]~~ (wrt μ) of $\sum$ sq. error of cond. pr. of successive bits, ~~[illegible]~~) $< -\ln \alpha(n)$.

If μ is t. generating D.F. & P is a universal D.F. (Norm or not norm),

Then ∃ that if μ had finite desc., ~~[illegible]~~ D bits then ~~[illegible]~~

$P > \mu \cdot e^{-D}$ for all poss. strings x_n. The Norm. constant (if P is unnormed) can never be $> e^D$, because if it were ~~[illegible]~~ t. normed P would be $> e^D \cdot P$ and $\frac{P}{\mu}$ would be > 1, which is impossible since μ is already normalized.

.0 **S**till, I must remember that each string $x(n)$ usually has a different Norm. Constant.

Then, in line w. .20 if $N(n)$ is t. smallest normalization constant for all strings of length n (~~[illegible]~~ for a given unnormed D.F. $P(\cdot)$), then $\sum$ sq. error in P_{Normd} will be < that for $P_{unnormed}$ by ~~[illegible]~~ $\ln N(n)$. (I'm considering a "norm. factor" to be ≥ 1 (by defn).)

.30 (.352) i.e. $\frac{P_{normd}(x(n))}{\mu(x(n))} > \alpha(n) \cdot N(n)$, (.37b) since $\frac{P_{normd}(x(n))}{P_{unnormd}(x(n))} \geq N(n)$ for all strings of length n.

obtained by multiplying (.37b) by .20.   See 293.00

4TM SM.00 (continued).

20 (Spec 288.40): On the other hand, if one considers predn schemes w.o. slippage, one can have very small daily yields, and still have a large amt of "α" to justify the predn scheme.

For a strat with consideration of slippage: Remove the slippage, compute α, then subtract the cost of the devn (in units) from lnα to obtain true yield: Then "subtract" slippage from that result.

The idea is that cost of devn of strat. comes out of the slippage-free yield.

It may, hvr, be poss. to do trades w. very low slippage — e.g. if one knows expected gain of very many stocks, one can look for "bargains" & buy/sell them.

At any rate, that the slip-free mult yield & the cost of strat. devn should be compared may be an impt. insight (if true!)

It is clear that a compressn. of sm. data of b bits is equivlt to a $\times 2^b$ slip-free yield, & that compressor devn. length must be subtracted from b.

10 An example: Carlos had a strat w. 2 params: Say 1 had 4 values, the other 3 values. One will run all 4×3 = 12 strategies & make each bet w. α amt. of mult & [illegible]. The W's are not updated simultaneously, i.e. each variant of the strat bets at different times (usually), But at each point, each strat has (cost) a certain mult. amt. of money, & that's the fraction of money one bets on it. The money actually bet on a strategy is a separate account from the mult yield of that strat.

Hvr, note that w. 12 variants, if one is much better than the rest, we still have to divide its yield by 12.

Actually, we need only keep track of the individual mult. yields of each (offline) individual strats: The final yield will be $\frac{1}{12}$ × sum of mult. yields.

Is there a way to use the info that certain params are "close to" others? Well one could divide each param. into 100 divisions & follow them as before! There would be many strats w. hy yield near peak.

0 (288.20 Spec): I think the toughest part of the problem is: for a given $\vec{M}$ and $\vec{Y_0}$ How does ms error of predn (w. [illegible] $\vec{Y_0}$) vary w. size of noise, N. I.e. True Variance of data $\vec{Y_0}$ is vare of $\vec{N}$. But best $\vec{C_0}+\vec{\epsilon}$ gets a smaller vare. for its predns.

Wow! I've been working on curve fit! Noise is added to predns only. In linear Regressn noise is added to data also. I'm not sure it's my problem, hvr. It seems like a convenience to have the $\vec{\vec{M}}$ values indep of N, but I'm not sure it's really needed for the analysis.

7 → Note: Perhaps try as "study problem": Proving that $\sigma^2 \to \frac{n}{n-1}\sigma^2$ Goes to $\sigma^2 \to \frac{n}{n-m}\sigma^2$: Then I think the other [illegible] factor, follows, perhaps from first factor: i.e. $\sigma^2 \to \frac{n}{n-m}\cdot\frac{n+m}{n}\cdot\sigma^2$.

Perhaps .27 can be found in any Statistics book! Maybe look at Rissanen.

7 Way to do .27: First do T.S. w. 1 param. to approx it: so we want M of seq. given $x_1 \ldots x_n$ to find $\mu \ni \sigma^2 \sum_{i=1}^{n}(x_i-M)^2$ is min. Show how σ^2 of [illegible] $= \sigma^2\cdot\frac{n}{n-1}$

31

32 do same in which model is $\sum(\alpha x_i - y_i)^2 = \min$. Note that this is same as .31, if $x_i = 1$ for all i.

$\sum(\alpha x_i - y_i)^2 = \min$

Finally show it for 2 coeffs $\sum_{i=1}^{n}(\alpha x_i + \beta z_i - y_i)^2 = \min$. This is presumably easily genzd to an arbitrary no. of $(x_i, z_i, w_i \ldots)$ vars.

Spec 290.20

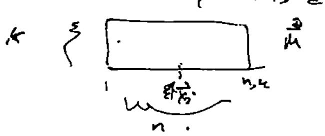
N4.04

4TM

SM .21

22408

20:287.40



Say $\vec{M}$ is a $\underline{n}$ col. by $\underline{k}$ row vector matrix $\vec{x}_i$ data ← Data array; $i = 1/n$;

[illegible] $\vec{M} \cdot \vec{c}_0 = \vec{y}_0$; $\vec{y}_0$ is noiseless Data

$\vec{c}_0$ is the true param vector: $\vec{y}_0$ is the true set of predn (w.o. noise)

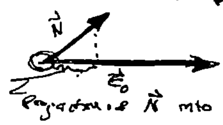
$\vec{N}$ is noise vector so $\vec{y}_0 + \vec{N}$ is the observed data.

Our estimate of $\vec{C}_0$ will be $\vec{C}_0 + \vec{E}$ which $\vec{M} \cdot \vec{E} \simeq -\vec{N}$ as closely as possl.

i.e. $\vec{M} \cdot (\vec{C}_0 + \vec{E}) = \vec{y}_0 + \vec{N}$ w. min total sq error so $\left(\vec{M} \cdot (\vec{C}_0 + \vec{E}) - (\vec{y}_0 + \vec{N})\right)^2 = \min$ so $\|\vec{M} \cdot \vec{E} - \vec{N}\|^2 = \min$.

Consider $\vec{N}$ to be an a vector in an n space: we want to approximate $-\vec{N}$ by a linear combination of the k n vectors that are the rows of $\vec{M}$. The k rows of $\vec{M}$ give a k dim subspace: $\vec{E}$ will be in that subspace ~~[illegible]~~ and $\vec{E}$ will be the projection of $\vec{N}$ into that subspace (??).

for N=2, K=1



Projection of $\vec{N}$ into subspace of $\vec{C}_0$

Getting the Volume of the set of row vectors is "Murray's problem". — Solvable by Gram. Schmidt meth. So to get the projection of $\vec{N}$ we first make an orth norm basis for the k, n vectors of M. From the basis, we get the projection of $\vec{N}$ — then we express that projection as a linear combination of the rows of M. (There may be a more direct way). We may be able to run Gr. Schm backwards on

16 $\vec{N}$ to get the expression of $\vec{N}$ in terms of the k row vectors of $\vec{M}$. $\vec{N}_0 \equiv \vec{N}$

17 A Gram Schmidt inspired way to get projection of $\vec{N}$ onto $\vec{M}$: For the first row vector $\vec{M}_1$ remove from $\vec{N}$: $\vec{N}_1 \to \vec{N}_0 - \frac{\vec{N}_0 \vec{M}_1}{|\vec{M}_1|} = \vec{N}_1$

18 $\vec{N}_{i+1} = \vec{N}_i - \frac{\vec{N}_i \cdot \vec{M}_{i+1}}{|M_{i+1}|}$ ← Normalization. The sets of scalars, $\frac{\vec{N}_i \cdot \vec{M}_{i+1}}{|M_i|^2}$ defines the projection of $\vec{N}_0$ onto $\vec{M}$.

Actually, 17–18 is a way to find the soln $\vec{C}_0 + \vec{E}$ for the data $\vec{M}$, $\vec{y}_0 + \vec{N}$ → 289.20

1 SM In auto prediction (1 stock) select model on basis of min $\sigma^2_{obs}\left(\frac{n+k}{n-k}\right)$

Vary both n & k in choice of model; Prob'ly best to choose a wind which has somewhat different formula: See TMW 1960, ect.

In cross prediction (more data in simple auto predn) an additional param is "size of population" that one selected "best" of. This is related to, but not identical to k.

SOY problem. (Spurious Optimization Yield). It's more impt in cross predn because the no. of pairs in a populatn is >> the population itself.

In using ∥ (parallel) (pc wtd) models: does SOY still occur? — There is no optza selection, I think.

Soy is for pure OZ probs only. If we do SM by predicted d.f., then use (Gamb. Ruin Modified Kelly) we can avoid Soy. (Probably ... I'm not abs sure). It does mean we cut ourselves off from many interesting SM strategies!

T. older argt: that any ~~[illegible]~~ legit strategy implies a pd ont. corpus → that strat would work. any pd. ont corpus is a coding technique & is subject to "cost of code" ~~[illegible]~~ constraints. I think both slippage & slippage free systems can be treated.

Anyway, say a strategy is able to give mult. of capital by factor α in one year, using Kelly betting or whatever. Then it should be equivl. to compression of ln α nits. The stock time series may have 10k bits/yr, so the 2 bits/yr. that we get from the strategy is quite small, so we can't learn much into the strategy design.

spec. → 289.00

11/3/04

4TM How Many Coifs (cont) Seems like V.G. fine (Soln. for linear/non linear: curfit & regressn)

RASHK: "Needs work"!

Spec OP: 2.86.96: Ord. linear regressn! A complete descn of corpus is ① first k values, k coifs, value of σ^2. From this info, one can assign a pc to each value ($k < i \leq n$) and to any proposd n+1th values.

Say we have a [illegible] set of k+1 params. How much info is in them?

Actually, we don't care! We just use all sets of k+1 params in ll codes & wt. them for optimal predn.

.04 This may give the $\sigma^2_{observed} \times \frac{n+k}{n-k}$ vars about the mean pred of the "center" code (≡ "best" code).

The d.f. in the a coifs & in σ^2 space are rather sharp — so perhaps the spray of the coifs & σ^2 are not very important. Each code has some pc but is mult by its accuracy (≡ "wt."),

07 to give the final distribn. Since we sum over all codes, the p cost of that part is ϕ!

.073 is true for various values of k, so we really don't care about the value of k!

0 — T. width of σ^2 is also unimportant. T. only impt thing is t. central predn & its width — which is (perhaps) $\sigma^2_{obs} \frac{n+k}{n-k}$ (w. modifns for x windo).

Should we give = wt. to the predn for each value of k? (for $k < k < n$)

∴ ($k =$ means σ^2 exists & we have many w. that varc.)

While each value of k has a predn of its own wt., we still have to give t. value of k to specify each set of all codes. Since $1 < k < n$ we can use a uniform d.f. — unless we have found in t. past, that for seqs of t. present type, some other d.f. on t. values of k gives better predns.

7 .00 — .17 Seems to solve t. HMC problem for linear regressn. It will prob'ly work for curfit & non-linear regn. & curfit if t. σ^2 is small enough so that t. system is locally linear.

T. unproved part is $\sigma^2 = \sigma^2 \frac{n+k}{n-k}$: I have proved it for $k=1$ & it's prob'ly true for $k > 1$. One way to do it is to consider ll codes as in .00 .04

Another unproved part: (possibly not exactly true!). I assume that the peak (for a given k dim space & σ^2 space) is rather sharp, so that averaging over those codes is very close to using the peak k vector, σ^2 code. Actually, it's More Complicated.

For each k vector we get a d.f. on σ^2's — i.e. t/ pc using those corpus σ^2's. It would seem that t. k+1 dim. "width" of that d.f. should be t. wt. of that k vector & σ^2 peak.

As I see it now, for each value of k, we have a d.f. on coif, σ^2 space.

Say $\vec{kC}$ a set of k coifs & σ^2 is an assoc varc. For each value of $\vec{kC}$ & σ^2 we have

10 — a pc for t. corpus & a wt. w; we also have a pd on t. predicted element & that d.f. will have wt. w.

— The d.f. on "x_{n+1}" t. next element, will have varc $\equiv \sigma^2$

.2 — So, each $\vec{kC}, \sigma^2$ will have a wt. (≡ pc of corpus) & a mean & σ^2 for predn of x_{n+1}. We obtain final d.f. on x_{n+1} predn by t. wtd mean of all $\vec{kC}, \sigma^2$'s of .32

The wtd ≡ of a bunch of normal Df's is a normal D.f.? No! But in many cases, yes. i.e. we can get t. μ & σ^2 of any bunch of (Df)$_i$ ... we have t. μ_i, σ_i^2 of each (d.f.)$_i$

If μ_i, σ_i^2 has wt. w_i, [illegible] then $\bar{\mu} = \sum w_i \mu_i / \sum w_i$; $\overline{\sigma^2} = \sum w_i \sigma_i^2 / \sum w_i$?

Needs work! — Maybe we have to do: for each $\vec{kC}$, first get a wtd sum of all of its poss'l σ^2 values. This gives a single σ^2 (& wt) for each $\vec{kC}$. We then take a wtd ≡ of all $\vec{kC}$'s & their assoc "summed" σ^2's.

N/2/04

4TM How Many Coeffs; Coos = HMC .00 f.b of 2 [67.00 - .17 seems to be FINAL SOLN] Apparent
e.g. ANN ← (in conflict w. Barron 1993).
(for non. linear regn)

00: : HMC : For linear regressn : (Non-linear reg. is locally linear when σ^2 is small (ent due to
01 large ssz.) . The formula $\sigma^2_{true} = \sigma^2_{apparent} \cdot \frac{n+k}{n-k}$

prob'ly looks. To do a prediction, LDL would chose k w. min σ^2_{true}.

ALP will wt. predns for all $0<k<n$.

To show .ok for $k=1$ is easy: To show it for $k>1$, use K. dim space w. the "correct" set of coeffs at the origin. Any deviation from the set of coeffs at the origin: ($\equiv \vec{D}$) will give an obsered error of $|\vec{D}|^2$. If σ^2 is true σ^2 for true coeffs, then using a finite ssz $<\infty$ ($n<\infty$) will give a smaller $\overline{err}$ due to overfitting.

~~Then since the~~ From this observed $\overline{err}$ one can find $\sigma^2_{true\ for\ generator}$. What one needs here, is σ^2 for chosen set of coeffs which is an additional amt >0 since the chosen set of coeffs $\neq \vec{\Phi}$.

$\sigma^2_{obs} \cdot \frac{n}{n-k}$ gives σ^2 of true generator. This has to be ↑ by factor $\frac{n+k}{n}$ to give compensation for fact that the set of coeffs chosen is not the ~~true~~ set of coeffs of the "True generator".

I may be able to do this more exactly: 2 ways:

1) simply vectors in n & K space & Gaussn D.F.'s.

2) In K space, integrate over all poss. sets of coeffs (& correct ~~set~~ one is M), ~~[illegible]~~ using wt. $\propto e^{-\sigma^2\ for\ that\ set\ of\ coeffs}$. I perhaps saw notes of ~~1960~~, Mexico City

---

If the form. is a really correct ALP soln, how does it fit in w. the idea of each extra coeff having a certain (standard) extra post? Also how does it deal w. the (about the same problem) extra bc as f assoc w. using ~~an~~ one more dimension in n dim space?

Perhaps "another" approach: linear regressn as a special case of curve fitting.

For each k there will be some optimum set of coeffs — a k dim vector $\vec{O}$. of observed output is $\vec{O}\vec{X}_i + N_i$ (k vector). $\vec{X}_i$ is i'th set of k data pts.

Say $\vec{A} = \vec{O} + \vec{R}_A$ ($\vec{A}$ is an approxn to $\vec{O}$). Then $\vec{A}\cdot\vec{X}_i = \vec{O}\cdot\vec{X}_i + \vec{R}_A\cdot\vec{X}_i$

So random k dimensional noise

Then we find Σ err^2 of $\vec{O}$ vs. that of $\vec{A}$. We want Misymbol of $\vec{A}\cdot\vec{X}_i - (\vec{O}\vec{X}_i + N_i) = \vec{R}_A\cdot\vec{X}_i - R_i$

$(\vec{A} - \vec{R}_A)\vec{X}_i = -R_i$. (I'm confused!)

On. derivns of n point corpus w. k point models: The relative wts are $\propto$ the density ratios in K & n space. A good code maps a broad distribn (low info) into a ~~narrow~~ narrow D.F. The σ^2's of the D.F.'s have to be dscbd, but since we are only interested in ratios of p c's of σ^2's, it's a bit easier! We know more or less "quantitative" sizes of σ^2 of output.

Still, comparing densities in 3 vs. 4 dim spaces (n spaces): meaning is not clear!

0/31/04

10 : $\left(\frac{m}{n-m\ell}\right)^m = m^m \cdot (n-m\ell)^{-m} = m^m n^{-m} \left(1-\frac{m\ell}{n}\right)^{-m} \approx \frac{m^m}{n^m} e^{+\frac{m}{n} m\ell}$

Let's look at $e^{\frac{m}{n} m\ell}$: on 284.20, we have an expression for R^{th} power.
we want to see how → Compares w. it. with $\left(z e^{\frac{1}{z}-1}\right)^R$ $z \equiv \frac{R}{\varphi n}$

03 ; $0 \lesssim m\ell \lesssim R-\varphi n$ | $z-1 = \frac{R-\varphi n}{\varphi n}$ | $e^{\frac{m}{n} m\ell} = \left(e^{\frac{m\ell}{n}}\right)^m$ (.03R) $0 < \frac{m}{n} < \frac{R}{n}-\varphi$
└ m is the exponent, R

$e^{\frac{m\ell}{n}}$ | from .03R $\frac{m}{n} < \frac{R}{n}-\varphi$; which tends to be small as φ is small. ; hvr $\ell\left(\frac{R}{n}-\varphi\right)$?
How does $\ell\left(\frac{R}{n}-\varphi\right)$ compare w. $\left(z e^{\frac{1}{z}-1}\right)$? See (284.20R) for sizes of z.

If ℓ is very long, maybe $\ell\left(\frac{R}{n}-\varphi\right)$ could be significant i.e. ~ 1.

But look at 282.36 $\left(\frac{m}{n}\right)^m \left(1-\frac{m}{n}\right)^{n-m\ell}$: where do these factors come from?

0 $\left(\frac{m}{n}\right)^m$ is the pc of the δ insertions: $\left(1-\frac{m}{n}\right)^{n-m\ell}$ is the ↓ of pc's of former symbols of corpus.
first factor; more exactly: m insertion is a total corpus long the of R $n-m\ell+m = n-m(\ell-1)$

!2 so $\boxed{\left(\frac{m}{n-m(\ell-1)}\right)^m}$ | for $\left(1-\frac{m}{n}\right)^{n-m\ell}$, the correction is more complex: The reasoning was we have a new symbol w. pc = $\frac{m}{n-m(\ell-1)}$

So we assume all old symbols have their pcs ↓ by factor $1-\frac{m}{n-m(\ell-1)}$ — and there are just $n-m\ell$ old symbols.

4 so the second factor is $\boxed{\left(1-\frac{m}{n-m(\ell-1)}\right)^{n-m\ell}}$

Note that the expressions of .12 & .14 are more symmetric. The sum of the exponents is $n-m\ell+m$ — as desired.

The product of the 2 is a common function of $\overset{x}{m}$, $\overset{y}{n-m\ell}$: $\left(\frac{x}{x+y}\right)^x \cdot \left(\frac{y}{x+y}\right)^y$.

$\frac{x!\,y!}{(x+y)!} \approx \frac{x^x y^y}{(x+y)^x \cdot (x+y)^y} \cdot \sqrt{\frac{2\pi\, xy}{x+y}} = (\downarrow) \cdot \sqrt{\frac{2\pi}{\frac{1}{x}+\frac{1}{y}}}$

.0

One error in .12, .14 is that when δ is defined, not all old symbols have their pc's decreased by same factor. The symbols in δ have their pc's ↓ more than others. Symbols not in δ may have their pcs ↑, since corpus is shorter, yet their case counts are larger. The "factorial" treatment of ≥ 141 respects this fact better.

Looking at 99 TM (Z141 122.08) ratio of ↑ of pc due to each:

$\frac{\beta+1}{\beta+\ell+1} \quad \frac{\beta}{m} \quad \psi\sqrt{\epsilon m} \quad \sqrt{2\pi} \quad \underbrace{\left(\frac{\epsilon}{\psi} e^{\frac{\psi}{\epsilon}-1}\right)}_{\equiv Z}{}^R$

R is case count of δ. (δ has no repeated symbols, cannot overlap itself) $\epsilon \equiv \frac{R}{n}$, $\psi \equiv \varphi$

0 for $\frac{\epsilon}{\psi}$ close to 1 $\left(\frac{\epsilon}{\psi}-1\right)^2$ $Z \approx 1+\frac{\left(\frac{\epsilon}{\psi}-1\right)^2}{2}$ (for small $\frac{\epsilon}{\psi}-1$) $\frac{\epsilon}{\psi} \rightarrow$ "z" of 284.20
— I guess we assume $\frac{\psi}{\epsilon} \gtrsim 1$, here.

31 ibid (122.10) $\frac{\sqrt{2\pi}\ \beta+1}{\beta+\ell+1} \psi \cdot \epsilon \frac{Z^R}{\sqrt{R}}$

32 4TM 284.21 gets equivalent of $\frac{1}{\sqrt{\pi}} \cdot Z^R \sqrt{\frac{\frac{\epsilon}{\psi}}{\frac{\epsilon}{\psi}-1}}$ $\cdot \sqrt{\frac{\epsilon}{\epsilon-\psi}}$

The big difference betw. .31 & .32 is the $\frac{1}{\sqrt{R}}$ factor : its quite large $\frac{\epsilon}{\psi}$, but presumably (Z^R) is dominant. Compare behavior of the 2 approaches w. fixed ϵ & ψ as n ↑.

.31 is $\propto \frac{c}{\sqrt{n}}$, .32 is not $\propto \frac{1}{\sqrt{n}}$

0/29/04
4 PM

! Since m of interest is mainly by m, we can use $\frac{1}{\sqrt{2\pi m}} \to \frac{1}{\sqrt{2\pi A}} = \frac{1}{\sqrt{2\pi(R-\varphi n)}}$ as a correction factor.

We need to find correct term

$$\frac{R!}{(\varphi n)!\,(n\varphi)^{R-\varphi n}} = \frac{R!}{\varphi n!\,(n\varphi)^{R-\varphi n}} \text{ mult by } \sqrt{2\varphi n} \text{ and } \frac{1}{\sqrt{2\pi(R-\varphi n)}}$$

$$Z = \frac{R!}{2(\varphi n)!\,(n\varphi)^{R-\varphi n-\frac{1}{2}}}\,\frac{1}{\sqrt{\pi(R-\varphi n)}}$$ which is ~ same as 282.36

Next, I want to look 28.06 in terms of $\left(\frac{R}{\varphi n}\right)^R = \alpha$

$$Z = \frac{R^R}{e^R\sqrt{2\pi R}(\varphi n)^R\frac{\varphi n^{\varphi n}}{e^{\varphi n}}}\frac{\sqrt{n\varphi}}{\sqrt{\pi(R-\varphi n)}\ \varphi}$$

$$= \frac{R^R\ e^{\varphi n}}{e^R\ (\varphi n)^{\varphi n}\ (\varphi n)^R\ (\varphi n)^{-\varphi n}}\ \frac{\sqrt{2\pi R}\ \sqrt{\varphi n}}{\sqrt{\pi(R-\varphi n)}\ \sqrt{2\pi\varphi n}\ \sqrt{2\pi}}$$

$$= \left(\frac{R}{\varphi n\cdot e}\right)^R\cdot e^{\varphi n}\cdot\sqrt{\frac{R}{\pi(R-\varphi n)}}$$ | $R-\varphi n \equiv M = m_{max} = A$

$$= \left(\frac{R}{\varphi n}\right)^R\frac{1}{e^{R-\varphi n}}\sqrt{\frac{R}{\pi M}} = \left(\frac{R}{\varphi n}\right)^R\cdot e^{-M}\sqrt{\frac{R}{\pi M}}$$

$$\to \left(\frac{R}{\varphi n\ e^{1-\frac{\varphi n}{R}}}\right)^R$$

$$= \frac{1}{e^{n(\frac{R}{n}-\varphi)}} = e^{(1-\frac{\varphi n}{R})\cdot R}$$

expected α

R−φn

$$\left(\frac{R}{\varphi n\ e^{\frac{\varphi n}{R}-1}}\right)^R$$

First write Brent's nearly proven analysis of conventional Z(4) using factorials.

$\frac{R}{\varphi n} = Z$ | $\left(Z e^{\frac{1}{Z}-1}\right)^R\cdot\sqrt{\frac{R}{R-\varphi n}}$ | $\sqrt{\frac{1}{1-\frac{\varphi n}{R}}}$ | $\sqrt{\frac{1}{1-\frac{1}{Z}}} = \sqrt{\frac{Z}{Z-1}}$

$$\to \sqrt{\frac{1}{\pi}}\ \left(Z\ e^{\frac{1}{Z}-1}\right)^R\cdot\sqrt{\frac{Z}{Z-1}}$$ (looks familiar!)

This looks like same deficit as before! i.e. $\frac{1}{e}$ penalty!

The $\sqrt{\frac{Z}{Z-1}}$ factor is peculiar! for Z close to 1, it becomes very large,

It came out as (∞)? $\frac{\sqrt{2\pi R}}{\sqrt{2\pi A}}$

The $\frac{1}{\sqrt{2\pi m}}$ factor is from 283.04 $\frac{m^m}{m!} \to \frac{e^m}{\sqrt{2\pi m}}$ — this is cancelled by another factor

Z	$Z e^{\frac{1}{Z}-1}$
1	1
1.1	1.0042
1.5	1.074
2	1.21
3	1.54
5	2.246
7	2.97
20	7.73
∞	$\frac{\alpha}{e}$

Z<1 is not relevant

Q: in 283.37 I say $L(1+\frac{1}{\alpha})\cdot L(1+\frac{2}{\alpha})\cdots$

Maybe should be $L(1+\frac{\varphi}{\alpha})(1+\frac{2\varphi}{\alpha})\cdots$ No

Check formula at 283.22 & .31 simple recursive formula for $\frac{R!}{(R-m)!\,\varphi n^m}$:

f(0) = 1 $T(m) = \frac{R-m}{nf}\cdot T(m-1)$

f ≡ φ

nf = 1/nf
nfi =
R =
[later, include "Q" as input variable]

B = (R−nf)

X = 1/sqr(2p) — type then.

S = 0

For m = 1 to B:
X = X·(R−m), nfi :
S = S + X * sqr(m)
N...

Print S.

9/29/04
4 PM

?! very appreciable if $m \ll R$

.00: for $m \gg 1$, lots of cancelation in 282.36 terms! If $m \ll R$ which is common!

.01 $\frac{m^m}{m!} \approx \frac{e^m}{\sqrt{2\pi m}}$ $\frac{1}{n^m \varphi^m}\frac{R!}{R-m!} \approx \left(\frac{R}{n\varphi}\right)^m = \left(\frac{R}{n\varphi}\right)^m$ $\left(1-\frac{m}{n}\right)^{n-m\ell} \sim e^{-m+\frac{m^2\ell}{n}}$ (.30 gets rid of this)

So we get $\frac{1}{\sqrt{2\pi m}}\left(\frac{R}{n\varphi}\right)^m \cdot e^{\frac{m^2\ell}{n}}$

key factor isn't trouble: as $m\uparrow$ we mult preceding factor by $\frac{R-i}{n\varphi}$.

until $R-i = n\varphi$: i.e. factor is 1. Then $\frac{1}{\sqrt{2\pi m}}$ factor

.09 Re : $e^{\frac{m^2\ell}{n}}$ $\frac{m}{n}$ has a max value of $\frac{R}{n}-\varphi$. ∴ so $e^{\frac{m}{n}\cdot m\ell}$.

0 Hmm of all the factors, .09 is most unreasonable! for very large n, $\frac{m}{n}$ will have a certain max size, but m can be very large, so $\frac{m}{n}\cdot m\ell$ can be very large!

This trouble is due to $\left(1-\frac{m}{n}\right)^{n-m\ell}$ factor. the $\left(1-\frac{m}{n}\right)^n$ part seems ok, it cancels e^m factor.

but $\left(1-\frac{m}{n}\right)^{-m\ell}$ doesn't. Take it to second order!

$\ln\left(1-\frac{m}{n}\right) \approx -\frac{m}{n} + \frac{m^2}{2n^2}$ $\quad \left(1-\frac{m}{n}\right)^{-m\ell} = e^{\left(-\frac{m}{n}+\frac{m^2}{2n^2}\right)(-m\ell)}$

$= \exp\left(\frac{m^2\ell}{n} - \frac{m^3\ell}{2n^2}\right) = \exp\left(\frac{m}{n}\cdot m\ell - \left(\frac{m}{n}\right)^2\cdot\frac{m\ell}{2}\right) = \exp\left(+\frac{m}{n} - \frac{1}{2}\left(\frac{m}{n}\right)^2\right)m\ell$

$\exp\left(\frac{m}{n}-\frac{1}{2}\left(\frac{m}{n}\right)^2\right)m\ell$

so second (or probably any) order, doesn't help:
m's available values are $\propto n$: ∴ for very large n, m can be very large.

| $\ln(1+x) \sim$
$x - \frac{x^2}{2} + \frac{x^3}{3}$

$\ln 1+.1$
$\sim .1 - \frac{.01}{2} = .095$
~~=.0955~~

$\ln 1.1 = .095310$

$.1 - \frac{.01}{2} + \frac{.001}{3}$
.005 .00033
.09533

0 Perhaps in 282.36, $\left(\frac{m}{n-m\ell}\right)^m\left(1-\frac{m}{n-m\ell}\right)^{n-m\ell}$ would be more correct! It would

21 certainly give a neat result! — No! $\left(\frac{m}{n-m\ell}\right)^m$ still gives $e^{\frac{m\ell}{n}}$ factor. → 285.00

22 $\frac{1}{\sqrt{2\pi m}} \quad \frac{R!}{(R-m)!\,(n\varphi)^m}$ This second factor is $\frac{R}{n\varphi}\cdot\frac{R-1}{n\varphi}\cdot\frac{R-2}{n\varphi}\cdots\frac{n\varphi}{n\varphi}$

max 0 1 2 ... $R-n\varphi$.

More exactly → $\frac{R}{n\varphi}$ $\frac{R-1}{(n-\ell)\varphi}$ $\frac{R-2}{(n-2\ell)\varphi}$ ← No!

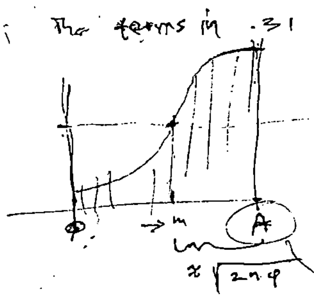
So this begins to look reasonable again!

No! it's $\frac{R}{n\varphi}\cdot\frac{R}{(n-\ell)\varphi}\cdot\frac{R-1}{(n-\ell)\varphi}$, | $\frac{R}{(n-2\ell)\varphi}\cdot\frac{R-1}{(n-2\ell)\varphi}\cdot\frac{R-2}{(n-3\ell)\varphi}$ | $\frac{R}{(n-3\ell)\varphi}\cdot\frac{R-1}{(n-3\ell)\varphi}\frac{R-2}{(n-3\ell)\varphi}\frac{R-3}{(n-3\ell)\varphi}$ |

I'm not sure it's ok., but it's beginning to look like a "Good Lead"

0

.31 $\sum_{m=0}^{A} \frac{(n\varphi)^{-m}}{(R-m)!}$ is much more well known. (i.e. no $\frac{1}{\sqrt{m}}$ factor).



The terms in .31 increase exponentially at first but then flatten out. At limit ε reached 4. So get or "half value point"; see how large m is.

Working backwards: largest term is L; next is $L\frac{n\varphi-1}{n\varphi} = L\left(1-\frac{1}{n\varphi}\right)$

.37 normal say $n\varphi = \alpha$; $L,\ L\left(1+\frac{1}{\alpha}\right)\cdot\left(1+\frac{2}{\alpha}\right)\cdots\left(1+\frac{R-\alpha}{\alpha}\right)$

$1+2\cdots r = \frac{r(r+1)}{2}$

$\left(1+\frac{r}{\alpha}\right) \approx \exp\frac{r}{\alpha} \to \exp\frac{r(r+1)}{2\alpha}$ = say $r(r+1) = 2\alpha = 2n\varphi$

$r \doteq \sqrt{2n\varphi}$

$\approx \sqrt{2n\varphi}$

$A = R - \varphi n = m_{max}$

0/29 04

4-TIM

2045 β

: Try $m=2$ in 280.22 : $= -2.22$

in 280.22 we have $\epsilon = \frac{1}{n(1-\varphi\ell)}$ for $m=1$ $\to \frac{1}{n}\left(1-\frac{1}{n}\right)^{n-\ell}\frac{1}{\varphi}R$

if $m=2$ $\epsilon_{max} = \frac{2}{n(1-\varphi\ell)}$ 280.22 gives $\left(\frac{2}{n(1-\varphi\ell)}\right)^2\left(1-\frac{2}{n(1-\varphi\ell)}\right)^{n-2\ell}\cdot\frac{R(R-1)}{2}\cdot\frac{1}{\varphi^2}$

$= \frac{2\cdot 2}{2}\cdot\frac{1}{(1-\varphi\ell)^2}\cdot e^{-2\cdot\frac{1}{1-\varphi\ell}}\left(\frac{R}{n}\frac{R-1}{n}\cdot\frac{1}{\varphi^2}\right)^{\pm 1}$ — $\approx 2\frac{1}{(1-\varphi\ell)^2}\cdot e^{2\cdot\frac{1}{1-\varphi\ell}}$

$\approx 2\left(\frac{1}{(1-\varphi\ell)\,e^{\frac{1}{1-\varphi\ell}}}\right)^2$

$z\,e^{\frac{1}{z}}$	z
2.72	1
3.69	$\frac{1}{2}$
6.70	$\frac{1}{3}$

But [range on m], no heavy factor

$\frac{m^m}{m!} \approx \frac{e^m}{\sqrt{2\pi m}}$ mult by $\left(\frac{1}{(1-\varphi\ell)e^{1-\varphi\ell}}\right)^m = \left(\frac{1}{(1-\varphi\ell)e^{1-\varphi\ell}}\right)^n \sqrt{2\pi m}$ (it is unimportant for large n)

$\frac{m^m}{m!} \approx e^m\sqrt{2\pi m}$

This is a little better than before: But now, the penalty for use of it is dependant on $\varphi\ell$ which seems strange!

T: source of 279.14: 279.34 : $\varphi(1-\epsilon)^\ell + \epsilon = \frac{R}{n}$. $\to$ $\varphi(1-\epsilon\ell)+\epsilon = \frac{R}{n}$

To second order: 279.36 $\varphi\left(1-\ell\epsilon+\frac{\ell(\ell-1)}{2}\epsilon^2\right)+\epsilon = \frac{R}{n}$

$1-\ell\epsilon+\frac{\epsilon}{\varphi}+\frac{\ell(\ell-1)}{2}\epsilon^2 = \frac{R}{n\varphi}$

$\varphi-\varphi\epsilon\ell+\epsilon = \frac{R}{n}$

$\epsilon(1-\varphi\ell) = \frac{R}{n}-\varphi$

$\frac{\ell(\ell-1)}{2}\epsilon^2 + \left(\frac{1}{\varphi}-\ell\right)\epsilon + 1-\frac{R}{n\varphi} = 0$: $\epsilon = \frac{\ell-1 \pm \sqrt{\left(\frac{\ell-1}{\varphi}\right)^2 - 4\frac{\ell(\ell-1)}{2}\cdot\left(1-\frac{R}{n\varphi}\right)}}{2\cdot\frac{\ell(\ell-1)}{2}}$

$\ell-1 \pm \sqrt{(\ell-1)^2 - 2\ell(\ell-1)\cdot\left(1-\frac{R}{n\varphi}\right)}$ over $\ell(\ell-1)$; $1-\frac{R}{n\varphi} = \delta$

$(\ell-1)^2 - 2\ell(\ell-1) - 2\ell(\ell-1)\delta$

$\ell^2-2\ell+1-2\ell^2+2\ell = -\ell^2+1 = \ell^2-1-2\ell(\ell-1)$

$x^2-2x+1=0$

$x = \frac{2\pm\sqrt{4-4}}{2} = 1$

for small $1-\varphi\ell$, (10) looks like its not far off from reasonable. Perhaps using full 2 law is better to do.

$\ell-1-\sqrt{(\ell-1)^2+2\ell(\ell-1)\cdot\delta}$ $\pm = -1+\frac{R}{n\varphi}$

$\left(\frac{1}{\varphi}-\ell\right)\epsilon + 1-\frac{R}{n\varphi} = 0$

$\epsilon = \frac{\frac{R}{n\varphi}-1}{\frac{1}{\varphi}-\ell}$

$= \frac{\frac{R}{n}-\varphi}{1-\ell\varphi}$

$\left(1-\ell\epsilon+\frac{\ell(\ell-1)}{2}\epsilon^2\right)+\frac{\epsilon}{\varphi} = \frac{1}{n\varphi}$

$\frac{\ell(\ell-1)}{2}\epsilon^2 + \epsilon\left(\frac{1}{\varphi}-\ell\right) + \left(1-\frac{R}{n\varphi}\right)$: $\epsilon = \frac{\ell-\frac{1}{\varphi} \pm \sqrt{\left(\ell-\frac{1}{\varphi}\right)^2 + 4\frac{\ell(\ell-1)}{2}\delta}}{\ell(\ell-1)}$ $\quad -\delta$ if $\delta = \frac{R}{n\varphi}-1 > 0$

$\sqrt{\left(\ell-\frac{1}{\varphi}\right)^2 + 2\ell(\ell-1)\delta} \approx \left(\ell-\frac{1}{\varphi}\right)\sqrt{1+\frac{2\ell(\ell-1)\delta}{\left(\ell-\frac{1}{\varphi}\right)^2}}$ Taylor $\sqrt{\ } = 1+\frac{\ell(\ell-1)\delta}{\left(\ell-\frac{1}{\varphi}\right)^2}$

$-\frac{b\pm\sqrt{b^2-4ac}}{2a}$

$\ell-\frac{1}{\varphi} \pm\left(\ell-\frac{1}{\varphi}+\frac{\ell(\ell-1)\delta}{\ell-\frac{1}{\varphi}}\right)/\ell(\ell-1)$ for "$-$" $\to -\frac{\ell(\ell-\frac{1}{\varphi})+\ell(\ell-1)\delta}{(\ell-\frac{1}{\varphi})\ell(\ell-1)} = -\frac{\delta}{\ell-\frac{1}{\varphi}} = \frac{R}{\varphi n}-1\Big/\ell-\frac{1}{\varphi}$

$= \frac{\frac{R}{n}-\varphi}{-\ell\varphi+1}$

There is another small factor! $(1-\epsilon_m)^{-m\ell} \approx e^{+\epsilon m\ell}$ in (10)

We get $\left(\frac{}{(1-\varphi\ell)\quad e^{\frac{1}{1-\varphi\ell}}-1-\epsilon m\ell}\right) = \frac{e^{1+\frac{m^2\ell}{n}}}{(1-\varphi\ell)\,e^{\frac{1}{1-\varphi\ell}}}$

ϵ_m as a function of m ?...

$\epsilon_m \approx \frac{m}{n}$

$\left(1-\frac{m}{n}\right)^{-m\ell} \approx e^{+\frac{m^2\ell}{n}}$

$\frac{m^2\ell}{n} = \epsilon m\ell$ which can be very large if m is large!

Enable — too large! Back to 280.22: if $\epsilon_m = \frac{m}{n}$ [this] $\boxed{\epsilon_m = \frac{m}{n-m\ell}}$ is correct.

$R-\varphi n$ — maybe also larger.

$\sum_{m=0}^{}\left[\left(\frac{m}{n}\right)^m\left(1-\frac{m}{n}\right)^{n-m\ell}\cdot\varphi^{-m}\frac{R!}{m!(R-m)!}\right]$ — for $m=0$ term is $\frac{1}{\frac{1}{n}\left(1-\frac{1}{n}\right)^{n-\ell}}$ — for $m=1$ term is $\varphi^{-1}\cdot R = \frac{R}{n}\frac{e^{\frac{R}{n}}}{e^{\varphi}}$

$e^{-1+\frac{R}{n}}$ — if $\frac{R}{n} = \frac{1}{e}$ $\approx$ cancel

$\frac{e^{\frac{1}{e}}}{e} \neq \frac{1}{e}$!!

Incorrect $\frac{1}{e}$ because $\frac{R}{n} > \frac{1}{e}$.

See 284.06 for approx. summation.

0/28/04

4 PM

IAP | Scheduler office 253 4788 — Post :::::: IAP url http://mit.edu/iap/ organize : non-credit. post and revise. [illegible]

0 : Actually, an easy way to check formula −1.22 or −1.25, would be numerically in Basic or Maple or Mathematica. I can start w first order Approx. of ϵ_m (−1.22R) then (279.34R) for second order. The params are R, φ, n, ℓ. To start, I want pc as a function of $\frac{R}{n}$ & φ (say n fixed).

For factorial, use my hi precision formula.

In Basic; I have forgotten how to do subroutines [for X! say].

formula −1.22 doesn't look bad!

I want to compare it w pc before coding: see −1.07 ; so −1.22 gives 1 in pc due to definition. It should be ? if $\frac{R}{n} = \varphi$; but we may get $\div \varphi$ in Restore (?).

.08 try it for $\frac{R}{n}$ very slightly $> \varphi$. Anyway, this gives $\epsilon \simeq 0$ — No!

.09 we sum m from R to R. one term $\varphi^{-R} = \left(\frac{R}{n}\right)^{-R} \cdot \frac{R!}{R!} = \left(\frac{R}{n}\right)^{-\ell} \cdot \left(\frac{n}{\ell}\right)^{R}$

0 So we have only the φ^{-R} factor: −1.07 is supposed to explain it.

1 : (−1.07)! The φ^{-m} factor does not occur; we [illegible] still

No! — (.08) is wrong, $\epsilon \simeq \varphi$. I have really gotten ϵ's values confused!

I really have to go over formula −1.22) and −1.07 !!

Re: −1.07 φ^{-} At certain points I may be confusing m and

Possi. confusion : I would want m to go from φ to $R - \varphi n$ (I was confused betw. $\simeq R$ and $\frac{\ell}{n}$ m)

Since $\epsilon \simeq \dfrac{\frac{R}{n} - \varphi}{1 - \varphi\ell}$, $\epsilon \simeq 0$ if $\frac{R}{n}$ is only marginally $> \varphi$.

Exactly what is m ?? I think m is the no of times δ appears in the code for the corpus. (ϵ can be $> \varphi$, here — m is just one code for the corpus)

So m is ϵn at most & φ at minimum. If m = 0 then pc of code = $(1-\epsilon)^n$

0 would it be adding only Correction for limits on m in (−1.22) to be 0 & $R - n\varphi$?

Viewed in this way −1.22 begins to look reasonable! $e^m (1-\epsilon)^{n-m\ell}$ for that = 1 for m = 0

i.e. $\varphi^\varphi = 1$ & $1^n = 1$, but It would be more realistic to have m = 1 & n large.

O.K., say m = 1; n >> 1 ; $R = \varphi n + 1$ $\frac{R}{n} = \varphi + \frac{1}{n}$. $\frac{R}{n} - \varphi = \frac{1}{n}$;

so for −1.22 $\left(\frac{1}{n(1-\varphi\ell)}\right)^1 \left(1 - \frac{1}{n(1-\varphi\ell)}\right)^{n-\ell} = \frac{1}{n(1-\varphi\ell)} e^{-\frac{n-\ell}{n(1-\varphi\ell)}} \simeq \frac{\exp\left(\frac{-1}{1-\varphi\ell}\right)}{n(1-\varphi\ell)}$

$\epsilon_{max} = \dfrac{\frac{R}{n} - \varphi}{1 - \varphi\ell} = \dfrac{1}{n(1-\varphi\ell)}$

$\epsilon_{max} = \dfrac{1}{n - n\varphi\ell} = \dfrac{1}{n(1-\varphi\ell)}$ approximations

$1 - \varphi\ell$ is > 0 because $\frac{1}{\varphi} - \ell > 0$ because $\frac{1}{\varphi} > \ell$ because $\frac{1}{\varphi}$ is the expected dist. betw. occurrences of δ: this must be $\geq \ell$.

In −1.22 we have $\frac{R!}{m!(R-m)!} = \frac{R!}{1!(R-1)!} = R$: If we summed from 0 to 1 ;

$\frac{R!}{0!(R-0)!} = 1$ $\left(\frac{1}{n(1-\varphi\ell)}\right)^0 \cdot \left(1 - \frac{1}{n(1-\varphi\ell)}\right)^n \simeq e^{-\frac{1}{1-\varphi\ell}}$

$\frac{R}{n} - \varphi = \frac{1}{n}$

$R - \varphi n = 1 = m$

if $R - \varphi n = 2$

$\frac{R}{n} - \varphi = \frac{2}{n}$

~~So it looks like −1.22 goes $(R+1) e^{-\frac{1}{1-\varphi\ell}}$~~

for m = 1 ; the term is $\frac{R}{n(1-\varphi\ell)\varphi} e^{-\frac{1}{1-\varphi\ell}} \simeq \frac{R}{\varphi n(1-\varphi\ell)} e^{-\frac{1}{1-\varphi\ell}} =$

for m = 0 : .. $e^{-\frac{1}{1-\varphi\ell}}$ } so this is much lower $\simeq e^{-1+\varphi\ell} \simeq e^{-1} \cdot e^{\varphi\ell} \simeq \frac{1}{e}(1+\varphi\ell)$.

Can m go from 0 to $n\left(\frac{R}{n} - \varphi\right) = R - n\varphi$? (.80)

we have R possible positions at $pc = \frac{1}{n(1-\varphi\ell)}$ so $\frac{R}{n(1-\varphi\ell)}$ we mult by $\frac{1}{\varphi}$ to get $\simeq \frac{1}{1-\varphi\ell}$

There is still 1 times $(1-\epsilon)^{n-\ell} = \left(1 - \frac{1}{n}\right)^{n-\ell} \sim \frac{1}{e}$. So I still seem to have this [illegible] $\frac{1}{1-\varphi\ell} \cdot \frac{1}{e}$ penalty for coding (1 occurrence of δ).

7/4/04 T.... (Urbana) Jacobson 617 253 0482

4TM

280

10: : Actually, t. pc of t. codes using t. γ defn. are much higher due to all coding: Of all of t. R occurrences of t. [illegible] ngm γ; any no. (between) combo coded by t. defn., & this combo done in many ways, so we get all these ll codes — many of different PC.

Here t. definition of γ has pc ϵ & will ↓ pc of each corpus by $\approx (1-\epsilon)^n$ (actually t. exponent is t. no. of symbols (other than t. one for γ.) in t. coded of t. corpus. $\approx e^{-\epsilon n}$

Say of the R occurrences of γ, we code m of them as γ defn. & $(R-m)$ of them as alphabet seqs.

07 T. pc of coding is $\epsilon^m \cdot (1-\epsilon)^{n-m\ell} \cdot \varphi^{-m}$ (original pc of corpus.) ← no. of symbols in coded code. Amount saved by not coding as alphabet symbols. No! — See +1.11 — where $m = \varphi n$ then $\epsilon_m = 0$.

for a given m there are $\frac{R!}{m!(R-m)!}$ ways to do this. ← ϵ is different for each m.

09 We want to sum $\epsilon^m (1-\epsilon)^{n-m\ell} \varphi^{-m}$. from 1.33: $\epsilon_m = \frac{\frac{m}{n} - \varphi}{1-\varphi\ell}$ — which suggests we sum m from φn to R. .20 Both big & low freq.

279.14 : EN Re: −1.10 To make this less. A.H., apply it to all occurrences of γ: i.e. If γ has been defined, whenever γ is created by alphabet" it is destroyed. T. only way a γ can be created is by its choice as a "symbol" (≡ "token").

An objection to .10!: Say γ is ℓ symbols long. I'm looking at a corpus in which the last $\ell-2$ symbols of γ have just occurred. There are 2 ways that t. next symbol can occur: ① By alphabet ② by γ selection so pc of that symbol is "very hi"; However, if we see $\ell-1$ symbols of γ in t. corpus, Then except for t. next symbol, t. one that completes γ, it can only occur via γ selection — so there is this discontinuity at just t. point at which γ is very relevant!

---

18 Perhaps drop t. −1.10 & (280.40) for awhile: The "unusually low freq. of an agent" Doesn't occur — what is that?

19 Very often. Until I can think of a possl cause of it in RW, it seems not-so-necessary!

20: .09 We have redundancy factor ~~$\frac{n!}{m!...}$~~ $\frac{R!}{m!(R-m)!}$ (pretty weird.)

22 So we want. $\sum_{m=\varphi n}^{R} \left(\epsilon_m^m (1-\epsilon_m)^{n-m\ell} \varphi^{-m} \frac{R!}{m!(R-m)!}\right)$; $\epsilon_m = \frac{\frac{m}{n} - \varphi}{1-\varphi\ell} = am - b$

+1.20 STRONGLY suggests limits should be $0 \to R-\varphi n$

25 $\sum_{m=\varphi n}^{R} (am-b)^m (1+b-am)^{n-m\ell} \varphi^{-m} \frac{R!}{m!(R-m)!}$ (Is R! a constant?)

$(x+y)^m = x^m + m y x^{m-1} + \frac{m(m-1)}{2!} y^2 x^{m-2}$

$(x+y)^3 = 1\ 3\ 3\ 1$

T. argt. of .25 is perhaps a monomodal peak function of m. These 2 factors are sort of like $(\alpha+m)^m (\beta-m)^{n-m\ell}$ & Best approximate this as Gaussn. (am+b) goes from zero to some fnct of R.

~~perhaps are coutt~~

Read (R.2) as Gaussn., then $\varphi^{-m} = e^{(\ln\varphi)m}$ so t. product is a Gaussn.

To get params of Gaussn. fcn: find peak & second derivative.

$(1+b-am)^{n-m\ell}$ may be expressible as $\approx e^{\text{something linear in } m}$ i.e. $\exp(-(am-b)(n-m\ell))$ ← so an "inverse" Gaussn.: e^{+m^2} factor.

$(am-b)^m = a^m m^m - mb\, a^{m-1} m^{m-1} + a^{m-2} m^{m-2} \frac{m(m-1)}{2} b^2$ ~~(...)~~

$a^m m^m - b\, a^{m-1} m^m$ compare a to b. $\frac{b}{a} = \varphi n \gg 1$. So second term is >> first.

Looks like t. ith term is $\propto \frac{(\varphi n)^i}{i!}$ which reaches its peak about $\varphi n = i$; easy to find width of peak of $\frac{\varphi n}{i!}$. It's not quite so simple! I have m^{m-i} and $\frac{m!}{i!(m-i)!}$

.0: 2 28A0 : So: when one defines a new ngm; It may be that the resultant grammar never gives same pc's as previous grammar! even if the new defined had same ~~pc~~ frequency as was expected from its symbols!

Remarks ① 275.15–16 still seems true ② Definitions are VERY important in sci discovery! BALDWIN effect

04 ③ –1.36–37 — Penalty of factor of ∈ for each γ occurrence → of interest (

Perhaps one can define Contexts (for produ) w. impurity.

Baldwin effect, Lamarck, Feed back from observation of ε channel of Fitness function

96 Some other ideas: ① γ can occur 2 ways: by defined symbol, or by creation by other symbols. Try code in which φ = twice expected from other symbols. Frequency of all symbols will change: See how coding cost of corpus varies as I change pc of γ & of other symbols.

0 ② For pc of ngm lower than expected & have operator that looks for γ, & destroys it w. pc = .9. Since this can be done in many ways, to produce observed corpus, we have ll codes & ↑ of pc! Since γ is less likely, the result of possible outputs of the grammar will have to ↑ somehow, to preserve "Normality".

→ The way this is done: any symbol in the corpus can be created (parsed) 2 ways: } so this leads to ↑ pc's of symbols. 280.10 [SOU 281.20]

14 ① By direct creation of β ② first create γ, & destroy γ (w. pc .9) then create β.

.06–.14 Seem like reasonable solns to the ∈ problem of –1.36–37

If we have no constraint on symbols other than their frequencies, it is clear that the code at –1.26–.35 gives us 2 ways to create γ & the total pc of it is much greater than (about twice) what was expected; On the other hand, the pc of the code seems ∈ times what it was before.

The result of this: If $\frac{R}{n} = \varphi$, then all instances of γ are accounted for as generated from ~~correct~~ alphabet. If $\frac{R}{n}$ is only slightly > φ, but corpus is long (n >> 0) then it "will pay" to define γ, but only a small fraction will be generated by the defn. γ — & we don't know which. If $\frac{R}{n} >> \varphi$ say 10 times as large than most γ's will be generated by the defn, but we don't know which; furthermore, there will be many ll codes because there will be many combinations of ways to create γ ... by defn, & by original alphabet symbols.

If $\frac{R}{n} = \varphi + \mu$ then defining γ so it has pc of exactly μ, will not work because the defn ↓ pc of other symbols so pc of creation of γ by alphabet φ.

If we define γ to have pc = ∈, then φ → φ(1 – ∈)^l (since all other alphabet symbols have their pc's ↓ by factor (1 – ∈) & γ is l symbols long.

0 So we want ∈ ∋ $\boxed{\varphi(1-\epsilon)^l + \epsilon = \frac{R}{n}}$ ... ≈ $\varphi e^{-\epsilon l} + \epsilon = \frac{R}{n}$.

for small ∈; ≈ $\varphi(1-\epsilon l)+\epsilon = \frac{R}{n}$ $\qquad \varphi - \epsilon\varphi l + \epsilon = \frac{R}{n}$ | $\varphi + (1-\varphi l)\epsilon = \frac{R}{n}$

.2 $(1-\varphi l)\epsilon = \left(\frac{R}{n} - \varphi\right)$ ← This is the amt. we have to ↑ the pc of γ to get to $\frac{R}{n}$

33 $\epsilon = \dfrac{\frac{R}{n} - \varphi}{1 - \varphi l}$ usually φl will be << 1. so ∈ will ≈ $\frac{R}{n} - \varphi$, which is reasonable.

34 More exactly, $\varphi(1-\epsilon)^l + \epsilon = \frac{R}{n}$ (pc of creation by alphabet; pc of creation by defined γ)

to second order $(1+\epsilon)^l = 1 + l\epsilon + \frac{l(l-1)}{2}\epsilon^2 \ldots$

so $\varphi\left(1 + l\epsilon + \frac{l(l-1)}{2}\epsilon^2\right) + \epsilon = \frac{R}{n}$ —

which can be solved for ∈ (a simple quadratic eq.)

This can easily be rapidly solved by successive approxn. $\epsilon_i = \frac{R}{n} - \varphi(1-\epsilon_{i-1})^l$ or if it doesn't converge do ≠

by $(1-\epsilon_i) = \left(\left(\frac{R}{n} - \epsilon_{i-1}\right)/\varphi\right)^{\frac{1}{l}}$.

01/26/04

4 TM

00 : T. result of -1.38 -.39 is simply obtained & very reasonable. T. previous stuff w. the factors of e & e.e have various diffys in them & may be wrong.

-02 Also this result & the idea of 275.15-.18 should make it clearer how to use unusually hy or low freqs of ngms to & code length.

T. way it works: If an ngm has "hy enuf" low enuf freq. & the savings $ it cost will pay for its definition. I I suspect low freq. ngms will not often pay for their defns, unless the corpus is very long!]. The way we code using the definitions:

<Say for ngms w. the statistically high freqs:> We make a definition of the ngm $= \gamma$. If γ had the "normal" freq. (by considering its components) coding using γ would not change the cost of corpus. The other ($= \gamma$) symbols will have their freqs changed, but in such a way to preserve their ratios; the all $= \gamma$ symbols will have their freqs lowered, but since we now have a new symbol, γ, it too has a certain freq.

T. forgoing analysis holds whether γ has unexpectedly hy or low frequency.

My guess is that we will not often make defns of ngms w. very low freq. because they will usually have only a small $ in bit cost — not enuf to pay for defining them.

T. formal eg. Thots were for coding using probabilities: If we code using (Dir's / Laps) rules (Dirichlet's / Laplace's) we should get about same results, there may be "e" factors, but they should cancel out when we compare different methods of coding (i.e. with v.s. w.o. use a defn).

I should calculate the cost of corpus in which I define a γ that has same pc as product of pc's of its components. To get the cost, w.o. "cost of definition"; just add a symbol, γ for purposes.

Do analysis 2 ways: 1) simply using probs that are to same for that symbol is used, .26 $(1-\varphi)^{\frac{1}{\varphi}-1} = (1-\varphi)^{\frac{1-\varphi}{\varphi}}$

2) using Dir Lap coding (Z-141)

I think T. formal stuff (277.20 — 278.22) (278.00-.22 in particular) may be a very impt. break thru in the Z-141 problem! It seems to solve 2 things ① the "factor of e paradox" ② How to define very ngms of unusually low frequency.

No! This is not a paradox, but Least factor! — see 292.18 and 278.2 — 278.0 for how $\frac{1}{e}$ arises

.24 For .21, use idea of -1.38 -.39 of coding a "typical" section of T. code, containing only γ. Say we have a corpus section containing "a" ordinary symbols in an alphabet of d distinct symbols & one γ of length L; so old corpus was of length $a+L$. Because -1.38 ≈ -.40; $a \approx \frac{1}{\varphi} - 1$; φ being the pc of γ.

Consider case in which φ is the normal pc assigned to & considered as symbols. In the $a+1$ symbol corpus, one symbol has $pc = \varphi(?)$. The other symbols have their pc's ↓ by factor $(1-\varphi)$ so, before a new symbol, the first a symbols had pc A, the last L had pc φ.

Total pc of corpus = $A \cdot \varphi$. With new code, the first a symbols have pc = $A(1-\varphi)^2$ and the last symbol has pc = φ. So total pc = $A(1-\varphi)^2 \cdot \varphi$. ANOTHER LOSS!! of $\frac{1}{e}$ $(1-\varphi)^2 = 1 - 2\varphi + \varphi^2 = (1-\varphi)^{\frac{1}{\varphi}-1} = (1-\varphi)^{\frac{1-\varphi}{\varphi}}$

AGAIN This looks like old Z-141 paradox: we get a penalty of about a factor of e for each occurrence of γ, where we define γ (!!). $\frac{2}{e} \approx e^{-1+\varphi}$

There may be a bit more constraint on the first a symbols, since we know that they can't end with "all but" γ - w. no consequent The resultant Bernoulli grammar w γ does not produce the same corpus that the original Bernoulli grammar did. γ occurs w. greater freq. than before. Is this the death of γ? definition

9/26/06
4 PM

20 : $m - (\cdot 01) \,\dot{\imath}\, .08$ the $(1-\varphi)^m = (1-\varphi)^{n-\ell R}$ has to be under the Σ sign.

$$\sum_{R=0}^{\infty} R \frac{(\varphi m)^R}{R!} (1-\varphi)^{n-\ell R} \quad [m \equiv n - \ell R] \qquad (e^{-\ell})^R$$

$$\approx \varphi m \sum_{R=1}^{\infty} \frac{(\varphi m)^{R-1}}{(R-1)!} e^{-m\varphi} = \varphi m \sum_{R=1} \frac{(\varphi m)^{R-1}}{(R-1)!} \cdot e^{-m-\ell R}$$

m contains R! so this is an approxn. — d [illegible]

$$= \varphi m \cdot e^{-n} \sum_{R=1}^{\infty}$$

$\sum_{R=0}^{\infty} R$ woops! $\sum \frac{\varphi m^R}{R!} \neq e^{\varphi m}$, because m is a function of R.

2 $(\varphi m)^R = \left(n\varphi \frac{(n-\ell R)}{n}\right)^R = (n\varphi)^R \left(1 - \frac{\ell R}{n}\right)^R \approx (n\varphi)^R e^{-\frac{\ell R^2}{n}}$ | $R_{max} = \frac{n}{\ell}$; $e^{-\frac{\ell n^2}{n \ell^2}} = \frac{n}{\ell}$

$e^{-\frac{\ell R^2}{n}} = e^{-1}$ when $\frac{\ell}{n} R^2 = 1$: $R^2 = \frac{n}{\ell}$: $R = \sqrt{\frac{n}{\ell}}$.

for $n \gg \ell$ (i.e. a long corpus), R will be large before this occurs.

$\frac{(\varphi m)^R}{R!}$ will perhaps be quite small; say $\varphi m \approx R$ $\frac{R^R}{R!} \approx \frac{R^R e^R}{R^R} \sqrt{2\pi R}$.

so not so small! $\approx e^R \cdot \sqrt{2\pi R}$. This is mult by $e^{-m\varphi}$ ($m\varphi \approx R$) to give $\sqrt{2\pi R}$ ....

which is not so huge compared to E[illegible] within [illegible] of [illegible],

The large stuff is very "floppy" : I should check this stuff numerically, Ideally, try a Monte Carlo

3 run to see how frequently an [illegible] approx!.

Look at it this way: we go along with sequence incrementing i. We have $pc = \varphi$ of emitting γ.

If we do emit γ, the next ℓ cannot emit γ, but then all we can emit γ again.

Let's look at d.f. of distances betw. γ's! Prob of ϕ (zero) is φ; prob of 1 is $(1-\varphi)\varphi$;

prob of n is $(1-\varphi)^n \varphi$. Expected distance is φ [illegible] $(\phi \cdot 1 + 1 \cdot (1-\varphi) + 2(1-\varphi)^2$

say $1-\varphi \equiv \alpha$ (so $\varphi \sum i\alpha^i = \varphi\alpha \frac{d}{d\alpha} \sum \alpha^i$ | $\equiv \sum i \alpha^{i-1} = \frac{\alpha}{(1-\alpha)^2}$

$\sum_0^{\infty} \alpha^i = \frac{1}{1-\alpha}$ $\quad d \frac{1}{1-\alpha} = \frac{-1}{(1-\alpha)^2} \cdot -\alpha = \frac{\alpha}{(1-\alpha)^2}$

$\frac{d}{} \frac{1}{1-\alpha} = -\frac{1}{(1-\alpha)^2} \cdot -\alpha = \frac{\alpha}{(1-\alpha)^2}$.

expected distance $= \varphi \frac{\alpha}{(1-\alpha)^2} = \varphi \frac{1-\varphi}{\varphi^2} = \frac{(1-\varphi)}{\varphi} \cdots = \frac{1}{\varphi} - 1$ ← I made error first time but this is OK.

Expected distance $= \varphi \cdot \phi + \varphi\alpha \cdot 1 + \varphi\alpha^2 \cdot 2 + \varphi\alpha^3 3 = \varphi(\alpha \cdot 1 + \alpha^2 2 + \alpha^3 3 \ldots)$

$1 + \alpha + \alpha^2 + \alpha^3 = \frac{1}{1-\alpha}$ $\quad \frac{d}{} \frac{1}{1-\alpha} = \frac{\alpha}{(1-\alpha)^2} = 1 + 2\alpha + 3\alpha^2 + 4\alpha^3$

$\alpha \cdot d \frac{1}{1-\alpha} = \frac{\alpha^2}{(1-\alpha)^2} = \alpha + 2\alpha^2 + 3\alpha^3 \ldots$

$\varphi(\alpha + \alpha^2 2 + \alpha^3 3 \cdots) = \frac{\alpha^2 \varphi}{(1-\alpha)^2} = \frac{(1-\varphi)}{\varphi} = \frac{1}{\varphi} - 1$. = expected no of symbols betw γ's.

So we have a "corpus" of length $\frac{1}{\varphi} - 1$ symbols followed by ℓ symbols. — which has 1 α in it.

~~[illegible] relative probabilities~~ so in a corpus of length n, we expect γ to occur

8 $\frac{n}{\frac{1}{\varphi} - 1 + \ell}$ times. This is $\frac{n\varphi}{1 - \varphi + \ell\varphi} = \boxed{\frac{n\varphi}{1 + (\ell-1)\varphi}}$

7 if $(\ell-1)\varphi$ is $\ll 1$ (which is usual); this is $\boxed{n\varphi}$.

0:225,40: $x \frac{d}{dx} \overbrace{\sum_{k=0}^{\infty} \frac{x^k}{k!}}^{e^x} = \sum_{k=0}^{\infty} \frac{k \cdot x^{k-1+1}}{k!} = x e^x$

01 so $\sum_{k=0}^{\infty} k \frac{(\varphi m)^k}{k!} = \varphi m\, e^{\varphi m}$ which is ~~more~~ unmeasurable from $e^{\varphi m}$!

My calculations of pc of just k occurrences may be wrong! (perhaps) minor point. $\sum_{k=0}^{m=\infty}$ + largest term is $m \frac{(\varphi m)^m}{m!} = \varphi^m \cdot \frac{m\, m^m e^m}{m^m \sqrt{2\pi m}}$

$(\varphi e)^m$ is usually quite small. mult by $\sqrt{\frac{m}{2\pi}}$ still very small. So the approx of .01 is safe! $= \varphi^m \cdot e^m \cdot \sqrt{\frac{m}{2\pi}}$

Have I really gotten rid of the duplication of cases where we have > 1 δ at same point"?

8 would be $\frac{m^k}{k!} \cdot \varphi^k (1-\varphi)^{m-\ell k}$ (ℓ?) $(1-\varphi)^m \simeq e^{-\varphi m}$ when we use cases the mult factor in .01 we get φm. which is quite reasonably! → No! – see 227.01 — Maybe $n - k\ell$

09 – as expected.

The distribution over probs of various k values is $\frac{(\varphi m)^k \cdot (1-\varphi)^{n-\ell k}}{k!} = \frac{(\varphi m)^k \cdot e^{-n+\ell k}}{k^k e^k \sqrt{2\pi k}}$

$= e^{-n} \sqrt{2\pi} \left(\frac{\varphi m\, e^{(\ell-1)}}{k}\right)^k \cdot \sqrt{k}$ $\Big|\Big|$ $\left(\frac{z}{k}\right)^k \cdot \sqrt{k}$ say $z = 10$.

$2^5 \cdot \sqrt{5} = 32\sqrt{5}$

· at $k=10$, $f = \sqrt{k} = \sqrt{10}$: as $k\uparrow$, $f\downarrow$; say $k = \frac{z}{2} = 5$

$k = 1$ $z=10$ $10^1 \cdot \sqrt{1} = 10$ so peak between $k=1$ & $k=10$ (for $z=10$)

$k=1$: $F = z$ — so probably a peak for k between 1 & z.
$k=2$: $F = \left(\frac{z}{2}\right)^2 \cdot \sqrt{2} = z^2 \cdot \frac{\sqrt{2}}{4}$ | looking at $\left(\frac{z}{k}\right)^k$ only take 2 root: $\left(\frac{z}{k}\right)^{\frac{k}{z}}$ $\frac{k}{z} = \alpha$
$k=3$: $F = \frac{z^3}{3^3} \cdot \sqrt{3}$

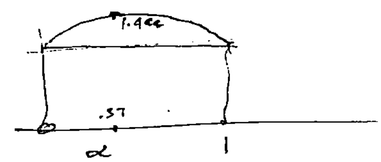
So peak is $\alpha \sim \underline{.37} = \alpha^{-\alpha}$

for fixed k: .55 1.3.

$\alpha^{-\alpha}$: .1 $= \left(\frac{1}{\alpha}\right)^\alpha$ .65 → 1.32 — A broad peak. α | 1.444 at $\alpha = .37$ | from 1 at $\alpha = 0$ & 1

$\alpha^{-\alpha}$	α
1.047	.01
1.25	.1
1.38	.2
1.44	.4
1.432	.45
1.414	.5
1.36	.6
1.28	.7
1.20	.8
1.10	.9
1	1

.45 → 1.432
.3 1.435
.35 1.44029
.37 1.444659
.4 1.442
.38 → 1.44438
.45 1.432



Taking the $z^{\frac{k}{z}}$ power of f will accentuate the peak. 1.444 peak value.
If z is as much as 100, the $\sqrt{k}$ is of no import — the peak will be at $\alpha = .37$ & its width depends on z

x^{-x} — $-x \ln x$ $\frac{-x}{x} - \ln x = 0$ $-1 = \ln x$ $x = \frac{1}{e} = .36787\ldots$

$e^{\frac{1}{e}} = 1.444667$ = peak, get second deriv. to get width.

$x^{-x} = e^{-x\ln x}$ (d) $= (-1-\ln x)\, e^{-x\ln x}$ deriv $= -\frac{1}{x} e^{-x\ln x} + (1-\ln x)^2 e^{-x\ln x}$

so sec deriv $= \left((1-\ln x)^2 - \frac{1}{x}\right) x^{-x}$ $x = e^{-1}$ $\left((1+1)^2 - e\right) e^{-e}$ $\boxed{e^{-e}} \cdot 1.444667 = 1.851657049$
$4 - e$ = second deriv at peak of x^{-x} peak $= x = 1.444667$

Hm, note that peak occurs at $\frac{z}{k} = e$ $\frac{z}{k} = \frac{\varphi m e^{\ell+1}}{k} = e$

or when $\frac{\varphi m\, e^{\ell}}{k} = 1$. $\frac{\varphi m\, e \cdot \ell}{k} = 1$ $\frac{k}{m} = \varphi (e \cdot \ell)$ .1

$\left(\frac{z}{k}\right)^{\frac{k}{z}} \to \left(\frac{k}{z}\right)^{-\frac{k}{z}}$ peak at $\frac{k}{z} = \frac{1}{e} = .36787$

even at "ℓ = 1" (.05 & .09k) we still have that $\frac{z}{e}$ spurious factors

0/25/04
4 PM

The Z141 Paradox: 10. A "Final Solution" seems to be 277.20–.40; Re 278.00 r.22 → Not such a great soln
This also solves problem of defining ngms w. unusually low frequency. → This second idea is not so hot

∝ 10
See 285.31–.32!
See 280.10–.19
See 282.10–.13 for how $\frac{1}{R}$ legitimately arises.

0:27.40 : One characteristic of the S-Parse of −1.34–.40 : While it is "relatively easy" ② to discover Grammars of this type from a corpus, they don't easily assign (assign?) a pc to an arby object. — For s Grammar models, the object has to be parsed! (The Rust and Rust of Matrix parsing would work o.k. for s grammars. It takes n^3 operations to mult 2 $n \times n$ (stochastic) matrices. (only n^2 for d-matrices for $n < 64$, say). These models can give Monte Carlo d.f.'s easily, however. The problem is in making a prediction: i.e. a continuation of a string ("completion").

N.14.04 The n^3 is deceptively large! Most of elements of matrices of interest are zero — particularly at the start of the computation. All non-zero mults & additions are necy & important in the final result. We may use d-Matrices to keep track of where non-zero elements are in the S-Matrices, so as to speed up calcn. The cost of keeping the d-matrices will be << that of multiplying the S-Matrices. So this parsing of S-Grammars may not be as bad as it initially sounds (in a cc sense).

): On the Z141 paradox: say φ is the pc of ngm γ obtained via pc's of its symbols:
say γ occurs R times in corpus of n symbols. — so $\frac{R}{n}$ is its observed frequency.
φ is the pc that γ would occur at an arby pt, 2, in the corpus. To estimate freq. of occurrence of γ in corpus in which the only constraints were freqs of subsymbols of γ: — Also some trouble w. overlap.
If γ is of length l, there are only $\sim n - Rl$ opportunitys for γ to occur (?)).
5 At any rate, if R differs much from expected no of occurrences of γ then we should have
6 a way to use this fact for prediction — i.e. a way to shorten codes. This argument is
7 true if R is too small or too large! I imagine its usually easier to find discrepity for
8 overly large R, because "small R" can only be no < 0; but large R can be many times > expected.

First consider case of γ being "non (self) overlappable" (the sequence "aaa" is (self) overlappable, the sequence abac is not.) [I think I did work this out in detail]

0 A way to do it: To get pc of just R cases of γ in corpus of n symbols: Consider the $n - Rl$
1 symbols that are not part of γ's: Any of these symbols can be followed by a γ [we consider these inserted γ's to
25 be of zero length, so several of them can fit into the space following any one of the $n - Rl$ symbols] (.29)

Anyway, for very large n, it would seem that γ would occur w. frequency $\boxed{R = n\varphi}$. & an'b't
R deviates from $n\varphi$ much (i.e. $\frac{R}{n}$ deviates from φ much) then we probly have a regularity.
In my Z141 analysis, $\frac{R}{n\varphi}$ → for "large" $\frac{R}{n}$ had to deviate from φ far too ... so this was paradox.

← φ is product of pc's of the symbols in γ

Also note .15–.16 gives idea as to how to use excessively low R for coding

24 (.25) Re: (.22–.25) to avoid the many possl orderings of γ's "at the same point": say $\boxed{m = n - Rl = \text{no of insertion pts}}$ for γ. My first approach was m^R different ways w. redundance at each pt. where we had ≥ 1 γ.
) A better — which is hard to compute; another way is do $m(m-1)\cdots(m-(R-1))$ different ways to insert γ — we only allow one γ at a point, — but I'm not sure of details.
On second thot m^R may be correct — if we insert a new γ just after an already inserted γ, or between a two symbols (not adjacent to an inserted γ) so $\frac{m^R}{R!}$ ($R!$ different orderings of insertions)

$= \frac{(n-Rl)^R}{R!}$ ways. If the pc of any one way is φ then pc of R occurrences is $\frac{(\varphi)(n-Rl)^R}{R!} = \frac{(\varphi m)^R}{R!}$

$\frac{(e\varphi m)^R}{R^R}\cdot\sqrt{2\pi R}$ | Expected no of occurrences: $\sum_{R=0}^{\infty}\frac{(\varphi m)^R}{R!} = e^{\varphi m}$ exactly which seems crazy!

φm would seem like a more reasonable expected value of R

woops expected value $\sum_{R=0}^{\infty} R\,\frac{(\varphi m)^R}{R!} = \sum_{R=0}^{\infty} R\,\frac{x^R}{R!}$ | $x\frac{d}{dx}\sum_{R=0}^{\infty}\frac{x^R}{R!} = \sum_{R=1}^{\infty} R\,\frac{x^{R-1+1}}{R!}$ → 276.00

6/23/04 S-Functs.25

4TM

[SN see 4TM 26 (41) for W. Paul's Grammar for ANL.]

0:273.40 t. QA's as possl. — i as "shortas possl." Th. Gore at this point is a bit vague, but perhaps acceptable. (Note 272.37). I'm thinking of t. BU method of soln. {I don't know how Lsrch would work here.} After TM has done a lot of this, we get it to do prediction occasionally ... but TM's main job is usually just finding short codes for things — finding a kind of "explanations" for a growing corpus. From these codes, TM gets lots of useful concs that can be used in prediction in Math problems. TM mite be able to do Lsrch well, after this kind of training.

.27 Re: Lsrch: My impression is that Lsrch w. a suitable TSQ is a good, reasonable way to do a TSQ: T. main problem may be t. representation of S-functions.

So, review recent work on S-functs: (essentially t. material up to 272.20)

Burroughs – Wheeler xform.

One recent ( ) ⊙ idea was use of Bernoulli: Essentially Z141. use of PPM/Burr Wh. Xform to find concs rapidly! Also look for subtree concs (≡ contexts). Also expansion of Z141 into CFG & CSG. This would give lots of concs & not bad prdn. Could I then introduce more general forms of S-functs? One more general format "context" is a d function strings: It amounts to a funct that tells if a string is a member of a set of contexts. If ≡ an ngrams. This d. funct. is one way to desc. a ngmst. (See 4TM 20.00 for discussion of how to create new ngusts from old. Also has ideas on S-functs.

An "S-gram" of D-agusts is t. "S-agust" which is a s-grammar ≡ a p.d. on all strings. 4TM 20.08 ff discusses crossbar combinations

0: [SM] on Sorting for PPM: One fast (cheap) way: use quicksort or Heap sort. Then, when new info comes in, make a new file w. new data sorted by itself. mini own file When creating new cands, use Both files. T. first (larger) file will usually contain all info used, but t. second will occasionally be used also. For t. second files we can use 2-3 tree or some expensive way to insert into, Because of smallness of t. files it will not be very expensive in Time or space.

Quicksort v.s. Heap sort.

25 → [SN] One way to decide on how TM should represent & search for S-functs: Write a TSQ in which includes a sequence of S-functions as solns. Then, for these particular S-functs list diff'nt ways they can be represented, & [redacted] give heuristics for finding solns. This will be initially done for Lsrch, but perhaps similar functional forms & search methods can be used in BU method. (34)

PCof "end symbol" pcof "end symbol" is determined by experience w. N problems in past.

. Try listing all "solns" (to date) of S function generation/search problem. Also note that things I wrote on soln. of 3 input uvnc. "P 33 or 133" — yellow paper I think I found a bug in it — but I'm not sure

How, I have models of 3 input uvnc. forms of S-functs that may help. I use a functional form for PC in terms of code length, based on past experience; one way is to have special "end symbol"

4:(30) One general kind of TSQ would be t. kind of data used for Z141: use Z141 models, then ngrams, CFG, CSG ect. This gives rise to a particular kind of models for t. S-functions. How general are they? — How Can they be gened? Most fancy models have continuous params, these params are automatically calculated, not "srched for". In (nonlinear) linear regression & curve fitting, Srch is always over a discrete space & contin params are calculated automatically (not "srch for") calculated "automatically"

See 302.00 for Good ideas on S-Functs

275.00

02/04/04

4PM

Span 10:272:40 ≡ [ZBN] in -1.31 - .40, it's not clear if I was thinking of INV, IND or OZ probs!

02 In -1.31 - .40; The idea is to decide what population to use when starting on a problem;

03 past populations consist of individual(s) and their Gore(s) w.r.t a particular Goal (.10)

In line w. the idea of .00 INV probs seem to be not so good, since they don't have info on "near misses" (i.e. Gore has only 2 values).

Perhaps Don't try to solve INV probs, but discover ways to convert them to OZ probs Like "GPS" — with its vector Gore.

On the other hand IND probs using Lsrch are a kind of INV problem, because of the particular Character of Lsrch we do get continuous Gore values, but not many of them, & they are usually not easy to compute because they are for different corpi .... Hvr. These objections may fade away after we work out a way to do GP on TSQ's!

(.03) 272.33 One way is to look at present problem & decide what probs in past are "similar/related" ... Developing useful relations of this sort is done by improving TM. In tight TSQ's "recency" is an impt. relation. The "compression trick" of Li-Vit (& others) is also useful, & can be improved when induction (≡ compression) methods improve — so it can be a self-improving technique.

14

15 [SN] In -1.37 - .39: Say we were (not never) able to find a single O^j that did the entire corpus, then to do a prediction, we could use the "closeness of Q" relation to decide which O^j would be best for a particular new Q (prediction). This idea seems close to the much older idea of having a "function net"/"looks at a new Q & decides which functs to use to (help) solve it.

20 { Alternatively, for induction (& other OZ probs), one could assign a "default" value for the QA's for which one has found no code. — So one always has a code for the complete corpus. E.g. for some Q's the O^j may give a pc = 0 to some A's. We assign a certain amount of the pc of all "O^j outputs that don't include all A's"} & we use this p.c. for "default codes" for all otherwise "unreachable" A's.

(SN) on SN: Pooling Coulomb compression of several choices, leading to wts of various various strategies or a power of at the pc of each? Power of 1 means usual method; > 1 gives "best" strategies; < 1 moves toward = wts for all strats. (combine?!?)

So 3 Apparent BIG problems

6 (1) How to use info from previous problems & problem solns. (.10 is an impt way) (.32)

Def (2) The Lsrch v.s. BU method (BUM). — If we do both simultly, how can we share info best?

Soln. to (1) may be diff't for Lsch v.s. BUM.

(3) (I hadn't noted this first time around) How to represent, srch, S-functs.

0 Another approach to (.10): "Update" is not done until a (new Q_{n+1}) (predn problem) comes in! We then look at Q's (i < n+1) that look "related to" Q_{n+1} & we "pass a grammar" thru their (Q_i, A_i)'s.

272.40

32: (.26) This can be done gradually; First we learn to solve individual problems — using, to some extent,

33: (.26) (codes) (bags) from previous problems. But at first, TM has no way to "recognize" a problem & decide what codes to use. The weakest possl. interaction is that of usual GA, Chill. Even after he solves a problem (perhaps in batch, rather to try Mahal or other ancient (or future) (ongoing) creations involving designing hill.

In (.33), the problems in the TSQ are related, but a "solution" doesn't have to be able to solve all of the previous problems — so we don't do the "conventional" QA TSQ at first. At first, our goal is not prediction! We are mainly interested in getting common solns for as many of → 274.00

9/19/04
4TM

~12

.20: (spec 271.40) : What are H cues for deciding when to put a D.F. at a particular pt. in the generation of the cand?

(Vaguely): That 2 or more Token choices at that pt. seemed to be of high "G".

In Lsrch, we will tend to search each branch pt. in a tree exhaustively.

In QA: Wouldn't it be reasonable to spend fraction of time "working backwards" from A toward Q? This is certainly an "Old" A.I. research technique.

Say one has found, viz PPM & Lsrch, good code(s) for $[Q_i A_i \cdots Q_i A_i]_{i=1}^{n}$.

PPM then gives a D.F. over (O^j_i codes) this set of $\geq n$ objects. We use this D.F. on Q^j's to get a D.F. on $Q_{n+1} \to A$, ... (I'm not yet clear on this ... But ... ⊕ ... this looks like

.09 a nice way to get **Some Wo. Backtracking!** — Since at all points in time, we have

.10 a universal D.F. available! Here the D.F. is on Deterministic O^j funcs, based on a corpus of $D \cdot O^j$'s that have been successful for $i=1/n$. ; I'm not sure that this is what I want!

Consider the set of short (deterministic O^j) pgms that have been able to do $Q_i \to A_i$ $i=1/n$

$\{O^j_i(Q_i) \to A_i\}_{i=1}^{n}$. This set of O^j_i programs, induces viz PPM a P.D. on O^j pgms.

If we use this test on ANL starting w. $+, -, \times, \div$. we would at first get a ppm.

(w. input $OP_i, 1, 7$ $\{op_i = + \times - \div\}$) output (P_c: + .25, − .25, × .25, ÷ .25) A_2, A_3
$A_1\ A_2\ A_3$

We could continue the srch to get better pc's for the corpus. If we allow "non-blind srch" T.M. could look at pairs of probs & successful solns. (which seems more like BU method)

.20

.21 The foregoing stuff is involved w. very general s-fault models & general search methods. (otherwise difficulty found)

Lets go back to GP: its 4 or 5 difficulties & how I was going to fix them!

258.06 – .22 is a relatively recent plan. (10/5/04)

250.11 – .29 ; also 252.10 – .14, .20 – .40 — 265.34: Not being able to watch cands being tested.

235.06 – .40 : "4 needed improvements of GP" ‖ Much Later I found a fifth. Also (6) (earlier) that idea of using faster approx. of Gen routine, in early stages of GP.

.25 → 1 Speed: Better models & model fitting algms for Gen func of some params — related to (6)(4)
    use of PPM v.s. Lsrch for both.
2 profit from past problems solns.
3 TSQ
→ 4 Lsrch? Also Q of Lsrch v.s BU system — like GA. or more optimized srch routine like $M \epsilon j$ ($= \frac{\sigma^2_G}{G_{max} - M_G}$) : or other ways to Model G as a prob. func of some params. Actually (4) is related to (1)
5 Being able to watch phenotypes in action (fail/succeed). Being able to watch Gen being applied.

.29 → 6 When Gen is Expensive, use fast rough approx. in early Generations & related to (1)!

.29½ so (1, 4, 6) (2, 3) 5. So More like 3 general areas of improvement.

.30 → Try to outline what I would do if I had to do a TM in 1 wk / 2 wks / 4 wks / 8 wks ....

.31 One simple way would be to use GP (w. or w.o. various augmentations) (258.06 – .22): The most interesting / impt. problem is TSQ / learning from previous problems.

Ways to do it:

.33 1) Use Lsrch-like TSQ: Solve each problem in order, keeping in active populations, cands that scored well in past problems solved. We can do this by saving the "k best" of each solving population or take the "10k best" & select k from them at random; This gives larger σ^2_G Variance in Gen.

Note +1.00

.37 2) Solve $Q_1 A_1$ to $Q_n A_n$ individually, using previous populations — but a soln to $Q_i A_i$ need not solve the rest of the QA's (yet). After we have these individual solns, we try to find common solns to pairs of QA's, then triplets .. etc until we have a soln for them all. → 273.32 spec → 273.32 → 273.00 spec

4TM

-0.01 SPEC .0 ! 270.19 ! → This is for Continuous D.F.: The Discrete D.F. seems much different: I have to find good way to do it! See (.33)

In the case of Lsrch: If I have a uniform d.f. over 0,1 as some of the elements of the domain — I will start the Lsrch with 0 & 1 as possible trial cands. ~~...~~ — because = for each possib.

Next we try the new 2 bit codes of $\frac{1}{4}, \frac{1}{2}, \frac{3}{4}$, (~~I'm not sure how we started 0 & 1~~); if we ~~treat~~ $\frac{1}{8}$, use TG27 we do $0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}$ on next round — Then

.04 $0\ \frac{1}{8}, \frac{1}{4} \cdots \frac{7}{8}$ on next round.

I'm not entirely ~~of~~ clear on .00–.04 — There are other symbols not code! — but

07 this is Lsrch, & we do try things in pc order.

Note that Koza did have a way of using GP to compute numerical params to hi precision. Just how did he do it? .... I think there is a Mt Carlo version of (.00–.04)

~~Add~~

09 If it's not Lsrch, then we can just use a normal s-grammar (or a GP population based s-grammar)

10 to generate Mt Carlo cands — using something like MEJ — ("Max Expected Jump" $\frac{\sigma_G}{G_{max} - G_{avg}}$)

.11 A possible serious diffy w. GP: That one could often come to branches & a one "climbs up the landscape". Normally, GP makes "decisions" at each branch & continues. To some extent this is "fixed" (or made less bad) by doing many GP runs out. Same problem — so alternative branches may be chosen. [I'm not sure GP "takes a specific branch": It can have population members on several branches.]

— Lsrch does have advantage of looking at all branches (up to current CB level). — but it's looking for a somewhat different thing. [— so how does it do "Premature Conv" = "Local Max"?]

There are "Multiple Goal" GA schemes; It may be that one can use them to investigate "Branching paths" better.

Ideally this (.11) branching should be addressed by a "subgrammar" (S) for each of the branches: The subgrammars would, of course, have many common concs, but would correspond to different ~~...~~ Branch of a ~~...~~ generating tree.

In a cross/mut system, we'd want parents to be (mainly) from the same Branch, if we wanted to do a Branching exploration of the "Fitness landscape". ← How one identifies the "which Branch" the parent is on, is unclear! Would use of a PPM type "Grammar" be able to deal w. "Branching" of this type?

---

27 (.00–.10) (Note .09–.10 in particular) does seem like a general way to ~~...~~ search solutions.

28 I'd like to link it to the Bernoulli ideas of 269.34 & 270.27–.29)

In (.00–.10) we chose pc values & try them out on the corpus. In 270.27–.29 we chose elements to do Laps rule on, & we computed the pc's. We should get about the same pc values.

30 So first, I want to be sure I know how to use .00–.07 for Lsrch! Then, generalize so I can do (.09–.10); then see if I can fit in (.22–.30).

.3 : (.01) In line (.01) I noticed that the Discrete D.F. seemed different from Continuous: i.e. ~~Discrete~~

Discrete D.F.: p_i ($i = 1/h$) the p_i's pc of tokens at a conform context. (= perhaps uniform d.f. modified by function)

Continuous is uniform D.F. over 0,1 mapped by some function onto a different range so it gets a different (usually non-uniform) d.f. In the first case the D version is a single token at this context. In C. often it is a single real (or complex) no. at a context.

In both cases there are ways to find "best fit" for params that define the D.F.

In the Discrete case, Bernoulli (Laps rule) & in the continuous case, maybe similar trick: (say divide 0,1 into a few divisions — then use finer divisions (more bits) if necy / warranted. (Spec 272.00)

9/16/04

4TM

00:269.40: 4) A variation on 269.34: We have a Universal mach set, S so strings of its symbols are ≈ "Lisp" pgms. Grammar G_2 is a (perhaps) CFG whose leaves are strings of S symbols & are ∴ PGMS.

G_2 generates CFG's. By looking at a corpus of S pgms, we can devise G_2 grammars for them. We get a density d.f. on S pgms. I had the idea some time ago that this could not give a s. funct that was universal in the intro because — I want (& wanted) a universal d.f. on s. functs. The idea being that I had a way of generating s. functs that was universal — so it would assign pc > 0 to any finitely describable s. function. (Want a universal s-funct D.f. on s-functs.) Well, 4 (.00ff) seems to do this. — NO, it doesn't! It is a (univ'l d.f. on s-funct) on D-functions.

07 Any (d.f. S funct (univers'l or not)) on s-functs is itself an s-funct.

.08 Any d funct on s funct is an s-funct.

09 Any s funct on d functs is an s-funct.

10 Are .07–.09 deceptive? I.e., a s funct D (i.e. P.d.) on a (D or S function f_1) is a function of the form of f_1 & is indep of the argument of f_1 — I'm not talking about a composition of functions.

13 In learning "d-functions": I use defns a/o (contexts ≈ PPM) for induction: to create new cands. If my original primitives are s. functs, then my old induction methods of defn & contexts will give terms that are sub-s functs & contexts that are s functs. — we will end up w/ s functs of (possibly) arby complexity — prob'y all possible s-functs could be generated, so the method is univl. — How to do this?

.17

While .13–.17 seems reasonable, I can't think of any examples of how to use it! Well: could I use it for the Bern problem of 269.28? (If it works in Csusy, perhaps it would be clear! → 271.00 5 Sep)

19

In APL & other similar langs there is an instruction that combines 2 1 dim arrays by $+ - \times \div$ $\vec{a} \equiv (a_1 \cdots a_n)$ $\vec{b} \equiv (b_1 \cdots b_n)$: $\vec{a}\, Op\, \vec{b} \equiv (a_1 Op b_1, \cdots a_n Op b_n)$

I may want to use this sort of thing to assign pc's to a list of "Tokens" or whatever.

(NB) There is a common (or just as in PPM or "Augmented" PPM) Bart type strategy in which one defines a context & this automatically defines a p.d. on the following Token. In this sense, the idea of the context is a complete defn of the s. funct. A context also can be used to define a pd on things longer than a single symbol —

26 A token can be an item of any size — (or perhaps be a s-string (≡ P.d. on strings).)

27 — In ~ .13–.26 Consider a common ("Bern" oulli) method of induction: One does this solely by defining items, itemsets, functions, contexts. Each of these operations can ↑ pc of corpus &

.29 — Can use previously defined objects as "parts" of them in its new definitions.

?→? An arbly complex O^J function can be thought of as a special case of .27–.29 ← [Later: How?] ? 35

30 Heuristics (for "suspecting" that a new cand. might be constructed in a class of ways) are also constructed of old theories & "parts" of them.

Initially, all of this can be done in PPM style, using BW xform. (wrought/harder)

Later we look for sub-graphs (fronts).

35 Re: .30! Perhaps this: That O^J operates on the string of $Q_i(A_i)$ pairs in the seq (≡ corpus).

0/16/04
4TM

Spec.
00:268.40 ; A trouble w. 268.36 : Its quite expensive to say which symbols to change, and to what: A better kind of S-function will have the symbols to be changed (Indexed Marked) — which is like 268.28; but we could use a default d.f. for the symbols, or use the d.f. implied by a recent context.

Just how to assign pc to forgy. is unclear! Say I use a special symbol in the code to denote the d.f. of symbol values. Say I have 4 of such symbols in my code for a cond. Then, to evaluate the pc of the "A" value that I need, I have to scan over all combinations of all 4 "special symbols" to find a comb. that gives the vite A. The pc's of "special symbols" is I, (boost 20). The d.f. implied by the "special symbols" gives the final pc of [illegible] code that produces the vite A.

One might have a standard "Correction Code" that is used to "Xform an unsatisfactory A, into the correct A value.

→ How does forgy. relate to Maxsize of pc of O^j functions in QATM??

Do the pc's of the special symbols correspond to the "R" values in the "simple" model of S-functs?

Ideally, I'd like to be able to take any O^j function & find a pc (>0) for the corpus wrt. that O^j function.

T. problem Seems to be: I have a few ways to represent S-functs: Now O^j's are S-functs. What are good ways to search over a space of S-functs, w. goal of obtaining O^j's that assign (w. its appx) [illegible] Hy pc's to the corpus. w. popular terms or intervals

I have a few such methods: ① Lsch ② Monte Carlo GP search, using Mut/cross ③ Monte Carlo search of outputs of S-Grammar.

Woops! In general, one can't simply replace any symbol in a pr. in a funct, w. any other symbol: we have to replace them w. symbols of same "arity".

To study S-funct forms: Give many Examples of S-functions that could be solns of QA probs.

1) In very early ANL, TM has learned each of +, −, ÷, ×, but doesn't know which to use! — so for all binary operators there are 4 possys in some known exprss like α, 3, 7 (in PN) in which α can be [illegible] the add, sub, mul or div w. = proby. The probys themselves could be obtained by a Bern seq. analysis of the corpus.

28 2) Consider predicting a simple Bern Seq. w. known finite alphabet. We have a Null (≡ constant) Q. So We [illegible] want an uncond. p.d. This seems like a very fundamental problem. How best to express a k-way probabilistic jump? Perhaps have it be a "primitive concept": We have a random no. betw 0 & 1. There are k intervals on [0,1] & so the random no. chooses one of the k:
k: w. pc (depending on the k & to interval size).
For the search process, we could start out w. k equal intervals,

34 3) One method of representation of S-functs and consider them is use of S-CFG's: Start out w. a simple Bern Model (promiscuous model) — Then look for production rules. The "best" probys are computed by parsing & counting no. of times each choice was made. ≥ This is usually a reasonable Approx..

I'd like to generalize this S-CFG to universality. Context Sensitive Grammars can be universal, — But I'm not sure this is a good, useful representation.

Spec.
→ 270.00

0/15/04

4TM

50 · 265.40 space : Doing both Lsrch & BU method seems like a good idea, but I would like to Link them — so they (265.00 space) help one another! Common s.ftns, defns, is one way. Both are Guided by PD's: How similar are these PD's?

Well, maybe the 2 PD's aren't so difrnt! I have been thinking of (D) IND probs for Lsrch, — not (S-IND Stochastic)! If we do S-INDuction, then Lsrch examines cands in order of their a priori of decns. The cands, of course, are s.funcs of the individual QA's in the corpus.

In BU method, we have a PD on s.funcs also. These s.funcs (as do those for Lsrch) assigning a PD to each QA. Equivalently, for each Q they give a PD on all possible assoc. A's.

The 2 pd's don't seem so difrnt! The Lsrch P.D. looks at "successful" cands (s-funcs) & induces a D.F. on s.funcs ... the pc that each one will be "successful". Note that "success" is a soft quality.

The BU method D.F. also tries to rank cands on expected future "success" (the same soft quality).

The 2 methods seem to be looking for the same thing — a cand w. much success w. current corpus. They could, indeed, use the same PD on cands!

Hv. BU using MEJ, would use a certain subset of cands to generate a s-lang (≡ pd on cands) And do Mt. Carlo trials on that lang. Lsrch would simply try to list cands in that lang. in ≈ pc order.

{ Also note, it's not clear how the Mt. Carlo srch of BU deals w. non-terminating cands }

Also, in general, I've not dealt w. the softness of the Search Gore for both methods — in both methods, we just search & keep track of the "Best K" cands thus far (or we sort all of the cands found w.r.t. the gore?)

In BU, there are at least 2 approaches: One, classical GP, uses Mut/cross to generate Mt. Carlo cands. Another makes a s-grammar to generate the cands. This last is closer to the Lsrch method, since we can (sometimes) use that s-Gmmr to generate cands in pc order.

The foreg. suggests a unity of Lsrch & BU method: The Generation of S-funcs may be very easy: All you do is include 1 or more sfuncts as "primitives". Some useful sfuncts to use:

28 (1) Uniform d.f. of x on interval 0,1.; for discrete d.f. (2) mult, div have pc's = ½.

This second is not so hot! The first I can xform to any d.f. I like w. suitable params.

I don't see how to xform the second!

[One A.H. way: Include a param that gives rel. pc of 2 choices: the param can have any amt. of precision, but pc ↓ w. precision (costs more bits).

Another way to deal w. PD's on discrete params is to use || codes. — one code for each value of the discrete parameter.

---

6 A related approach: Brainy idea! We have a d-pgm D that does Q→A. To get a s-distrib for A, we make changes in D to get it to create other A's. The info "needed for these changes is then added to the bcost of D, to get the pc of the new A. A simple way to modify D, is to change individual symbols. next, pairs of symbols, etc. → 269.00

4TM

ANALOG: Rite way to do ANN (Also RANN, but finding peak takes longer)
.03-.06 is rite way to do ANN (& maybe RANN).

: In normal feed forward ANN: What is normally done: Pick random initialization pts & recourse hillclimbing until they get a peak on "test data". Repeat this random choice run many times, then average y. results.

Variant. Unnecessary to have "test data": Do as here but don't stop until final peak is found. (Rs will usually be grossly over fitted). At each peak find "size" of peak in (finite) param. space (possibly by local M. Carlo trials). Do Ry so many initial pts & average, using as wts, y. local size of each peak, mult by its Gauss pt. "fitting data" → S ≈ 293.20 on Rs

A possl. objection to .03-.06: That almost all final peaks will be narrow & ∴ no better Gauss for tng. data. That there will be only a few "Good peaks" & Ry will give quite different peaks. As a result, one will have to run y. prem. a very large no. of times to get an acceptable value

Sample for a good peak.

To speed up convergence use that v.g. non-linear optim method. (— · · Marquardt method)

This method also gives "width" of each peak it finds.

In terms of no. of trials needed to find peak each, .03-.06 looks like it might be same speed (same no. of trials needed) as Esch(?)

RE: .03-.06: Just how is it better than choosing params at random, then using produ. for that set of params by type of y. various wts that set of params? I think .03-.06 is more efficient, but it really has to be shown. Each peak has a "basin of attraction" There are probably many fewer peaks than no. of pts we would need to get accuracy viz. .16-.17. Hm, This is true only if y. model is good enuf to give only a "few" "Basins of Attraction".

Anyway, foregg discussn. strongly suggests that ANN could be very useful for learning real functions. For learning Discrete functions? I'm not so sure!

SN I had this idea of using a (large? large?) Nucleus of an Atom as a "very fast" dynamic system, so was could get it to do computation very rapidly. One big difficulty was getting in & out of Nucleus. One way might be NMR. — That is computation could change the Mag or electric (or whatever) Moment of the nucleus — which could be detected by NMR. It may be that almost all nuc/reactions in reactor are very short lived, so difficult to detect — i.e. life is ≪ time of possession of Nuc in Field of accessable strength (say 9T = 90K gauss)). Normal NMR MRI is ~ 1.5T = 15K gauss. There are some nuc states w. very long lives (Years) ··· but I think they are rare → ? But Actually any nucleus, stable or not, can be regarded as a "nuclear state" ··· . There are certainly lots of nuc. states w. all sorts of ½ lives!

10^{-4} cm $\lambda \cdot f = 3\times10^{10}$ cm
$f = \frac{3\times10^{10}}{\times10^{-4}} = 3\times10^{14}$
$= 3\times10^{5}$ GHz

ANALOG Compn : .ooff

DA: primitive processes are Addn, Multn, integration. very reasonable

Analog Compn.: Differential Analysers, Digital: DiffAn "DDA": Shannon paper on DA.

Paper by Chris Moore (PS: 10/13/04) Also Moore's paper in 1996 on how to map dynamic systems into a Turing. Could NMR be used to probe state of nucleus, that was doing some computation? ~~Also NMR~~

If so, it should be possible to use NMR as input to ~~exciting~~ nucleus.

Just (how) is a Diff anal. "Universal"? Can it (at least) do all prim rec. functs?

All DA's are of limited precision. Can we use a finite no. of them to get arb. good precision — like Shannon's channel capacity Theorem. (Actually Moore was the book by Conway & Wiegand on how to do this w. ANN, I think, but I never figured out how it worked. McCulloch was involved in this to some extent. Shannon did some work on this, but I don't know if he got any very useful results (He called them "crumby components")

One idea of "universality" that we have is a kind of set of interconnections; but a set of (continuously) variable inputs. That any configuration can be simulated by a suitable set of analog initializations!

To convert an analog value (X) into a switching signal: integrate x (v.s. time). It will → +∞ or −∞ (depending on its sign). If we have "threshold" functions, $TH(x,y) = \max(x,y)$ Then we can decide if $x > 0$ or < 0. i.e. if $\int_0^{+\infty} x \geq 1$ then... or if $\max(x,0) = 0$ then $x \leq 0$. $neg(x) = -x$ can be done by Multiplying x by −1.

To simulate a ("non-connection") between 2 operators, have multiply via "wire" between those 2 operators by 0 or +1.

Also, there is (perhaps universal) "chaotic computing".

For induction it may not be necessary to have exact computations: Say we have slightly "crumby" components: We scan thru input config/param values until we get output close to corpus, & we observe "next output" (= pred). We then repeat calc. w. same & nearby input values, & get D.P. on "next output". We may then continue scanning over inputs to get better "match" of reproduction of corpus. ("closeness" to corpus can be RMS, ~~or~~ in ~~Sak~~ R windo or xwindo). How could we do ~~an~~ analog Time Series Pred? Cross-Pred? (subsequent corpus)

There are 2 inputs to "Net": ① Config of net ② input data. We also have to have a way to efficiently "postprocess output" (perhaps have several possible outputs & decide on "best one" post hoc). ~~We~~ for each config. input, we run several ex. to get i/o pairs & see how outputs correlate w. "correct" pred. Appropriate an input config would be related to (identical to) the density of inputs in Real input region, that give outputs "in a region"

ANN's are an interesting example: We can have connections fixed & vary inputs; or vary/train connections & inputs. Another large expansion is RANN (Recurrent ANN). In both cases (ANN, RANN) most of the remarks of ~20–30 hold.

SN In normal ANN (or maybe RANN as well?) We normally use first partial derivs to guess at next trial. If we also used second order derivs & solved resultant eqns for extremum, we would get much faster convergence! How to adapt RANN to this idea (or even first derivatives) is not immediately clear.

0/12/04
4TM

00:264.40: Note that after t. TSQ of "phase 1" TM itself has a choice of what induction methods to use: So it should itself be able to decide on Lsrch & v.s. BU method — or how much (relatively) time to spend on each.

03 During normal "Adaptive Lsch" TM develops an assoc "GPD". Could t. same GPD be used somehow, by t. BU method? — Or would the BU method have to develop its own techniques indep'ly of t. Lsrch's "GPD"? If either method uses "dctns" or "srtns" they might be able to use one another.. On trying to "mix" data used by 2 methods → 268.00
06

Q → Very Impt. Q! Compare "Updating" process in Lsrch v.s. BU method.
10 Also, compare srch procedures.
In Lsrch, t. search is always "standard" (say, TE2T) —
11 — In BU method, there are many possl. ways. — Tho in GA ⓑ & MEJ, I had been considering populations in which we could find deterministic or stochastic constraints on t. population to control $\bar{G}$ & σ_c in a (computable, approximable) known way.

Xtn. strings for UMCs. If they are partially ordered, they can be dealt with as a Boolean algebra. Is this useful? Usual times order linearly order! Can UMCs be linearly order'd? Maybe, but perhaps not "effectively"

In Lsrch we have population ordered in probability that a individual will meet criterion; t. criterion being (for IND or INV) problems — that it is an acceptable code. (Yes/No). So induction becomes an INV problem. T. updating of t. GPD involves adding definitions, 2 Rhyn pc's, (partial) (context dependent) conditional pc of next symbol (if we create cands sequentially). Updating can occur by adding positive or negative cases to
19 corpus, or by statistical &/or logical analysis of "traces" of t. system. (.30)

20: 242.35: Another approach to Bifurcations in SRT. If there are "too many" of them, the cardinality of t. set of UMCs will be too large: i.e. uncountable in t. sense of "Reals" (continuum).
Consider t. branching factor as a function of length of such ducts.
T. no. of Tmc is countable & t. UMCS is a subset of Tmc so its cardinality can't be that of t. continuum.
— [redacted] We got that same result by t. "proof" of 242.35 – .40, which invoked randomness.
So maybe a relationship betw. randomness and cardinality or .... ??

0: (.19) For SS2 = 0 we have standard sub cands for Lsrch: [redacted] In Lsrch this zero SS2 case is dealt w. by t. "precorpus" that gives > 0 pc to a universal set of symbols/insts/tokens.
— For SS2 > 0, we look for various regys in t. corpus! Bernoulli first, then perhaps ngrams, n grams, cart multi rules for agms, etc. Context dependence, common subtrees,
34 DNA type analysis of codes, etc.
35 I did try to "soften" t. Lsrch success criterion by using sample sets of t. set of problems (of QA's) TM is to solve (i.e. from t. TSQ). This way, each cand would get a score > −∞ if
.37 it solved > ϕ problems. Actually, t. Geco criterion is complicated: it's t. list of successful codes w. t. pc of each code.
Just as in .32–.34; I BU method, updating can look for all kinds of regys in t. corpus (which includes ~~g's~~ & soft "a" info (not just a = +∞ or −∞)). (35–37) tells t. Lsrch does similar analysis.

A mean bcost could be used: but all problems are not of = difficulty!

268.00

0/11/04
4TM

try (mutation) a moose w. no tail to be sure the cut-off ceremony were still being done. (like circumcision)

Anyway, GPS does look at testing of cands. to suggest modifns in "Next round" of trials.
Lsrch can do this a little by looking at common characteristics of failures, or "Near Misses" ≈ hy but not very hy G cands. I think it does this in Phase 2.

[Built Phase II uses (TM2): Phase I does not. So in any Lsrch TM.... — we do lrn. to work for kinds of probs that TM2 would solve, but we don't yet apply this capability to TM itself.

→ Another feature of Phase II could be observation of Evoln. of Cands: so it can understand their success or failures. (It may not be necy to put this in Phase 2).

[SN] Perhaps look up {Elitism, Genetic Programming Algorithms} Google. See if anyone has discovered "optimum degree of Elitism".

In Jurgen's OOPS: It may end up w. progs that have max pc of jump to → Gmax — when it works on O2 probs — which probably usually will also be Mej!

[TDM Method] Out 2 basic methods of solns to inductive problems: [Top down method — TD method. Lsrch method. TDM ≡ Lsrch]
1) Lsrch finds cand. w. max $\frac{pc}{cc}$; taking time ∝ $\frac{pc}{cc}$;
If it doesn't find anything it finds Nothing!

[BU of Method] 2) Common method starts w. A.H or Previous Grammars & mutns/crossovers or other such methods to try to ↑ boost. — But at any time, it has (a) best found thus far. (b) "" "" ≡ all cands thus far. [BU Method may start w. Bernoulli dev. of corpus: (in some, they're all) cases, this is Both A.H. & Promiscuous!?? [Bottom up Method. BUM ≡ often BU Method "default"] ≡ "random", "default", No! AH & from Grammars are [illegible] quite default. If Grammars transforms may have same PC, hvr.
It, too, has a threshold better it finds something better at all than initial code.
(default need not be random; it can be Lapl rule code (Bernoulli)).
Usually t. cc for threshold of Lsrch >> threshold for "Common" method.

→ One Argt for Lsrch was that if one could find a soln, t. TSQ was inadequate.
In such cases, trainer has to ↑ IPC of system (more time, (many) or ↑ probability by "hints".
In Real World, the trainer isn't around to give "hints"!
Hvr, in Phase I, trainer is present — and anything that gets thru Phase I (: its capability of doing Phase II problems) is o.k.

So, perhaps pursue Both strategies simult. (Time share, perhaps) ← easily parallelized.

T. TSQ is designed so that Lsrch works w. it. Whether TM can continue w. probs that are not structured so that Lsrch will work, is unclear. If an Lsrch soln. is possl. w. t. cc available, I'd certainly want TM to acquire it, because it is indeed (almost always) a v.g. soln. (i.e. very hy pc.).

In RW TM will perhaps often be asked to solve probs in which t. Lsch would not be able to get soln in available "time". Hvr., A priori, one doesn't know this.
If one starts out w. BU method, one will get some f.back on how things are going, by amt of ↑ in boost of corpus pc or cc. Hvr, Lsrch seems to afford no such f.back!
It either codes t. corpus or it doesn't code t. corpus.
In a TSQ, if t. trainer thinks that t. problems should all be solvable by methods t. trainer is aware of, he should be able to write a TSQ for it, or TM should be able to find a soln within maybe 10 × "soln cc" constructed by Trainer

6/9/04
4 PM

(134)

? many known have cognate do water; Cands in rotation

:0;262.40 : This "Optimality of Lsrch" is also tied in w. Subscript (Sz2) in Sol 89 (P5). Perhaps to the effect that any conceivable heur can be expressed as a modifn. of the GPS for Lsrch. [I think one can realize this: Any Legit heur is ~~~~ logically &/o statistically derived from observations on past.
→ As such, it should be poss. to express its results as modifn of the GPD. (some @ of this/asd)

Hvr. This technique of heurs will occur in Phase II, when TM is looking at prob. soln of 8 past.
I recently on 6/10/04 wrote a dende to Commentsal "P5 ~~footnote~~".
One Conclusion: GA & Adaptive Lsrch could look at Phenotype directly or ... (.34)
~~watching~~ watching evolution, to suggest ~~improvements~~ — Hvr. GPS can't do this

:26 & trying to use this info to speed up present Srch.

1) So; write review of S89 P5 f.n. & include it in Commonsal S89 folder. [Mention "unsolved" problems "(goal/reason)"
2) Note area of superiority of (augmented) GA over Lsrch. q.v. 262.37-38
3) Another impt pt. in GP v.s Lsrch is ~~the app~~ difference in approach to IND: (261.30-.35 also
4) After doing (1), write a review of the GP v.s. Lsrch problem: This should ~~~~ get me back "on track"!

0

N.B. MEJ is one way of modeling G as a funct of Cand. derv. Tho it seems to work well for certain derv methods, it is definitely not the only way to model G as a function of Cand derv! Another (not fully developed) way was to try to model some monotonic δ(G) funct of G as the pc of some S-lang. (S-Grammar). I haven't (yet) done much w this last.

Also Note! Any expression of a G d.f. as a function of cand params is an S-lang.
(Any ~~~~ p.d. on any arguments is an "S-lang".)

Actually, MEJ doesn't model G as a funct of cand derv. It finds a lang of ~~~~ MEJ, then MC Carlo tries its population. This is a way of getting hy G conds! It's similar to using a "selected" tournament size, m (for MEJ) (q.v. 261.00-.07) in ordinary GP.

0

-1: ~~2.00~~ 2.20: ~~How to Get Good MEJ ...~~

[SN] On the Universality of feed forward ANN! My complaint about the proof was that for a function in N space in a unit cube of resoln Δ; $\propto \Delta^N$ params were needed to describe the funct.
Hvr if the original values of the funct to be simulated was "random", there is no better way! — the argument is with the constant of proportionality — a coef of Δ^{-N}. How much (if at all) does it deviate from "1"?

[SN] on ANN: In feedforward ANN! Conventionally, one does ~~~~ repeated runs w. randomized initial conds. Result is like "ll coding" & cancels out pc of ~~~~ minimization params. [A similar thing ~~is~~ is often done in GA for same reason]

0 So it sounds close to the correct ALP method! Expensive, Hvr. — Not so expensive if one keeps track of the s.d. of ~~~~ Macrotrials & ("Sequential statistical analysis") & quits when ~~acceptable~~ acceptable precision is obtained.

:.(08) [S.N] This Deficiency of GA & Adaptive Lsrch! — Not being able to watch cands being ~~tested~~ (evaluated) so as to be able to suggest (corrections, changes, modifns, improvements) for next generation of trials.
235.40
Is this would being able to do this be kind of "Lamarkism"? Usual example of Lamarkism is: Tails of all mice are cut off soon after birth — results in mice born w.o. tails. If the "modifyer" were watching the tails being cut off, he would see they would have no utility & not issue them in subseq. generations, if he thot the cutting off would continue indefinitely — He would occasionally

7/9/04 ~62

4TM SM .00

0: [SN] **SM**: A ≈ "Non Parametric" way to do Time Series predn! Consider 2 T.S.'s!
x & y: y (series) is used to predict x. We use ≈ PPM, but in a continuous way: To do k history prediction! ~~take~~ at time t, ~~last k values of y, from~~

predn. at time t is $\sum_{j=t}^{j=t-\infty} x_j \, f_\alpha\left(\sum_{i=1}^{k} (y_{t-i}-y_{j-i})^2\right)$ — or as much data as one wants to use. Σ error input for predn of x_j

$f_\alpha\left(\sum_{i=1}^{k}[y_{t-i-1}-y_{j-i-1}]^2\right)$ ← or use absolute value. Some other (symm?) function

$f_\alpha(\ )$ is the wt. of predn. x_j. It is a ↓ funct of error, f_α

This depends on α. We optimize α for the relatively recent data in the T.S.'s.

Q: If this method signif'ly different from linear predn. w. k coeffs? For the sequence (.14), I should run it. It would be better than linear regression. Hvr, it may not be better than ANN. Both are essentially non-linear methods.

$f_\alpha(z)$ might be $e^{-\frac{z}{\alpha}}$; we optimize α. Also normalize α so $\sum_{j=t}^{j=t-\infty} f_\alpha(\ ) = 1$
i.e. wts. should sum to 1. We may want to ↓ f_α as $t-j$ ↑ ... This would make normzn. possl. If the ↓ is exponential we have a "window".

So $f_{\alpha,j} \propto e^{-\frac{z}{\alpha}} \cdot e^{\frac{j-t}{\lambda}}$: where λ is the decay distance.

Other forms of $f_{\alpha,j}$ might be better.

4 I might try it out on the sequence $x_i = y_i$: i.e. single time series. $x_n = S\left(a x_{n-1} + b x_{n-2} + c x_{n-1} \cdot x_{n-2} + d x_{n-1}^2 + e x_{n-2}^2 + \text{noise}\right)$

S is a squashing function like tanh x

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1-e^{-2x}}{1+e^{-2x}} \text{ ``}\approx\text{''} \frac{1}{1+e^{-2x}} - 1$$

$$= \frac{e^{2x}-1}{e^{2x}+1} \qquad \left(-\frac{1}{e^{2x}+1} + 1 = \frac{+e^{2x}}{e^{2x}+1} = \frac{1}{1+e^{-2x}} = \tanh x - 1\right.$$

$$\tanh x + 1 = \frac{2e^x}{e^x + e^{-x}} = \frac{2}{1+e^{-2x}} \quad \text{So } \tanh x = \left(\frac{2}{1-e^{-2x}} - 1\right)$$

0: (Space 261.40) A "true" Lsrch would only consider cands that were codes for output, and it would list them in $\frac{pc}{cc}$ order.
So the "true" Lsrch has as input, what "real" Lsrch has as output! T. difference between the 2 ways used for Lsrch v.s. that used for GP is that they are arrived at in different ways.

"Restart"! One path was to greatly Improve GP! Actually The stuff of 258.06–259.05; 260.07–.20 –261.00–.11
"Hashing on improving GP." Note Remark of 261.08–.11

Still, lets compare ordinary Lsrch & GP on induction problems. The design of 261.30–.35 is reasonable.
In 261.30: Any "predictive context" is an (impt kind of) "short code" (hopefully "short"). Hvr, every context of this kind has a G > −∞ — so we tend to get some "slope info": which is not true in ordinary Lsrch.
The only "slope info" in Lsrch is obtained by "long duty Lsrch". T. optimality thm for Lsrch is based on "All info is in guiding pd" — which implies "long duty Lsrch": — ideally one would use dynamic pgmg to get info (by low expected G trials) to enable fast ↑ of G, in t long run: (Which looks like strategy for a reenforcement Machine!).

Hvr, I don't want TM to get into this "Dynamic Pgm" rite now. It should pretty much concentrate on "short" problems, w.o. possy (yet) of deciding to do "self improvement" during a problem. This sort of thing should be worked on by TM Later.

→ T. first problem is to get to Phase 2 as soon as possl.
.37 That said; is GP or Lsrch a faster (Better) way to do Phase 1? A bit of a problem: MEJ is beyond
.38 Phase 1 — it "understands" optimization better than Lsrch does in Phase 1.

→ 263.00

9/7/04 ~01

4TM ~~[illegible]~~ LM386

0 :260.20 ; T. Discussn of 260.07-.20 was for GP using a corpus in which indivs were put in bins, or strictly sorted by G. An alternative approach: Use tournaments to select top 2 for ~~mating~~ mating, Say we get 2 children: We replace the bottom 2 of t. tournament w. them (unless they are worse than the bottom 2 — in which case we discard one or both. The size "m" of t. tournament determines "elitness" of t. distribution (see 260.35 for t. def). We select a m value ∋ $\frac{\sigma_G}{G_{max}-\bar{G}}$ is max. How to get these values for t. population, must be worked out → No to that these values of σ_G & $\bar{G}$ are for the children, not t. population itself. → Just keep track of G & G² of cands w. G areq of x & y (like 260.07ff)

.07 (the "G_{max}" is for t. population).

.08 In t. forgg. models, the MEJ eq. was used to try to get hy G in offsprings. It would probly give results that are better than current GP. Hvr, I don't know whether it would be very much better.

0 A key idea is t. TSQ — While it would be nice to improve t. "normal" operation of GA, this improvement

.11 is not really necy for us to implement TSQ's. ||| It mite be very much better, since MEJ gives it a much better "understanding" of OZ than "normal" GP or even "Lsrch in Phase 1"

.12

10 All of t. recent ~~forgg.~~ stuff has been to Improve GP: What I've forgotten is t. special Value of Lsrch. for IND probs! ≡ Searching in G (≡ pc) order for a cand that will fit. An inverse way: To start w. known codes for corpus & try to find shorter codes, i.e. compress them. This last is more in t. spirit of GP! — It is one good way to get Grammars for Corpi (e.g. KSG diss?). We start w. Promiscuous a/o AA. Grammar.

.4 This sounds more like what "people" do: i.e. look at corpus & try to find regys (compressions) in it. With experience, one acquires a large body of regy types & means for recognizing when they may be relevant (≡ "Coding & Recoding"). ← (.24) can be done by Lsrch also! We have an "a priori" ordering of trial cmcs. (regys) that mite be successful compressors. Presumably, as we code more & more of t. tsq., ~~our ordering~~ t. pc's of our regy trials will "adapt/improve".

0 How does GP "normally" do induction? One way is to search for predictive contexts. A context could be a suffix of t. corpus a/o one or more conditions (obs) that corpus satisfies. Contexts can be evaluated by empirical Laplace rule, using not merely entire corpus, but "sampling". There seem to be 2 different Methodologies! (1) GP: A Biased M.C. search for cands (≡ "Predictive Contexts") of hyest possl G. (2) Lsrch: A search for cands (i.e. "predictive contexts") in pc order.

.35 MEJ is example of optimum Biasing in (1)

Some confusion in my mind on these problems: Do some "rambling": For IND probs, Lsrch does a "kind of" optimum srch ... if we know a soln., we know its cjs (for "pure" Lsrch; No long-term t.-srch). GP solves IND probs as OZ probs — by M.Carlo srch over a Biased S-lang. After some probs have been solved, Lsrch is "also" over a biased S-lang, but it may be a different S-lang. Lsrch does t. srch in what appears to be pc order (≡ G) order.... but not exactly. The srch is for hy G cands that fit exactly. → 262.20 spec.

0 1 So, for m=11, 1/10 off. way down from max, we get $\frac{1}{e}$ decay ≥ ½ way down e^{-5} decay.

The restriction $n >> m+d$ may not be important. ~~It holds for small d~~

m is usually << n; The d can be almost as large as n, we are mainly interested in d values near the top; i.e. $d < \frac{n-1}{m-1}$ and maybe d as large as $\left(\frac{n-1}{m-1} \times 2\right)$

If m is 11, This is $\frac{n-1}{5}$ which is perhaps smaller so that if $n-1 = 10^4$, $d = \frac{n-1}{m-1}\cdot 2 = \frac{10^4}{5}$ order

d+10 is ~ $\frac{1}{5} \times n-1$ so the approx is prob. still O.K.

At any rate, the decay rate $\boxed{e^{-\frac{(m-1)\cdot d}{n-1}}}$ for $m+d << n$ looks very useful

07: 259.05 Put population in G local bins. Take individual from level X random node. Take indiv. from Level Y, random branch — combine: keep track of G levels of combination X & Y, we can put branch of an individual at another node of same individual. When we have enough data, optimize $a + bx + cy + dx^2 + fxy + gy^2$ (6 params) as approx to $\bar{G}$. Also approximate σ_G as a function of x,y. Then find x,y combinations w. highest $\frac{\sigma_G}{G_{max} - \bar{G}}$. ($G_{max}$ is present Global G_{max} — not nec. in the "X·Y" "[illegible]" of ~~a~~ alternatives

An easy way to get σ_G: do a curve fit like .10 for $\bar{G^2}$; then $\sigma_G^2 = \bar{G^2} - \bar{G}^2$

$$\frac{\sigma_G^2}{(G_{max} - \bar{G})^2} = \frac{\sigma_G^2}{G_{mx}^2 + \bar{G}^2 - 2\bar{G}G_{mx}} = \frac{\bar{G^2} - \bar{G}^2}{G_{mx}^2 + \bar{G}^2 - 2\bar{G}G_{mx}}$$ } No great simplification other than no need to take sq. root!

$$= \frac{\bar{G^2} - \bar{G}^2}{1 - 2\bar{G} + \bar{G}^2}$$ } if one normalizes G w.r.t. G_{mx} so $G_{mx} = 1$. → $\frac{\bar{G^2} - \bar{G}^2}{(1-\bar{G})^2}$

The numerator will be a quadratic in x & y: the denom. the square of a quadratic in x & y.

Since we have strong ideas on what G & G^2 look like, It prob'ly would be good to use ANN, Hvr, since both $\bar{G}$ & $\bar{G^2}$ are very probly smoothly monotonic in x & y, it may be easy to "Maze Me"

→ ~~260.263.21~~ ~~to produce good MEJ faster~~ → 261.00

0

21 : : error in 259.22 from 259.21 $\sqrt{\frac{n-1-d}{n-m-d}}$ $n-1 \equiv N$ $m-1 \equiv t$ $\sqrt{\frac{N-d}{N-t-d}}$; $N-d \equiv S$ $\sqrt{\frac{S}{S-t}}$

$$\sqrt{\frac{n-1-d}{n-m-d}} \cdot \frac{(n-1-d)^{n-1-d}}{(n-m-d)^{n-m-d}}$$ $S \equiv n-1-d$, $t \equiv m-1$ | $n-m-d = S-t$ | → $\sqrt{\frac{S}{S-t}}$ $\frac{S^S}{S-t}$

$N = n-1$ $t = m-1$

$$\sqrt{\frac{N-d}{N-t-d}} \cdot \frac{N-d^{N-d}}{(N-t-d)^{N-t-d}} = \sqrt{\frac{1-\frac{d}{N}}{1-\frac{t+d}{N}}} \approx 1 + \frac{t}{2N}$$ ≈ const

α: $\frac{N-d^{N-d}}{(N-t-d)^{N-d}}$ · β: $(N-t-d)^{t}$

for d, t << N!

$$\left(1+\frac{t}{N}\right)^{N-d} \cdot \left(1 - \frac{d+t}{N}\right)^{t} \cdot \left(\frac{1}{NT}\right)^{const} = e^{\frac{t}{N}(N-d)} \quad e^{-\frac{t(d+t)}{N}}$$

$$= \exp\left(t - \frac{td}{N}\right) \quad \left(\frac{1-\frac{d}{N}}{1-\frac{t+d}{N}}\right)^{N-d} \cdot \left(1 - \frac{t}{N}\right)^{t} \approx \exp\left(\left(-\frac{d}{N} + \frac{t+d}{N}\right)(N-d) - \frac{t(t+d)}{N}\right)$$

const = α; $\frac{t}{N}$

$$\exp\left(\frac{t}{N}(N-d) - d\right) = \exp\left(\frac{td}{N} - d\right) = \exp\frac{(t-1)d}{N}$$

$\alpha = \exp\left(\frac{t}{N}(N-d)\right) \to \exp\left(-\frac{td}{N}\right)$ | $\beta = \exp\left(-\left(\frac{t+d}{N}\right)t\right) = \exp -\frac{dt}{N}$

35 So $\alpha \cdot \beta \to \exp\left(-\frac{2td}{N}\right) = \exp\left(-\frac{2(m-1)\cdot d}{n-1}\right)$ (= $\boxed{\exp\left(-\frac{2(m-1)\cdot d}{n-1}\right)}$ ← Previous version lacked a factor of 2 for $n >> d+m$

I checked the "factor of 2" again $\exp\left(-\frac{2td}{N}\right)$ really seems right!

Seems quite rt.

0/6/04
4TM

.0 : [illegible] tourn of [illegible] by [illegible] .

Do 258.36 ff after one has adjusted X_0 via usual MEJ. See if 258.36 increases or ↓ MEJ. If [illegible] ↑ of α makes MEJ worse, then ↓ α & see if it helps

.03 Then ↑ or ↓ X_0 & see if adjusting α can ↑ MEJ (so move X_0 & α to try to ↑ MEJ). [illegible]

(SN) Some Q's about GP! ① When 2/cands are tested for breeding, why exchange branches?

25 If a branch is good, it should be tried on all reasonable places. → 260.07

② In tournament selection! What kind of DF. is obtained for the top choice?

Say population is n individuals, all of different α = 1 to n. We select m of them at random.

What is prob that the best of the m, is distance d from top? $d = 0 / n-m$

Soln: Prob that the d^{th} place is the best of the m:

0 is: $\frac{1}{n} \times \left(\frac{n-d-1}{n} \cdot \frac{n-d-2}{n} \cdots \frac{n-d-(m-1)}{n}\right) = \frac{1}{n^m} \cdot \frac{(n-d-1)!}{(n-d-m)!} \approx \frac{1}{n^m} \left(\frac{n-d-1}{e}\right)^{n-d-1} \sqrt{\frac{n-d-1}{n-d-m}} \frac{e^{(n-d-m)}}{(n-d-m)^{n-d-m}}$

m-1 factors. $n-d-m+1$

$= \frac{1}{n^m} e^{m-1} \left| (n-d-1)^{n-d-1} \cdot (n-d-m)^{-(n-d-m)} \right| \to \left(\frac{n-d-1}{n-d-m}\right)^{n-d-1} \cdot (n-d-m)^{m-1}$

$-(n-d-1)+m-1$

More exactly

$= \frac{1}{n} \times \left(\frac{n-d-1}{n-1} \cdot \frac{n-d-1}{n-2} \cdots \frac{n-d-(m-1)}{n-(m-1)}\right) = \frac{1}{n}\left(\frac{n-1-d}{n-1} \cdot \frac{n-2-d}{n-2} \cdots \frac{n-(m-1)-d}{n-(m-1)}\right)$

$n-m+1$; $n-m+1-d$

actually, let $n-1 \equiv S$, $n-m \equiv U$: $\sqrt{\frac{S-d}{U-d}} \frac{(S-d)^{S-d}}{(U-d)^{U-d}}$

$= \boxed{\frac{(n-1-d)!}{(n-m-d)!} \cdot \frac{(n-m)!}{n!}} = e \cdot \sqrt{\frac{(n-1-d)\cdot(n-m)}{(n-m-d)\,n}} \cdot \frac{(n-1-d)^{n-1-d}}{(n-m-d)^{n-m-d}} \frac{(n-m)^{n-m}}{n^n}$

"constant" wrt d

.0 regard n & m as constants: so just

21 $\sqrt{\frac{n-1-d}{n-m-d}} \cdot \frac{(n-1-d)^{n-1-d}}{(n-m-d)^{n-m-d}}$. Say $m \ll n$

$n-d \equiv r$ | $S \equiv r-1 = n-d-1$; $r-m = r-1+1-m = S+1-m$; $m-1 \equiv t$

$S \equiv n-1-d$; $S \equiv n-d-1$; $t \equiv m-1$; $n-1 \equiv N$; $S = N-d$

-22 → $\sqrt{\frac{r-1}{r-m}} \frac{(r-1)^{r-1}}{(r-m)^{r-m}} = \sqrt{\frac{S}{S+1-m}} \frac{S^S}{(S+1-m)^{S+1-m}}$

Wrong! see 260.21

$= \sqrt{\frac{S}{S-t}} \left| \frac{S^S}{(S-t)^{S-t}} \right. \frac{S^S}{(S-t)^S}(S-t)^t = \left(\frac{S}{S-t}\right)^{S+\frac{1}{2}} \cdot (S-t)^t$ includes. So fix t & vary S. $S = n-1$

→ Error! see 260.21

$\left(\frac{S}{S-t}\right)^{S+\frac{1}{2}} \frac{(S-t)^t}{S^t} S^t$ Say $S \gg T$ $\left(\frac{1}{1-\frac{t}{S}}\right)^{S+\frac{1}{2}} \left(\frac{S-t}{S}\right)^t \cdot S^t = e^{\frac{t}{S}(S+\frac{1}{2})} \cdot e^{\frac{t^2}{S}} \cdot S^t$

$e^{t+\frac{t}{2S}+\frac{t^2}{S}} \cdot S^t$.

)0 $= e^t (eS)^T \cdot e^{\frac{t}{2S}} \cdot e^{\frac{t^2}{S}}$. | $t = m-1$ & fixed.

$n-1 \equiv N$

main factor: $e^T \cdot (N-d)^T$ so look at $(N-d)^T = \left(\frac{N-d}{N}\right)^T \cdot (N^T)$ constant

$e^{-\frac{d}{N}T}$ $\propto e^{-\frac{d(m-1)}{n-1}}$ for $S \gg T$ i.e. $n-1-d \gg m-1$; $\boxed{n-d \gg m}$ — $\boxed{n \gg m+d}$ | tho' $m+d \ll n$, md can be $\gg n$.

for $n \gg m+d$ → exponent should be twice as big! — see correction 260.21 .35

Say $n = 10^4$, md can be 10^3

Should be $e^{-2d(m-1)/(n-1)}$ see 260.21

$= \left(e^{\frac{m-1}{n-1}}\right)^{-d}$ Which is not a very sharp decline, since $\frac{m-1}{n-1} \ll 1$.

On the other hand $\frac{d}{n-1} \approx \frac{d}{n}$, the fraction of the way down. So!

$= \left(e^{-\frac{d}{n-1}}\right)^{m-1}$ The $\frac{1}{e}$ distance for d is $\frac{n-1}{m-1}$ Say $\frac{d}{n-1}$ is a fraction between 0 & 1

if $\frac{d}{n-1} = \frac{1}{m-1}$ we have $\frac{1}{e}$ decay So $\frac{1}{m-1}$ gives the fraction from the top for which we get $\frac{1}{e}$ decay.

0/5/04

4TM

.02 : It may be that Phase 1 (Ph1) doesn't have to be very smart in order to go to Phase II (Ph2): It could be slow in Ph2.

So T. general plan is to use ≈ GP (+ PPM) to lrn. a TSQ (probly Algebra). Some diffctys are

(1) Getting GP to do a TSQ: (2 recent Batches of notes on this) → 252.10 — .40 ; 253.08 — 11 — Also 253.12 — .21, .22 — .40 is good; So 253.08 — .40; So ≈ 25 2.10 — 253.40 ≈ 2 pp.

(2) Using PPM and GA(b) for search. (Not so much of an unsolvd problem).

A simplifn. But may be — 0.4: 253.27 (No explicit use of R (recognition) function).

.06 So: General plan: Quick once over!

1) Design. GP system that can work ALN & similar problems. After each problem is solvd, we save all (or almost all; random save) of population: we may only save "top 10%" or whatever.

2) After we have population, we implement ≈ PPM on entire population. & See if it works.

3) Next do $x_0 = \frac{\sigma_G}{G_{Max} - G}$ ← ≡ GP(b). See if it ↑ rate of G ↑. (≡ G′)

4) Expand TSQ: Try harder Alg. problems: Try probs ≠ Algebra. See if we can get it to invent ≈ "R": (??) or perhaps Give it "indexes" so it can categorize easily, then have it learn to do categorizn. (or use A. N. Compression method for categorizn?).

_in (5) : I have to get a PPM working first. Try a regular BZ2, but break it down so I can get pc's directly. Make Modules in direction of My own version of PPM. See if it's better. Perhaps try variants on Eng text, on Lisp codes, on / short Lisp codes from solns of GP problems.

Expand outline of .06 ff, looking for diffctys; look up relevant refs in my [redacted] notes & other papers.

5) Advance TSQ to Ph2 (Phase) type induction problems.

6) Phase 2.

SN: There is a version of B-trees that's designed for large Data Bases using HDD; Could be used in GP(b)?

SN: in GP(b), If we can't get G to ↑, try reducing x_0 (a fair amt). Arr., this might be done Automatically by T. System!? So as to ↑ $\frac{\sigma_G}{G_{Max} - G}$ (.21R) (.23)

.22

.23 : (.21R) → Another, perhaps better, way to get similar effect: Arrange to have dist. over G values of entire population. Say x window, w. adjustable width: width arranged for max $\frac{\sigma_G}{G_{Max} - G}$. [Note that: But if we use "x window" & G_{Max}, G should be "linearized". ≡ Expected ΔG ≡ EDG — Exp. ave. ΔG. MEG (Max EG); MXJ; Max Expected Jump (MEJ) — MEJ Max

In T. case of induction, I don't know if pc is linear — ~~lin PC (e.g. induction problems)~~ or maybe ln pc? — or neither. For small ↑ in G, hvr, linearizn is not (so?) important,

.4 (MEJ) → Since non-linear is "linear" for small jumps.]

There's some Q on t. utility of t. targ: After all, all of t. S-lays used have universal outputs so any (desirable) code has pc > 0. Hvr, If we could legitimately get a big MEJ by modifying t. S-lang in this way — Real Fine! T. form of .23 doesn't look like a "fine pt". For a Boroughs Wheeler Lang distribn; ~~..~~ in Computing pc's w/o generating M. carlo trials: Look at G of a cand. in population. A priori reject it w. probly ∝ exp(−G/λ) so large G gives small prob of rejection. To save time, do several trials for each accepted G value.

.36 A Different possl. way to ↑ variances: For S-lays based on ≈ "Laps rule": In computing numerator, denom. of probys: Lsp will give a certain constant for ≤ SZ = 0. By increasing these constants, we flatten t. distribn. We can ↑ constant in num, denom by same constant factor. The effect is minimal when ~~...~~ ≤ SZ is large. A way to do it for even large SSZ: ↑ (constant in) denom by factor α so that it's = to rest of denom. Then ↑ constant

4TM

256.40

256.40: It's not clear as to whether to always use Rissanen combined conc. for predn.

Also, I'm not quite sure that Cover's analysis applies to my analysis of PPM & its variants (like use of sub-trees as contexts).

SN There may have been 2 versions of PPM analysis: one in which we use "context" as in PPM: in the other, we just make delimitations of ngrams which are supposed to include & to have to be predicted. This last is more like Z14|.

6 → One big difference is Rissanen PPM, m, i.e. no. of distinct α codes, & no. of contexts, s is very large.

27 Also Ris m & w. corpus length. — How is main difference is that I don't really do all those in codecs!!!

For a particular predn, say I consider 10 contexts. I only consider those α & α codes for that predn. Now, there are an enormous no. of other contexts that did not occur" — they are the α_i — parts of various codes: There is the enormous no. of α_i codes that I don't consider.

— If I did, they might have very large wt. w.r.t. the 10 "positive" contexts that I do consider.

Those α_i codes perhaps average to a p.d. $\approx$ the "default code" & low freq. of occurrence of various tokens.

Say one of the possible codes had a wt. of 100 w.r.t. the "default code", if there were 1000 α_i type codes, just $\gg 100$ then we end up w. something close to the default code for the entire corpus. (No real compression!)

So, if we did it exactly right w. many α codes, the Cover analysis would be correct, and we would end up w. very little compression.

The way I had in mind — i.e. .07-.09 isn't directly analysed by Cover's analysis. How Good my method is, is an empirical Q.

2 254.05 also 254.06-.20 → SN DON'T FORGET DNA Analysis techniques. Looks like they can be useful for recognizing regs of the type (255.00-.04) in which a .... cfg occurs several times on the corpus w/ different ngrams btwn "a" & "c". This is a very impt. kind of regy in RPN terms of functions. Could this fact be relevant to its importance in DNA analysis?

SN Re: Regs of type (255.00-.04): If functions are created so that their first arg. forms a useful ngram w. the function name, then we have a useful regy. Perhaps any randomish method of GA such would usually end up w. Mostly functions of this kind, because they were useful.

On the other hand, if we have a binary, non-commutative function, we simply don't have the option of using a different order of arg. commonly. One way to fix this is say w. "SubFact" function (non-commut.: $= a-b$) So a, b can be converted to sum a, neg b = sum neg b, a. We can do similar thing w. Division (non-commutative). However, I'm not entirely sure this will work! The others seem to be to have all functs be commutative. So we need neg & recip. unary functions to implement subtraction & division.

0/4/04

4 TM

.0: 255.40 : In Coverzudking See pp 255⅓, 255⅔. This is disturbing, because up to now, I considered the wts. that I was using to be optimum: They are for large corpus about equiv to using t. single best PEM. — and a constant mixture value may do better than best single PEM!
I don't have a "coding argt." at all to decide what to do. 255⅔.07 & following suggests/tells how to do ∴ "constant" wts.

It looks a lot like Multinomial prediction of Time Series! Maybe there is a matrix inversion method to get good wts!

We have a seq. of vectors, $\vec{x}_1, \vec{x}_2 \ldots \vec{x}_k$ m components, giving t. pc's of t. event that did occur, as viewed by t. m PEMs.

We want a single normalized vector $\vec{V}$ ∋ $\prod_{i=1}^{n} \vec{V}\cdot\vec{X}_i$ is max

$$\frac{\partial}{\partial z}\prod_i f_i(z) = \left(\sum \frac{\partial f_i(z)}{\partial z}\frac{1}{f_i(z)}\right)\prod f_i(z)$$ ⇐ is this true? — proof by simple induction.

$$\frac{d}{dx} f_1\cdot(f_2\cdots f_n) = f_1' f_2\cdots f_n + f_1 (f_2\cdots f_n)'$$
$$f_1(f_2' f_3\cdots f_n + f_2 (f_3\cdot f_n)')$$
$$\left(\frac{f_1'}{f_1}+\frac{f_2'}{f_2}\right) f_1 f_2 = f_1' f_2 + f_2' f_1$$

Say $\frac{d}{dx}\prod_1^n f_i = \sum_1^n \frac{f_i'}{f_i}(\prod f_i)$; $d(\prod f_i \cdot f_{n+1}) = \left(\frac{d}{dx}\prod_1^n f_i\right)\cdot f_{n+1} + f_{n+1}'\prod_{i=1}^n f_i$

$$= \sum_{i=1}^n \frac{f_i'}{f_i}\prod_{i=1}^n f_i \cdot f_{n+1} + \frac{f_{n+1}'}{f_{n+1}}\prod_1^{n+1} f_i$$
$$= \sum_{i=1}^n \frac{f_i'}{f_i}\prod_{i=1}^{n+1} f_i \Big\} = \sum_{i=1}^{n+1}\frac{f_i'}{f_i}\prod_{i=1}^{n} f_i$$

So its true.

Back to .09;

19 $$\frac{\partial(.09)}{\partial V_j} = \sum_{i=1}^{n}\frac{X_i^j}{\vec{V}\cdot\vec{X}_i}\left(\prod_{i=1}^{n}\vec{V}\cdot\vec{X}_i\right) \;\; \text{'' } \pi$$

0 $$\sum_i \frac{X_i^j}{\vec{V}\cdot\vec{X}_i}(\pi) = 0$$

Likely $j=1|m$ → $\sum_i \frac{X_i^j}{\vec{V}\cdot\vec{X}_i} = \phi$

trouble: all $X_j \geq 0$; all V components ≥ 0. unless we are at edges!

Well, w. Lag's mults, we have a bit more: $\sum V^j = 1$

$$\sum_i \frac{X_i^j}{\vec{V}\cdot\vec{X}_i}(\pi) - \lambda = 0 \quad \text{and}$$

$$\sum_i \frac{X_i^j}{\vec{V}\cdot\vec{X}_i} = \frac{\lambda}{\pi} = \text{constant}.$$

I don't see how to solve this → : (255⅔.14 — .40) looks better!

8: (255⅔.28) ⇒ For m params do I need to solve for ~ $\frac{m^2}{2}$ matrix elements (a covar matrix)?
In t. usual n.linear optzn techniques, they may assume all off diag values = φ!
Hvr, they may be good enuf to get pooky by successive approx.

0 To approximate π ∝ (.09) as a function of $\vec{V}$, need constant & linear as well as 2nd order terms,
I think the method was simply usual const+linear+Quadratic terms: for m args = $1+m+\frac{m(m+1)}{2}$ coefs
Somehow, t. optzn method usually works if we de-fuse crossproducts! — surprising! — Both "Math Recipes" and "Gershenfeld" may discuss this.

Anyway, I think t. Big Point is that my old method of combining 11 codes was usually not so good ⅓ — that it usually ended up w. t. "Best" code having almost all wt.

Say I'm using various prediction ahss. (= "contexts"): I have a test s'm ahss. Each one gives a diff pd. over t. next token possy. Other than give = (or near = wt), I could invent a new ahs. that is coo'covariance of them & use its p.d. on t. past as t. pd. for t. next token. One problem is ssz.

.40 : Each of the P.E.M's will have about same entropy, yet any mix up of them will reduce noise.

I. larger no. of them we mix, the less noise so → "true" pc.

Hvr. in K & C, the individual PEMs had rather different entropies — yet mixing did ↓ entropy to < that of "Best Guesser"!

.4 A case where simple mixing certainly ↑ entropy: One of PEMs is exactly ~~correct~~ pc. All other PEMs to be mixed are uniform distributions over the alphabet. So mixing simply "flattens" the correct p.f. & moves it away from correctness.

.7 Consider this method of wtng: $a_n^{(i)}$ = wt. of i^{th} PEM at time n : $a_n^{(i)}$ is adjusted so that if it were at that (constant) value most of past, the resultant pc of ~~[illegible]~~ corpus would be max. Hvr, finding $\vec{a}_n$ ($\equiv (a_n^1, a_n^2 \cdots a_n^m)$) seems very diff'lt. We can make rapid approxns by sampling the corpus. Small samples, then slowly ↑ ~~[illegible]~~ size as we slowly zero into a good $\vec{a}_n$ function. The closer we get to max, the larger size we use, & the slower we are to make changes. Usually, our sample will be most recent data. For m PEMs, we are looking for a Max. in m dim space. (.31) A good way to find this is the Normal Non-linear opt'n method... possibly using [illegible] & gradient — also sampling of each pc. of PEMs (.29)

.14 For a first Approxn, use geom./mean pc per symbol of each PEM as its rel wt.

Proof King Cover Gary of 12 Guessers: pc ratio varied from ~~[illegible]~~

.6 .273 to .149 — a factor of about 2. — But a factor of $2^{.61} = 1.52$ (13R)
2.445 to 3.732 or $\frac{1}{2.445}$ to $\frac{1}{3.732}$ (seems more reasonable)
.409 .267 for Approxn (.14)

.14 ~~[illegible]~~ In the ~~[illegible]~~ example of .04–.06, the correct ~~[illegible]~~ PEM gives a certain pc/symbol which is not = 1. The others will give less pc/symbol, so they will get smaller — so .14 will not be exactly right. Hvr, for a large corpus, it will be found that giving the correct PEM most of the wt. will give ~~[illegible]~~ max pc of corpus.

.2 Hvr. the method of .07–.13 doesn't tell how to deal w/ a very large set of PEMs.

.3

.4 (13R) Practical compn. of $\vec{a}_n$: It's easy to do a run of maybe ~100 randomly sampled $\vec{a}_n$ points for pc of the entire corpus. Then pass an optimum quadratic form thru these points. Use the best-fit form to get a good guess for $\vec{a}_n$; then distribute a bunch of random trials about that pt. & pass a new quad form thru that data & get its max, etc.

.28 I've forgotten the details of .24 !! → 256.28

Objection of .22 : It doesn't yet fit into AIP.

.31 (.15) Another form for .07–.13 : Instead of using entire corpus to optimize $\vec{a}_n$, use a window on corpus: The length of window also must be optimized. Using random sampling of ~~[illegible]~~ corpus is much harder, hvr. — if at all possible!

→ 4TM 256.00

1.20.04

KING and COVER : 1978

$255\frac{1}{3}$

Comments on King-Cover paper: IEEE-IT trans. July 1978 p 413.

.00 XIII. (1) Conclusions: p 439 how they got 64% redundancy is unclear.

$\log_2 27 = 4.755$ bits/char. 1.25 bits/char is $\left(1-\frac{1.25}{4.755}\right) = 1 - .262 = .737$ = 73.7% compression

— 64% compression: $.36 \times 4.755 = 1.71$ bits/char. Only 2 out of 12 subjects got worse than that!

→ (2) P 418 II last ¶! letting $\alpha_n^{(i)} = \frac{1}{m}$ for committee; means using mean of their b estimates

.04 for each ~~box~~ bet. That it should end up Better than "Best subject" is surprising! Perhaps

.05 Cover (or his co-author) have shown it to be usually/often/always true! → (3), also (.26)

(3) If "Jefferson & Virginian" was available on line, I could run BZZ on it, using my lower case. — In both test texts, there was only 1 sentence to be predicted, so that the period & double space at end of a sentence do not occur.

Or, get it from library & scan in a few pp! → (5)

(4) from (2): Finding the best $\alpha_n^{(i)}$ ($i=1/m$ ≢ $\alpha_n^{(i)}$ indep of n) would be easy (I think — this is ≈ what I did in my "optimum portfolio") & got a v.g. yield. Was Cover's univ. Gambling scheme averaging over all $\alpha^{(i)}$ values ≥ 0 (again same for all n)? — Anyway, if $\alpha^{(i)} = \frac{1}{m}$ did v.g., a guessing it to be optimum would do Better! — i.e. Better than 1.25 bits/symbol.

from (3) (5) try doing ~~[illegible]~~ given ~~text~~ from "Jeff & Virg" on ~~BZZ~~ directly, see how much compression it gets! I need to make a file w.o. caps or CR&LF's. Perhaps create one via Basic in some file area, so I can load it into BZZ. Perhaps Delay doing this until I have a version of "BZZ" that I can control better! ⟶ This may be available online — try Google. Doesn't seem to be — Markov & Cover.

For first letter of a word, use freqs of first letters of words for context "space" — i.e. don't use uniform default distribution. For second letter, use digrams freqs for starts of words (The BZZ may do this normally)

(6) On second thought, I shouldn't expect ~~BZZ~~ BZZ to do much good w. such a short corpus. It wouldn't know ngm freqs. Using the entire book as corpus would be useful, hvr. A human wouldn't need such "external" info, because he has experience/memory of general English text from R.W. Perhaps start out w. large English text from Calgary corpus, or that enormous English text that B&W use.

26 (.05) 0.4.04 (7) Hvr., the way I would do it might still be better than "committee": my method is to redistribute each $b(x_k^i)$ = $\sum_{i=1}^{m} b_i(x_k^i)\cdot a_k^i \Big/ \sum_{i=1}^{m} a_k^i$

a_k^i = product of probs assigned to correct symbols up to k^{th} by i^{th} estimator

I Risk this is not the same as giving $\frac{1}{m}$ to each estimator & letting them bet endlessly & then summing, — which is "average Capital" method. — The, for a long corpus it usually approaches betting all on best estimator, since "best guesser" gets almost all wts for best later in the corpus.

As is, total capital for committee was ~~[illegible]~~ $\frac{12.4}{1.5} = 8.26$ times that for "Best guesser"

"Average Capital" method was $= \frac{\text{Best Guesser}}{\cancel{}11.7} \approx \frac{\text{Best Guesser}}{12(=M)}$

[Note p 418 (of King/Cov): Column II from Corpus middle to bottom — starts on cth bottom eq(28): they do consider "(My)" method. They show that it → Best Guesser as n→∞, which is correct. Hvr, he seems to prove that the yield is identical to that of "a" — I can't see anything wrong w. the proof!

It is disturbing that one can do better than "Best Guesser" — an example: Say the true pc's at ~~[illegible]~~ time K is $\tilde{p}_K^o(i)$ ← $0 \leq i \leq D$ D = radix, we can have M different PEMs w. $p_K^n(i) = \tilde{p}_K^o(i)$ + noise; then $p_K^o(i)$ = normalized $\tilde{p}_K^n$. → 200

0/3/04

4TM

: **In a PPM analys of RPN functions:** ~~Easily~~ consider $\underline{a}(b(d,e), \underline{c(f,g)})$ (at 254.08):
c fg will be regarded as "following" b(d,e); rather than being "one of the args of a".
on the other hand b(d,e) is regarded as following "a" — which is a likely more relevant [T. function name]
Can I find a way to (store, annotate, ..) the corpus so that both of these kinds of regys
.04 Would be recognized?

A possy: to include ~~the~~ the corpus coded w. order of each function's args (reversed/permuted).
[I did consider this as one of 4 simultaneous models of the corpus ~~for~~ for PPM]

.07 A slightly different tack: Regard all unidirectional sequences ~~from~~ thru a conditional tree as legit. context strings:

So a b e j is such a string
a c g n " " "
a b d h " " "

Hwr, if we ~~want~~ want to retain all such strings; [for each leaf there is one such string]: So the no. of nodes in our corpus is [(no. of leaves) × d] the depth of corpus. ~~Normally~~ Normally the no of nodes in our corpus ~~is~~ (for binary branching only) is only [(no. of leaves) × 2] ← Not a gross ↑ in needed Memory

---

.15 [In generating a string (root / top down) we ask: in view of the previous vertical context (.07–.14)
.16 What is the d.f. for the (left token / rt. token) of this context? After we decide on left token, we may
.17 or may not use the left token as partial context for d.f. on rite token. Would the corpus of substrings of (.07–.14) be enuf to do .15–.16? (Not including "lateral" context, perhaps) [of (.16–.17)].

Part of realizing .07–.14: each node (other than top node) ~~has~~ has auxillary label of
.20 "left" or "rite". This enables us to get d.f.'s ~~for~~ for each context for left & rt. arguments.

.22 **So: A list of** PPM-type regularitys: ways to compress:

.23 1) Normal PPM regys: (perhaps using my possibly more exact parallel codes: See 227.00–.01 for refs of dcuns. of these methods & 227.00–.40 for discussn. of their Legitimacy.

2) PPM using subtrees as contexts — (see refs of .23 for dcuns, expressns) — 260 Note (4) (.28)

3) For PPM of function trees (Lisp or AZ): Use RPN than normal PPM: Also use PN & Normal PPM.
A cording to analysis of 254.06–.20; This is equiv. to permuting order of args & not changing from RPN to PN.

.28 4) As a perhaps cheaper way to do 2): do (.07–.20)'

5) a) Use Z141 & mono-delections:
b) " " & expand it to do vgensts & eventually CFG's & CSG's.

.30 6) Try any of the ~~above~~ going Backward. (This is not the same as reversing order of all arguments).

A BIG Q is how much time/memory it takes to do these predictions/compressions.

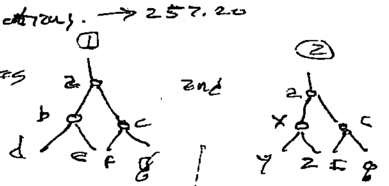
9/3/04

ATM

.0 [SN] T. kinds of regys detected for DNA analyses (matching of substrings w. Gaps) ~~Seems~~ t. same as t. kind of regys found by common subtrees..(≡ common sub-functions if one uses Polish or RPN)! Also note that Pgms for DNA analyses are trying to get SPEED! — So they may be what I'm looking for.

Also, Porg suggests that DNA may be better understood as a notation
25 that denotes a composition of sub-functions. → 257.20 .05R

06 T. idea of .00ff: consider subtrees ① and ②



08 ①.13 $a(b(d,e),c(f,g))$

= a b d e c f g. Polish notation ≡ PN

0 ② = $a(x(y,z)),c(f,g))$

= a x y z c f g

So if we use PN, we have a —— c f g matching.

RPN ①

If we used R PN: $((d,e)b),((f,g)c))a$

= d e b f g c a

5 RPN ② $(((y,z)x),(f,g)c)a$

y z x f g c a

.15R

So in this case RPN gives t. similarity by simple suffix matching (PPN)

In any long function tree, if we suitably switch between both PN & RPN,

0 — We could detect any subtree as a pure suffix.

→ Is it t. same as switching order of arguments in functions? Yes exactly t. same.

So [N.B.] keeping track of Corpus in both RPN & PN does double storage cost, but it also does ↑ no. of regys found. — T. problem is — does it ↑ rate of entropy ↓ to justify ↑ in cc of storage & execution of both PN & RPN? → .36

N.B. For my purposes RPN = PN with order of arguments reversed. That's it!

[SN] In Ga(b), we generate M. Carlo trials ~~on~~ from a corpus ~~of~~ cands that have their empirical G > x_0. Could we generate M. Carlo trials that

.3 would recognize subtrees? (.31)

[SN] My Paris Paper "A New method ..." ≈ 1960 (or 1959)!

Instead of a "teacher", ~~try~~ ~~the~~ look for candidate regularity in t. ~~[illegible]~~ entire corpus: If it compresses t. corpus, it is a cand rule to try.

Hvr, rules "discovered" this way must be arranged so one can occasionally "backtrack" (≡ undefined them) on them, since they will occasionally be

8 ~~[illegible]~~ ~~prove~~ globally "non-optimum".

1: (.23)↗ It would be instructive to compare various methods of generating M. Carlo trials from a corpus.

Mutations, Crossovers of various kinds would be ~~[illegible]~~ subclasses of the methods.

Strictly speaking, (cross) that is not ~~really~~ exactly a sample of (trial) one, but both are ≈ true!

It would seem to be easier to make such an analysis ~~if~~ if Lisp w. RPN is used, than if other languages & notations are used.

36 (.15R)↗ It may be that as we get closer to ~~[illegible]~~ goal ordinary RPN, say will begin to "flatten out" & at this point, it becomes useful to start using both notations of corpus. Analyses leading to / copy of 124.33 is relevant (probably).

Spec.
.00:252.40 dream of ~~a~~ OZ problems and learns to correlate them w. contexts (or other things that help generate cands) in t. cands that have successfully solved those problems. 235 rev.

[SN] Use **Top of page More!** If there are an new impt ideas, or impt. continuus of old ones: write note on top of page! Give back refs. if relevant. This helps much in reviewing!

.01 [SN] **SM:** Use External Contexts of stock characteristics (industry, earnings ect) to help group stocks for likely cross correlns. Otherwise, if one uses very large no. of candidate "indicators", one will "overfit". So this External Context idea enables one to use a much larger set of cand. indicators.

08 [SN] Random Notes on Review of 250.11 ff:
I think recent ideas centered on dify of Getting GA to do TSQ's: T. trouble was, in "pure" GA, I'd like to
include t. hy G population of all previous problem so far. ① As t. TSQ gets larger, This
11 population Gets too large. ② Hvr, another problem is that as TSQ grows, we cannot have (.22)

2 (SN₂) A possl way to deal w. this: For t. entire population of cands, there will "be
3 an optimum x_0 ∋ $\frac{\sigma_G}{G_{max}\bar{G}}$ = "max". As the population ↑ G_{max} ↑ a little, but $\bar{G}$ ↑ only a little &
σ_G may be about constant. So for each problem, its population of cands w. $G > x_0$ will be
about constant & t. total population of cands will be ∝ no. of probs in TSQ.
To deal w. this; store on Disc, random samples of cands from each problem
w. $G > x_0$. One can have, say 100 sets of cands for t. entire TSQ on t. Disc.
Between problems, one loads a random population from disc into RAM — so
3 each problem one has a (usually default) population for induction.
21 [Re. 07.13 See 239.07 for defns of x_0, G_{max}, G, $\bar{G}$]

This may be solved by sampling.
22 (.11) of time to test each cand. on all previous problems. ③ A third problem is: Using t. entire
past corpus w. = wts. to solve t. present problem seems to throw away much impt info
we would like to include contextual info, that related "~" problems in past.
This last can be solved in a couple of ways ⓐ for QA problems, include external context
(Math problem, diff eqs., occurs in chemistry, Given to me by Prof Z ... ect)
27 also recency context, ... ⓑ Don't solve this problem: Let TM discover "O" functions
that examine each problem & decide what concepts to emphasize
in solving it. Want to try to find ways to encourage TM to find solns. of this type, by including certain "primitives" in it, or by other tricks
ⓒ Solve it this way! TM looks at problem of past & solns to them,
0 Tries to correlate "obs" "features" of problems w. concs (include primitives & defined & sequences (= contexts)
of primitives). This could be done by "Entropy analysis" that Li-Vit used to
discover "categories" in Music, Plants, Literature. A more General soln. : Given Q_i
we want A_i ∋ t. entropy of $Q_i A_i$ is min. Hvr, we may want to use as corpus, t.
set $[Q_j A_j]_{j=1}^{n} = S$; Then we want A_i ∋ t. entropy of $S \cup Q_i \cup A_i$ is max or;
more exactly, we want t. entropy distrib of $S \cup Q_i \cup A$ over all possl A_i — tho we are
only interested in A of low entropy.

Hvr, if we are considering Q_i as a (partial) context for A_i, we must use a more
general context that PPM (or even more subtle suffix structured contexts). All of Q_i is t. context &
any abstraction on it is a defn of context.

4TM

20 (spec 20 / 250.10 / 250.129); Another trouble w. 250.11 ff: Its for INVs (Formal Induction), not OE. This may be O.K. — i.e. we don't need to do OE until Phase II. The GA-like systems do OE problems well, They are not able to learn from previous OE prob. solns.

2: 250.10 ff was a particular method of dealing w. the difficulty of Testing all new candidates. Entire Corpus of problems thus far. Another Method was the "R" system of IDS:A §1.2 p.10 ff. 247.12 -.18; 247.00 ff discusses this: Also "Soft" R functions: a strategical testing — related One application of this idea is 250.10

14 Another Approach would be to use the updated ideas of IDS:A §1.2, with a view as it is, or to generalize Parameters to update rules to be used for "soft" (= probabilistic) R functions.

0: Another approach to "improving GA" was modeling the relation of a candidate to its Score (= Fitness) One way used a S-Grammar to try to approximate Score or γ(Score) for some monotonic γ. However. The criterion of 239.07 (max $\frac{G_\alpha}{G_{max} - \bar{G}}$) is a better Utility Criterion for a grammar. ("optimum" as just "OK") upbound

(Superficially, it seems to not use γ(Score) from the criterion, but it does. Both G_α, $\bar{G}$ are the proper things to depend on the linearization of G. After an Grammar model was formed, Monte Carlo trials were made, & if it was found to have merit. Good candidates, $\frac{G_\alpha}{G_{max} - \bar{G}}$ will change a/o we have to find a new model in view of new data. (loop to α) until satisfied. T. Grammar has to be such form that enables Monte Carlo trials

.20 is a kind of OE solver in the spirit of GA. It doesn't seem to be v.g. at TSCQ (true, however. Its not clear how to use it for a VLSI TSCQ or QA problem, because we would have to test each candidate on entire corpus to get $\bar{G}$ values that were comparable for the different candidates.

31 A method that's a bit like .20 ff is 234.19 ff, 236.21 -.402, 239.00 -.14 This uses ≈ PPM to model the population and the G values. We end up using the Criterion $\frac{G_\alpha}{G_{max} - \bar{G}}$ (239.07) But it's easier to compute than (.20 - .30) & the grammar is/can be much easier to find! (T. Grammar of .20-.30 is potentially much better: it can be S-CFG or even S-CSG (sensitive)).

SN T. main reason GA is poor for OE problems, is that we it needs a way to look at a problem & decide, from its past experience, what a good S-Grammar would be for generating candidates. This decision could be made by a function that is created by "standard" GA methods or by any of the methods of 250.11, 252.10, .20, .31. One possible technique looks at "features" in the

4TM

0 :250.40 spec: we can get over randomness effect by specifying a real# in t. interval [0,1]. T. p.c. of the specification depends on t. precision we need, but ordinarily, we will consider all values of X in that interval as giving rise to 11 codes for t. "A" being derbd (via "QA.TM").

→ T. fancy. way solves a very general Gaussian Probability problem! Say we always start w. X being uniform on [0,1]. We can use any function we like to map 0,1 into any interval, & get correct sundry probabilities. T. pc of the function (→ 1 x from) will depend on t. function — i.e. (how freely it was used on the pass). I.e. pc of its depth, in terms of nested other functions. In particular,

07 if we need a mapping onto the integers 0, 1 ... ∞, we can use this mapping to get it.

0 — [This seems to clarify a lot of my thinking about ... positive integers or a d.f. on reals from −∞ to +∞. It does mean that t. d.f. is pretty ruby — a dependent on t. pc of deriving t. function — which in turn, depends on t. freq. of use of that

3 · fancy (or on t. freq. of use of the "components of t. function), in the past.

14 **But**: What about **Discrete** P.D.'s? We could have a bunch of random bits (or Bytes?) as part of a function deriv, that would give a probabilistic mapping. However, we would like it to be possible, to go from a finite deriv of deterministic params to a distribution on an ∞ of possible outputs. — presumably, an infinite source of random bits is needed. (This is perhaps = to "R" input to t. (celebrated ∃ input vine).

One way is to have a p.d. over the positive integers (see .07-.08), which gives a pc for every finite binary string. T. mappings from now on usually are not critical: for integer X, we can use.

0 — Restricted $A2^{\wedge}(\log_2 X + \log_2\log_2 X + \cdots)$ or $\frac{A^{\alpha}}{X^{\alpha}}$ ($\alpha > 1$) or normed $\frac{1}{X_i}$ for a finite no. of X_i's. (normed)

would this really be possible?

The binary string X can be one of t. "primitive inputs" of t. A2 function (≈ LISP).

We could have >1 "X" type inputs.

Having many such R funcs doesn't cause trouble: we can just use a few (by pc) bits from each

25 of them if necy.

A method (apparently "Not Bad") method of Gaffey discombo PD's in A2: T. s-function is derived by specifying certain parts of its function tree & leaving others probabilistic. Unfortly this gives only a finite no of pdfs, — not all pdfs

29 A Gauss (compliment) of .29: That t. methods derbd in 250.30 to 251.30 do not enable all concievable

0 — outputs: i.e. they give pc = 0 to (most outputs (some & usually). — Clearly a Non-Universal ("") point of view"!

32 → One possible direction: S-funcs have no terminus/non symbol to start off! They specify (Looks promising!) part of t. s-funcs T. read off t. bits (including a terminating symbol) — it's ∈ R input.

[In t. Discussion of .14 - .25 t. input variables can be of variable length, so they could, in principle, be used to derb. any thing.

35 250.30 ff → 251.35 give some new ish ideas on how to implement (≈ universal) S-funcs. When I actually Need to use s-funcs I will probably have ideas as to what they should look like & looking at t. fancy. will suggest ways to implement them.

9/26/04 Review
4TM.

PD's on [S-FuncB] 250.30 – 251.40

[Review] of Approaches to TM: .11 – .29

0:248.14: "Working new problems" — depends on what the problem is: INV, OZ or prediction (= recovery). INV problems can be solved by Lsrch. Predn problems simply use the current updated pd for new predns: OZ problems. — well, these are the many problems! After the GPD has been updated:

Tho, TM could start out w. QA problems only. — Then not try OZ probs until Phase II.

0:248.12 → Do this now, w. Enuf details so I wouldn't forget: Then Do IRS!

11 1) ≈ IDSIA: Phase I; "Problem Pool" (248.21–249.10): In "steady state" we have a bunch of solns. to the QA, TSQ of IDSIA report. That TSQ idea is not the same as the "Problem Pool" idea.

3 Anyway when a new problem occurs, we generate the "most likely" bunch of cands from the corpus, (using ≈ PPM to get a P.D. ∋ ∵ Lsrch to get "most likely" cands. The predn is a wtd Σ of these cand's predns d.f.s. We save those "most likely cands" ∵ use them for their updating.

7 To update: we first [struck] the D.f. w.r.t. the entire corpus of "successful cands" to generate cands. We test them on the new / known QA (to be updated) only. When we find a bunch of them, we put these cands in the corpus ∵ generate new cands [Monte Carlo] (w. some [struck] on the newly inserted cands), in attempt to get a cand that will work with entire corpus. — things for MTM (≡ Σ-induction)

0 When we find one (∵ preferably more), update is done. (→ New problem → .13)

21 The .13–.21 is not same as "Problem Pool" approach .11: we might use Problem Pool at the beginning of TM's [struck] education, to get a large enuf no. of good solns in our corpus, so we can do .13 – .21

One poss. trouble w. .11 ff is that we will be doing "statistical testing" of cands, so we are normally not very certain that our best cand. really works w. all problems to date. This may screw up "MTM Lrng", — But may be O.K. for most NMTM lrng (≡ S. functions).

I'm not sure about best Update Algm, but .17 ff has some good ideas on how to do it: By finding some cands that solve the new problem (of course) — putting them into "update corpus", then Monte Carlo srch — then adding to update corpus, etc.

29 → (252.00)

30 [SN] [struck] [P.D's on functions]: [struck] PPM (extended, by tree concepts ∵ other [struck] extensions of the "context" idea) gives a P.D. on functions. If the primitives included some probabilistic functions, the output could be a d.f. on probabilistic functions. To what extent can (extended) PPM be used to model a Universal function? — (Both d ∵ S [struck] universal functs).

Well the forge is a version of the AZ language which can be made universal.

→ [ALSO NOTE!] We can probably get universal S functions by including [struck] one (or more) S functs as primitives: e.g. say one of the "primitives" is the uniform D.f. on [0,1.]

We may want >1 indep sources of randomness — but it's likely that >1 source is not necessary to get universal stoch functs. If we have say only 1 randomness source, (251.00) stream

9/25/04

TTM

among known cands, then we have a good chance of finding a new cand that does them all". In general, we should try hard to find cands so that each problem has at least one (& probably ~~many~~ a few more) cands that can solve it.

The situation at 248.37-.39 is not so good — try to avoid it by making lots of "quick tests on "hard" problems — esp in 249.37-.38, using the p.d. generated by cands that solve other problems.

If we continue to have trouble w. several of the problems, try using cands that are not as closely linked to the present "most successful set of cands". This might be done by using data from less successful cands. I might be able to implement this by accepting data (in PPM, say) from less successful cands. (This is equiv to ↓ X_0 in 234.19 -.22, .37

SN Make a list of ~ 4 or 5 Approaches to TM that I'm interested in, — Discuss advantages, disadvantages of each. See if JD can use them to fix one or other) — 250.10

SN In deciding on the entropy of string b wrt. string a: the usual way is to find entropy of ab & subtract entropy of a.

An alternative way that may be better: from a, make PPM Σ a prob of a symbol following any entry. Use this to find the pc of each symbol in b; T. in effect pc is it. It differs from .13-.14 in that in .13-.14 the corpus to determine pc of next symbol, includes the rest of "b plus tsw"; i.e. .15-.16 we do not include the "b" in the corpus used to create the "pd for next symbol".

The "problem Pool" approach at 248.21-249.10 is an idea I had for an advanced method of Machine Learning. The machine, Given the pool of problems, decides itself which ones to work on, when, & how much time to spend on each. These ~~decisions~~ allocations were to be made by TM_2. So, in its "Phase I" it would guide itself in its choices of problems to work on.

All the systems that have Phase I, Phase II aspect; w. TM_2 at work in Phase 2. In "idsia", Phase II may be a little difficult. So I can use any type of Phase I (including conventional GP, say), and use the induction techniques learned to work on Phase II of any of the models.

I vaguely remember a discussion of the distance between the GA(b) approach and Ideal Phase II: ——— Where was this? 241.34-.40, 243.00-.05: I guess the diff was that GA(b) would work for normal GA problems, but not for TSQS: It didn't have a way to restrain info from previous problems. If I simply included my G cands of previous problems in the corpus — however, the problem of finding a cand that works on all of the presented past corpus.

What I was considering recently (248.00 ff)

20:24:7.40 : Consider ordinary GP as a u.g. oz problem solver: A main diffy in getting it to do TSQs is That Testing would take too much cc; Each cand would have to be Tested on whole past TSQ.

A way to ≈ fix this for GP: Each problem is solvd individually; new cands are tested on that problem only. The TSQ enters this way: For GP, the population includes top several cands of recent past successful prob. soln. We can keep pop size reasonable by only keeping a total of say 500 in population. As new problems are solved, we delete (older/less good) members of the population.

A (perhaps) better way is to store S-Grammars of the population. If we use PPM-like "Grammars", we have to decide which cands to include in the "population" — a/o what Gores to assign to them (i.e. diff. problems).

Another way is to use ≈ PPM & store only the last 200 (or 25 or 50 or 100) solns to problems (Recency heuristic). The forgg. is about "Updating the [illegible] PD." To solve new problems, perhaps use Lsrch to generate most likely "solving" function. — or use a wtd sum of the most likely "solving" functions. } ?? → 250.00

14 Just what is the problem? ① I want to store/have available solns & abs. used in solns of previous problems — in order to conduct an intelligent srch for solns to new problems. ② I want to be able to narrow down the set of abs most likely to be useful for a new problem soln. ③ I want to be able to update both ① & ② after each new prob is solvd a/o after the "correct" soln. is obtained.

As part of ② (perhaps) to be able to store info from a very large PD of previous probs, yet

0 — be able to narrow down likely relevant things (as in ②)

21 Another Tech: (I've written on this approach Before): we have a large set of sample problems. Our first goal is to get OV's that can work as many of them as poss. Usually it's easy to test cands because they fail on the first few problems tried (at random).

Say there are 20 problems in the pool (") we have several cands that can solve α problems. When we test a new cand w. random problems from the 20, we can tell, from the no. of solved v.s. unsolved probs thus far — what is the pc of it's solving α or more of the problems . . .

We can use this as a rejection criterion for new cands ("sequential testing").

For small values of "α" (≈ 2) we will have to do at all 20 or so problems befor we reject a cand . . . But this is O.K. Each test has a good chance of discovering a cand w. a useful "regularity". As I see it now, no matter how many trials of probs we have to make,

0 each of these trials is useful & has possy of yielding impt. info.

As the forgg. testing proceeds, we will find certain problems tend to be hard to solve, so we will test on those first, to eliminate inapt cands as soon as possl. (Also at certain points in the search, we may want even more cands that can solve the "easy" problems, so we can get a better P.d. to generate trials that might solve the hard problems.)

→ This is a Bad selection, see P 249.03

37 When we have many cands that can solve 18 of the 20 problems: If its the same 2 problems that are unsolved by all, then we will use ≈ PPM for our pd on new cands

.38 to try to solve the 2 hard probs. (see 249.03) If the 2 unsolved problems are drastically

0 (space 244.18): In working on "S" (stochastic) problems, using soft R functs: For S problems, testing can be statistical. — No need to test all cases. We only get a "probabilistic certainty" on a "probabilistic Question"!
There is an optimum amt. of time one should do statistical testing — it would be a "sequential test", w. no "a priori" numerical cutoff ... cutoff pt. depends on "how test is going". Again, the thresholds of "certainty" would be determined by an analysis like bank-loading sit.
Equ. at 124.33

06 This "statistical testing" idea may be VERY IMPT ... if it works! → (.27)
T. necessity to test a new O^j on all past problems is clearly impractical. The "Hard R's" is one way, but I think soft, statistical R's seems more reasonable, more generally applicable. (.34)

09 SN in .eoff almost all of TDS's computers, only to certain limited precision is needed. It is very wasteful to use 32 bit numerics if 16 needed! IDEALLY we should design our computer so it was able to use only whatever precision it needed on use as to take advantage of it & speed up cc in all other cc economy.

f (.09) Oh so picture this! We have a "front-end" that looks at problems & assigns it (probabilistically) to a PEM. This is essentially "Phase II" of CDSIA Report — It does not seem to same as "soft" R functs, for problems like ANL — even on a S-version of ANL! — where we assign pc's to various qualit. functs of Alg. Ack passing.
T. way "S" R functs work: we input problem to T. front end assigns it, probabilistically to a Probabilistic funct Rest for PE assigns a P.D. overall pass A's. PP 10 ff (§ 1.2) of CDSIA: an updating! Has ideas on how to do it w. d-R functs. They would certainly have to be modified from S-R functs! (.50) P_j
Accepting a Modified P_j defined by

2 : (.06) On "statistical testing": If we make a "mistake" by not testing a certain case Q_i, A_i — This is not too bad, because we will eventually test that case on subsequent problems — the only effect P_j has been modified.
Furthermore testing P_j on only some of the QA, means we will have to decide how good it is by comparing its pc's for a subset of QA's with the pc's assigned by several different P_j's. — So it isn't very clear. If the pc for all past accepted P_j's was $\geq .9$, say one can easily test the rel. new P_j meets Pass Criterion. Also, one might have calculated a mean lower pc per A ... & use Pass criterion for testing new P_j candy.

) : (.20) I guess this "a level" O^j function is OK, w. stochastic R. A non-st. approach would try to optimize O^j directly. By eliminating O^j into R & P functs, we probably have to give separate criteria for updating each. Hmm, ultimately, the system will probably settle on one P function that does all the work. Since this is Non-st., we wouldn't be really giving long term restrictions on the system by Pass R, P division. A big Q is how to update R & P funct.
Another Q: When TM "normalizes" itself into merely one R — How do we generate test? — How do we avoid testing on all past QA's? This problem would be "Automatically" dealt w. if we use a good TM & doing the updating & improving the update method.

9/22/04

4TM UMC's and simulation strings.

.00 : 242.40: As for t. no. of branches due to duplication (in the tree for Min sim strings), t. corresponding ($\lim_{N\to\infty} \frac{M(N)}{N}$) = 0, for [illegible] same reason, as 242.40, but I don't know how to show that $\lim_{N\to\infty} M(N) > 0$ — or that $M(N) > 0$ for any N. — i.e. that there are > 1 "Min codes".

.03 Q: Is t. theorem of 242.38–.40 (i.e. $\lim \frac{M}{N} = 0$) related to the P ≟ NP problem? It has to do w. t.
.04 rate of growth of duplicating programs. (.10)

.09 : 242.40 NB: Actually t. theorem of 242.38–.40 says that t. sum of t. no. of bifurcations due to earlier duplication & "simultaneous duplication" ("simultaneous" means "same length") of 242.31–.34 must behave as $M(\infty) > 0$ but $\frac{M(\infty)}{\infty} = 0$.
The argument that t. sum $M(\infty) > 0$ is for t. not same length duplications

{ Actually t. $\frac{M}{N} \to 0$ theorem is Non-intuitive Because as N↑, t. no. of UMC's simulated "thus far" so t. one would expect t. no. of "duplicating a previous UMC" branches would become more likely!

(.03, .04): A possibly related (to P ≟ NP) is t. idea of Prim. rec. functs: that are expressible as a finite no. of nested loops of certain kind (s).

Every partial ordering relation can be expressed as a Boolean Net: Would it be instructive to look at t. sim string from this way.

Or t. Branching of t. sim. tree for UMCs.

9/22/04
4 PM

10 (Space 246.40): Also note that Lsrch is mainly for INV problems & one usually doesn't solve INV probs as such, but usually converts them first to OZ probs.

Is t. proof of optimality of Lsrch ~~appl~~ applicable to OZ problems?

[SNO] T. "proof" of optimality of Lsrch is very ~ to "proof" of optimality of Occam's razor: Both are contingent on "All info is in t. GPD"

[SN] The Early Sections of t. IDSIA Report have a lot of reasonable-sounding ideas on heuristics, & ways to design TSQ's: Perhaps reread this every once in a while!

An objection to "proof of optimality of Lsrch": That it doesn't allow for lrng (modification of GPD) during srch. Well, if GPD changes during Lsrch, it could still contain all info available for t. srch": Hvr., to "Plan" t. srch properly, something like "Dynamic Prgmg" is needed. Say "Dyn Prgmg" info is in t. PD. ~~Could this of~~

0 How could this "info" modify t. Lsrch "optimally"? Say one of t. choices in searching to use "Dynamic prgmg" to decide on optimum strategy for making cand. trials — Is this a meaningful thing to ask?

20 : [SN] Tree Contexts are "cc expensive": ~ 10 to 100 times as much (space & prob'ly time) as normal PPM contexts. Hvr, we can balance pc corpus ↑ v.s. cc of this update system — yielding an analysis like that that led to / equiv. of 124.33. i.e. decide how much time to spend (continuous) on updating.

A v.g. way to ~~design~~ avoid this cc is by Definitions. A definition need not be substituted for its sub-tree (or whatever) every time that sub-tree occurs. T. substitution can be done when one is "fairly sure" it's appropriate. Also, this substitution is usually reversible w.o. much diffy. — One may want to "Unsubstitute" certain defns when reparsing t. corpus.

It will certainly pay for me to spend a fair amt. of time trying approximations & shortcuts to implement tree-contexts.

0 Another perhaps equivt. & perhaps better way (~ continued definitions) is Z141 and CFG discovery & Context. Grammar discovery.

9/21/08

4TM

10 : see 243.40 Will be few problems, This will not take long. Later, X_0 will be large and we will test on many problems. Hvr, most will fail on ~~early~~ enuf so that their G's will be $< X_0$ so we can quickly discard them. Only cands w. ~~[illegible]~~ close to or $> X_0$ will take much testing time.

We may isolate certain problems that have been dif'lt this far & use them for testing first or "early". For large X_0, we want to find failures as fast as possl., to discard (cands w. $G < X_0$) rapidly.

Keep census of ~~[illegible]~~ fraction of failures for each problem. (Presumably it does $X_0 \uparrow$.)

At certain pts. in TM's testing, it may be possl. to test on certain problems & do some random tests on t. rest of t. problems. For a cand that is at ~~or~~ G_0 or $\geq G_0$ we test on all problems.

0 → Hvr, TM should be able to work on t. problem of ~~[illegible]~~ finding more efficient ~~[illegible]~~ methods of this sort.

.. I could give it an initial corpus of "more efficient" srch methods — have it look for common s eqs / subtrees in that corpus.

2 In what ways (if any!) ~~did~~ was my approach to ANL in IDSIA report better able to deal w. this problem, that 243.27 ff (GA) ? In IDSIA rept. I had "Recognition functions" that would look at t. problem & do a much more "focused" search for soln. Because of the "Digital" nature of "R" functs, there was no need to test solns on ~~[illegible]~~ any but a few of t. problems. "No need to test a soln. of a Math problem on all Chemistry problems."

It would of course, be possl. to devise "soft" R functs, but they would not eliminate ~~[illegible]~~ — V.G.!!

18 the necessity to test all problems on new cands. — (Well → I'm not entirely sure!) — see 247.00 !

With a "soft" but relatively sharp F we could decide not to test an algm on certain probs because it was

0 very unlikely that t. algm would be tried on those problems. One could note that t. algm hadn't been "tested" on a certain "domain" & so one would not expect to use it in that domain

22 w.o. suitable testing.

This is Mindful of PSM's operating on problems. Each PSM is, say, usable for only a certain set of problems. If we modify t. PSM, we only have to test it on t. set of problems it is restricted to.

We would, hvr, like to have a PSM that can work on all problems (deciding, itself, which sub-progs to use or which "external" PSM's to "CALL")

Hvr., in ANL (.12)† we have a d-problem — we need a d-funct to map algo expressions into events ≡ "values".

Note: In IDSIA report, ANL was being solved like INV problem, using Lsrch. I think this is the wrong approach! An INV problem should be given a "continuous Gorc & solved as an OZ (or GPS) problem.

0 Occasionally INV probs can best be solved by Lsrch — but this is unusual — only done when we can't assign a reasonable "Grey scale" Gorc.

243.27 – .40 is one way to xlan a ANL problem into a "multilevel Gorc" OZ problem. INV: 2 level Gorc

Tho it's a somewhat general method, it doesn't find a v.g. Gorc .... It's Gorc doesn't really understand what needs to be done — (or, more exactly, its "understanding" is not "v.g.")

SN ① Note discn. of "Analysis, Synthesis; (Breakdown, Unification of R-functions) in (IDSIA Report): These operations are an essential part of "Human" "Sci Method". — p.12 end of § 1.2

② Lsrch & GA ($\equiv \frac{G_0}{G_0 - G}$) seem to be ~~[illegible]~~ powerful, but essentially distinct, approaches to prob. solving. Pricing max — is "like" Lsrch; but Mt carlo (analysis of (search) resultant (best population), is not.

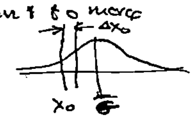
Also note "≈ Proof" that Lsrch was Best Possibility, w.i. a factor of ~~[illegible]~~ maybe 2.

space. 245.00

Spec. :241.40: Hvr, [redacted] GA(b) does limit us to <ragy types in corps> of 241.36–37. This may, indeed, be good enuf for almost all problems, but we do want t. system to be able to discover arby ragys.
T. BEST way to do this is to give TM t. problem of improving its own Ragy discovery Algm.
GA(b) is [redacted] (w. suitable TSQ) almost certainly able to usefully work on such a problem.
→ IN Designing t. TSQ: I will know what kinds of ragys I expect TM to dscvr. — So I shouldn't approach this problem w. such trepidation!

Another angle re PSG discy — Here we devise definitions, subgrammars, recursions. We can have
Context dependence to ↑ power of Grammars (perhaps, in theory, Universal).
I could just introduce this as an improvement of "PPM". — But it would probly be much slower than "PPM".
→ Re: CSG's To detect some contexts is very easy! First obtain CFGram. Then, when corpus is parsed, keep track of "contexts" for each substitution (Just as in PPM). If a context condition can ↑ pc of t. corpus, Do it! .... Perhaps, hvr we don't want t. context to make its assoc substitution
? → Mandatory, so, like normal CFG's, we want to be able to parse in alternative ways,
Spec. 15ff
Well, t. subs rules can be mandatory & still one could have alternate parsings.
— Depends on what one means by "Mandatory": In CSG's we will have alternative production rules, just as in CFG's. So: If a production rule is used in a CSG, we do have to always respect its context (if relevant).

: 239.18 ⟺ SN Ont. "On line" adjustment of X_0 ■ After one has just chosen an initial or a new X_0:
Pick a ΔX_0 "region R" $X_0 < X < X_0 + \Delta X_0$. As we continue t. run,
Keep track of $n, \Sigma G_i, \Sigma G_i^2$. [and G_0 (= Gmax)] Do this for all $G > X_0$: but also keep a separate file for these 3 quantities for Region R. From this data, one can tell if $\frac{\sigma_G}{G_0 - \bar{G}}$ is greater for t. recent corpus with & w.o. "region R". So we can tell whether we want to move X_0 to $X_0 + \Delta X_0$. It may be possl. to estimate ΔX_0 from a Theoretical Model.



Consider 241.34 : Start TM w. GA(b) .... etc.
We begin w. a tsq for which we know acceptable answers (not nec'ly min answers).
Doesn't seem to work w. t. problems at t. beginning of IDSIA Report. T. diff is: As I give successive problems, I want for each problem a soln that solves that problem & all preceding problems.
One way to think about it: Say my ANL TSQ has a bunch of (Somewhat) ordered problems: I start out w. a nearly null corpus: so probys of all insts are same, & no context dependence.
Next I try to solve problems in t. TSQ, individually — I solve as many as I can. Then w. t. set of solns as corpus, I use "PPM" to try to solve new probs a/o the problems in t. corpus — w. k as large as poss'l. (a Fitness function like in GA).
So I have this corpus of Cands: Each has solved a no. G of t. problems. I can then restrict my corpus to cands that have $G > X_0$ (i.e. have solved $> X_0$ problems). X_0 is chosen by 239.07 $\left[\frac{\sigma}{G_0 - \bar{G}} = \max\right]$
At t. beginning X will only have 2 or 3 possl. values ... but keep on searching until X is much larger.
At first we will have to test each cand on all TSQ problems, but since there spec. (245.00)

On UMC's: Ordering them by simulation insts.

00: (238.40: spec) must be non-universal, so ~~[struck]~~ a relevant case simulate a UMC, so there can be no connection of $S_{12}0$ to $S_{12}1$ that simulates U_1 .... Yet we know that there exist S_{21} so $S_{12}S_{21}$ simulates U_1 so. ~~S_{12} can't be followed by~~ Either $S_{12}\phi$ or $S_{12}1$ or both must be a sim. inst. for some UMC.

03 So it looks like the idea at 237.00ff is correct!

04: 238.29 Consider "random" ~~[struck]~~ min sim. instructions. The model at 238.15–17 ; say we have this SRT (238.02). Say the frequency of "λ" in the graph is some constant, $c>0$. If this is so, then by moving down the graph and picking the rt. hand choice ~~[struck]~~ λ branch, we will get a ~~[struck]~~ min sim. code with ~~[struck]~~ a fraction "c" more ~~[struck]~~ 1's than zeros. This cannot be compensated for by having more 0's in general, because if this were so, one could go down the tree and pick the left (φ) choice each time & get a ~~[struck]~~ preponderance of φ's.

10 One possible way out is to have freq. of $\lambda = \phi$ — they never occur or occur a finite no. of times.

12 If they never occur, then the SRT (238.02) tree is a single infinite string, and all min sim. codes are prefixes in this string! — A truly surprising result! → Buffalo
Assoc. w. each UMC is an associated infinite ("random" string) that ~~[struck]~~ has all min sim. instructions of all other UMC's.

(They have to occur a ~~00~~ of times! See .29–.36)

16 If min length sim. insts can occur > 1 way, this can't occur "too frequently"; if it occurred frequently enuf, we could trace a path that ~~would pass~~ picked min length

.18 codes in a way so as to make no. of 1's significantly > ~~[struck]~~ no. of φ's.

(I haven't proved this ~~[struck]~~ convincingly, yet!)

20 So: Another possible result is that duplicate min. codes occur infreqly (or never)

If .12 is true, then all UMCs are related to all other UMCs in a simple ordered way;]!!
~~[struck]~~ Each has ~~[struck]~~ unique integer that tells how long the code to simulate the other is.
Also, there is just one ~~[struck]~~ UMC's w./ min descr. length n. If there are more, then one or more of the 2 assumptions (.12 & .20) can't be true. If there is < one UMC for min descr. length n, then it looks like serious trouble in the gen'l. model (?)

In fact, I've been considering only one UMC & its sim. insts. — However, in the complete tree of all UMC ~~or simulation~~ sim. strings, we can pick any ~~[struck]~~ point on the tree, ~~[struck]~~ the UMC at that point will have as its sim. tree, the ~~[struck]~~ sim. tree that it is the parent of. So ~~[struck]~~ the sim. tree for any UMC will contain the sim. ~~tree~~ for all other UMCs. ~~These [struck]~~ These trees contain UMCs only.

29 If we look at representations only have min codes, the SRT (238.02) will not have complete SRT's as subtrees,

This is because every SRT has to have all UMC's & ~~[struck]~~ any subtree of it cannot contain all UMC's.

31 I forgot ~~[struck]~~ to make it clear that the original UMC tree must have some branches w. bifurcations. If it did not ← Not SRT, which contains no duplicates.
then it would be a single string, and since each UMC has to be able to simulate itself (See 238.36)
— Say the $S_{12}S_{21}$ Then the ~~[struck]~~ entire string would be $(S_{12}S_{21})^{(\infty)}$, which can only simulate

34 a finite no. of UMCs.

(The tree on all UMCs once cannot be able to simulate ~~[struck]~~ itself — there would be duplication)

35 So, the original UMC tree must have 'some bifurcations' but it can't have 'too many' of

36 them (see .16–.18)! (Also note .20!) The mean no. of bifurcations per symbol, as one moves down the tree, must → φ. → 265.20 →

38 More exactly: Say N is the distance along a string of the UMC tree and M(N) is the no. of bifurcations thus far. Then $\lim_{N\to\infty} M(N)=\infty$, but $\lim_{N\to\infty}\frac{M(N)}{N}=0$. The argument of .16 can be made rigorous: if $\frac{M(N)}{N}\to k>0$

40 then there must be a way of coding the string more compactly. (This is a proof). → 246.00 Also NB 246.05!!

.01 : Comment on IDSIA report. P19: $h^{j,r}_{j,\ell}(G^{j,\ell})$ is a prob density. Is this density in r. G or T directions or both? T is fixed, so the density must be in G direction.

Eq. (11) (ibid): if we use $d G^{m,\ell}$ should we have $G^{m,\ell}$ under the S? or perhaps $h^{i,'}_{m,\ell}$ is the inappropriate one.

.04 eq.(11) ···> $\int_{-\infty}^{\infty} G\, h'(G)\, dG$. Looks O.K. Its the first moment of $h'(G)$.

There is no natural analog of the scope, θ of (240:21h), since the origin at G=0 is of no particular significance.

In fact the criterion of selection for GA(b), which is 239.12–.14 is identical to (.04)! So the main difference between PhII & GA(b) is the method of construction of the set of PSM's (≡ cands) to be selected from! — which seems

10 very distant.

GA(b) can be regarded as a particular PSM: — or rather, the choice of X_0 ←(234.21) gives a particular PSM. This PSM consists of multiple trials on the lang of strings defined by the value of X_0. As we execute this PSM, we slowly ↑ X_0, we change the PSM, to a more optimum PSM, in line with the criterion

$$\left(\frac{\mathcal{E}}{G_0-\bar{G}} = \text{Max}\right) \leftarrow (239.07, \equiv 241.04–.10)$$

.16 : To "sharpen up" GA(b)! The original set of cands that defines the s. lang, are carefully chosen

.17 so that they are very "relevant" to the present problem — the criterion of "similarity" being based on

.18 comparison comparisons: i.e. we look at many "problem domains" / consider the set of problem domains that

.19 are closest to the present problem. Take the set of cands that did well in solving those problems

.20 — and use those cands (via PPM) to devise a s-lang using X_0 ←(234.21) to get a PSM to solve the new problem.

.22 .16–.20 looks V.G., But .17–.19 is not so clear! → Note (.35) — i.e. I'm not sure it can be meaningfully defined/realized.

Another tack (perhaps better), is that ultimately, we don't want to have to Match PSM's

25 to problems: We want a single MASTER PSM that is able to solve any problem by looking at the problem, deciding what kind of problem it was, and deciding on the best

27 approaches for that problem. The techniques I've been describing so far, for the most part, more "el" in that they develop special techniques for looking at the problem, deciding "what kind of problem" it is, & applying techniques that have in the past

.30 been successful w. that "kind" of problem.

This (costly less ell) method (.25) could be done using a suitable TSQ. The TSQ is devised so that the methods of .27–.30 are likely to be discovered, but the search method used is for a single general soln. to all problems (or a wide set of solns to all problems). → But wts. are made of problem domain.

34 Viewed in this way GA(b), using $\frac{\mathcal{E}}{G_0-\bar{G}}$ & a suitable TSQ may be good enough to be a "final system".

35 • Note that this is w.i.o. the ATTempted refinement of .16–.22

36 Also Note that the "PPM" used in GA(b) is only as good as the cand "solns" it has

.37 in its corpus plus the techniques for finding regy in that corpus. ——→ see (243.00)

9/18/04

4TM

30 : 239.40 : Phase II ≈ GA (ⓑ) try to solve OZ probs in (apparently) quite dif'rnt ways: GAⓑ has a somewhat narrow [illegible] s-lang of PSM's to try. This (set ~~is~~ of PSMs) s lang gradually changes as to mean Gore of trials 1. But GAⓑ simply tries to get as much in G per trial as poss'l. (Using the Gore of 239.07, it is better than simply trying to get a better G than up to now ... so its better than normal GA. It does have a better "understanding" of optzn than Normal GA has.

In Phase II, I will have an s-lang of PSM's. It is obtained by "factoring" a large set of v. G. PSM's & expressing them as some kind of s lang. The assignment of PSM's to probs is a diff't problem ... looks about the same in Ph. II or GAⓑ.

Many PSM's modify themselves to best deal w. a particular problem; for PSM's of this kind, a "frontend" of a function that looks at a problem & decides which PSM to use ... is unnecy.

0 In GAⓑ T. PSM's do not ordinarily try to adapt themselves to the problem. The "pre-adaptation" occurs in the P.D. on PSMS defined by the s-lang. (Tho, actually, there is no reason why these PSM's couldn't adapt themselves to t. problems, just as in Phase II.

3 **DEF** Def "Phase II" means Phase II as Dscrbd in IDSIA Report: §3, 11 18, 19, 20 describe appl'n to OZ probs

4 Def "GAⓑ" means the GA system of 236.21, .24, presumably w. refinement of 239.07: $\left(\frac{G'}{G_0 - G}\right)$ 234.19 & 235.25 - .40 give possl. models of GAⓑ

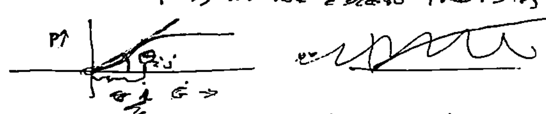
It mite be well to dscrb PhII in a little detail now: My present dscrn may be somewhat difrnt from IDSIA report, in view of my recent analysis of PPM & PPM "like" systems. — Assume we solve only OZ probs. (to start).

So! We have the set of PSM's [PSM_i], which (presumably) were obtained first as a discrete set of

17 10 or 20 PSM's that I inserted into the system. [[At a more advanced stage, I factor these PSMs, & express the set of PSMs as a s-language — so (perhaps) any PSM will be in that set. — If it's truly a Universal]]

10 Before we do .175 ff, we have a (finite, discrete) set of say 10 or 20 PSM's. A problem comes in, Prb_j.

21 From our experience w. PSMs & probs, we are able to take PSM_i, Prb_j & T_0 (t. CB for an optzn problem) and obtain a curve



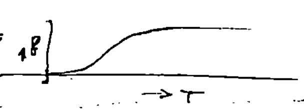
that gives t. prob'y that PSM_i will obtain Gore G in time T_0. The angle θ_{ij} (or its tangent) gives an index as to how good PSM_i is on Prb_j for Time T_0. — we use θ_{ij} or tan θ_{ij} to order the PSM_i's for our trials to solve Prb_j.

If we use the "advanced version" (.175 ff), we can create an [illegible] of PSM's & select t. one w. max of tan θ_{ij}.

Something Wrong here! $\frac{P}{G_0}$ is [illegible] at that point! But t. Gamb. house theorem would not apply in this case!

It applies to INV problems in which we have a additive cost function per trial. Doesn't seem to apply here! — This is all wrong! it applies to INV probs, not OZ probs. See .31

0 I think it was for inv. probs



P = proby of soln. in time ≤ T.

31 So how did I apply it to OZ problems? In fact, I applied t. Gamb House theorem I (IDSIA P17 fn4) to INV problems not OZ problems. The methods of WON were used for INV, not OZ problems. I think that on pp 18, 19, 20 of IDSIA I did OZ probs & used "Expected value" of G as an ordering param. for trials. (By assumd "linearized Gore"). → (see 241.00 ff)

00: (spec 236.40): Another Diff betw Ph. II & (b)(236.21,.24)! In Ph II I pick a PSM carefully, then spend a fair amount of time applying it, until the problem is solved or until that PSM no longer looks "Best".

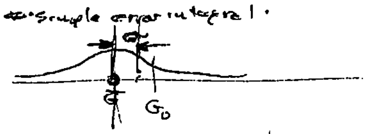
.02 In (b), hvr, I just make many successive trials — perhaps rather quickly — → (.32)

It could well be that they are essentially different techniques: Tho it may be possl. to apply the "Gamb. House Thm." to the Method of (b): To show $\bar{G}+\sigma_G$ order is the best possl. ordering for trials.

Hvr. suppose I want trials in order of pc that a cand will be better than Best G thus far?

.05 For Gaussian DF's, if I know μ, σ & G value that I want to exceed, its a simple error integral.

.07 Or, more [struck] the pc ordering its $\left[\frac{\sigma}{G_0-\bar{G}}\right]$



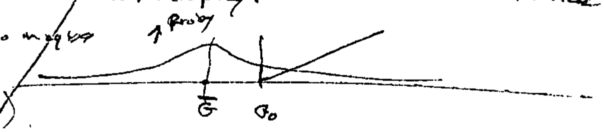
There is at least 1 situation in which the Gov. of .07 may not be so hot: i.e. it may give a very hy pc for a very small Δ in G: hvr may prefer a smaller pc of a much larger Δ in G.

NB .11ff. This formula is optim. ⟺ Utility of $\bar{G}$ has been linearized

.10

.11 O.K. Then the "linearization" of G comes into play. We want to maximize expected value of G that's above $\bar{G}$. So maybe

.12



G · P(G) Gauss. $(\bar{G}, \sigma_G)$

.14 We want to max $\int_{G_0}^{\infty}(G-G_0)\,P(G)\,dG$. So find $\bar{G}$, σ, G_0 triplets that do this.

.15 → T. Gov. cr. .07 does exactly work for this Criterion!

Also Note: We can use .07 in "Steady State" (non-batch / generation) mode: when we Test a new cand: We can tell how σ, $\bar{G}$ (& perhaps G_0) are modified — This will tell us how much (perhaps) to change the X_0 (Threshold) of 234.21 or other param(s) that determine the S-Lang that generates Cands. → (243.20) →

.18

NB For the optimal $\frac{\sigma}{G_0-\bar{G}}$ S-lang. it would seem that Lsrch would not be relevant. Picking trials on Mt. Carlo basis may be Best! → While → Doing Lsrch may not get desired $\frac{\sigma}{G_0-\bar{G}}$ ← !! ← NB

Cover–Kelly (1978?) NB the Cover-Kelly Paper says Kelly betting is best even if pc of ruin is >0. ... clearly on money alone this it seems wrong. — Prob. Soc's on Cover's Paper That is relevant.

.20: [SM] on UTILITY functions: I discovered that probability of "Gamb. Ruin" was an imple negative Utility in SM strategies: That a ln of "Money yield" Gov. was good: it was not the ultimate Gov: One used "kelly coeffs" <1 to reduce pc of "Gamb ruin" (Gamb. Ruin) in a rather fuzzy, Ad H. way.

How to do it more accurately? Perhaps try linear or ln of Money yield version PLUS a large Neg yield for Gamb. Ruin. It may be possl. to design strats in which pc of Gamb. Ruin = 0 (I.E. in response to utility of G. R. = −∞).

So: try strats w. U(G.R.) = −∞ w. U of Money = Money & also w. U of Money = ln(Money).

One Attractive (at first) scheme for U(G.R.) = −∞: Bet ½ of Bank: Invest ½ Bank in fund w. low yield but cp. of G.R. = 0. Trouble is, if one loses all of Bet, one ends up with $\frac{B}{2}$ (= Bank) and the process is repeated again and again. Eventually one can have B so small that one can no longer bet (≡ G. R.).

.30 So it looks like while U(G.R.) = −∞ is a reasonable/acceptable strat. for 1 bet, it will not be workable for a "Long Time strategy".

2, (.02) Phase II vs. (b) (236.21–.23). ① In GA one uses the G funct to derive a TSQ for an OZ problem. In GA, we try various Modifs of PSM's. There is an Enormous no of PMS's to try & we "hillclimb" on them. The PD for the Cands is devised by looking at the Problem & Correlating its properties w. those of similar problems of the past.

Phase II has a much smaller set of PSM's, but it is (perhaps) able to more carefully judge the applicability of each of them to a particular problem. The technique for judging applicability is same as that of (b) in GA, but the ssz is perhaps larger. (?).

4 TM Simulation Tree of UMC's

Spec: .0: 238.40: This is because the length of a code is ≥ its row, & we always rotate. First row a code occurs in ∴ min length. So only min length ~~code~~ simulations occur in it.

.02 Def (SRT ~~=~~ Super reduced tree (SRT).
reduced

(1 4) The ~~[illegible]~~ original reduced tree, is self-similar, in that any subtree rooted at a particular ~~[illegible]~~ simulation of U_i, is identical to any other subtree rooted at a sim. of U_i.
"The super reduced" tree does not have this redundancy, since it consists of min codes only,
it may be that this tree is completely random — it has no regularities at all!
(I'm not sure of this "randomness", hvr). — If it is a random ~~[illegible]~~ tree, then considerably property:

.09 At every node the proby of a (left) branch is (.5). This would enable one to determine the density of UMC's in psm space!

.1 If .09 is true; does this imply expected length of tree is ∞? if not then its wrong!
The probability of any particular path is $\frac{1}{2^n}$ (n is length of path) & there are 2^n paths of length n —
So we expect one path of length ∞.

Unclear: At a node: 4 possys of = proby: 1 2 3 4 , null, /, \, ^.

---

15 Say at the n^{th} length, the pc of the states is $P_1(n)$, $P_2(n)$, $P_3(n)$, $P_4(n)$ ← Meaning?

16 My impress. That at the n^{th} node, the expected no of / 's is $2^n \cdot \frac{1}{4}$;

" " " \ " $2^n \cdot \frac{1}{4}$

17 " " " ^ " $2^n \frac{1}{4}$

So at $n+1$ node expected no of / is $2^{n+1} \cdot \frac{1}{4} \left(\frac{1}{4}+\frac{1}{4}+\frac{1}{2}\right)$

I'm still confused about this. The meaning of the "randomness" of the minimal pgms, etc.
.0 A node could have 1 or more children = A, if one ~~[illegible]~~ is not Umc. (Both could be non-umc or in matching but unclear). Once both children are repetitions of a previous (shorter) code.

From .15 — .17 The expected no of nodes at the n^{th} ~~[illegible]~~ level is constant, indep of n. — ~~[illegible]~~ So for large n, we ~~should~~ must have Gambler's Ruin. — So it couldn't be right!

→ If the [illegible] fraction of \, /, ^ at each n were the same — then no Gamb. ruin (or ≠ pc = 0).
This last seems more reasonable! In fact, if the pc of "total end" is $< \frac{1}{4}$, one gets Gambler's ruin $pc \to 0$. (The pc ≠ 0 but → 0 as $n \to \infty$).

Every Univ sim code must have at least 1 sim. code child. ... That child may be the same as a

29 shorter sim. code, in which case in SRT(.02) that child is aborted. → 242.04

.0 Some Q about whether the model of 237.00 ff is correct! The defn of universality:
If $F(x)$ is an arby (partial recursive) function, then if U is a universal Umc (or function) ∃ a string S_F ∋ $U(S_F\,{}^{\frown}X) = F(x)$ (concat) for all x for which $F(x)$ is defined. In particular, $F(x)$ may be any other Umc.
$U(X)$ has to be defined, hvr. Say U is a / U (3 tape) machine; so ~~[illegible]~~ w. input tapes X,
U ~~[illegible]~~ eventually produces a finite or infinite string — this is its "output".

Could the sim codes be a prefix set? Well, looks impossible — proof: Say S_{12} is a sim code from

36 U_1 to U_2, and S_{21} is a sim code from U_2 to U_1, then $S_{12}S_{21}$ is a sim. code from U_1 to U_1.
~~So~~ Both S_{12} & $S_{12}S_{21}$ are legal sim codes for U_1, & clearly S_{12} is a prefix of $S_{12}S_{21}$.
Also, if S_{12} is a sim. code for U_1 ~~[illegible]~~ to U_2, then ~~[illegible]~~ $S_{12}0$ and/or ~~[illegible]~~ $S_{12}1$
must be a sim. code ~~[illegible]~~ from U_1 to some Umc. If not, then $U(S_{12}0 \cdots)$ and $U(S_{12}1 \cdots)$ 242.00 spec

9/17/04

4PM

The Description Tree of the Set of all UMC's .00ff

0 : on UMCs: Say we have a particular UMC: U_0 Consider codes for UMC's :

Any other $U_i(\alpha) = U_0(S_i\alpha)$. for all α. Consider $[S_i]$ such that $\equiv S$

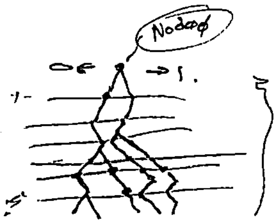
all U_i are UMC & all UMC have S_j in $[S_i]$.

[See 238.30 -.40, 242.00 -.03 for proofs of some critical statements below

1) every S_i is the prefix of another S_i

2) All the prefixes of an S_i are in S.

3) The members of S form a simple tree like :



But the tree is infinite and there are no "terminal" pts.

.06 4) One way to enumerate them: say. n is the order (row) of the node. w. top node row=0 Start w. top row, list all nodes; then list all nodes in 1st row .... 2nd row .... ∞th row.

Note that $S = [S_i]$ is assoc. w. a particular UMC. Each UMC has its own set, S.

5) The tree ($\equiv S$) is ("self similar" "recursive") in the sense that as we move down any branch, we will (usually / occasionally) eventually come to simulation of U_0 (= identity). At any such point, we can put Node ϕ at that point.

(It is possible to walk down the tree & never come to a point that simulates U_0.)

6) If ever along a path down the tree we repeat the simulation of any particular UMC, U_j, then we can do the recursion of 5).

7) More generally, in line w. 5) & 6) if the UMC corresponding to 2 different pts on the tree are identical, then the downward continuations (branches) of these pts are identical

0 8) At any non-splitting Node, if one takes the choice not used by the tree, that alternative will not be a UMC & no continuation of it can ever be a UMC. So the tree derived contains all UMCs and only UMCs.

.22 9) Consider the part of the tree that has no repetitive simulations. We mark each branch where a repeat occurs w. the name of the UMC it repeats, but we don't include that part of the branch. [If it's not a branch pt. we just stop at that pt. Call this tree w.o. repetitions "The reduced tree". This tree does have terminals. Every UMC occurs once & only once on the "reduced tree" ← (This last statement may be false ! — see .28 -.31) → But .32 fixes it — see (.39)

.28 — 10) If S_i' is the shortest simulation of U_i (w.r.t. UMC U_0), then must S_i' be on the "reduced tree" ? Well, say there are 2 (min) codes for U_i of same length!

.30 — 11) (10) suggests that "reduced tree" could contain repeats of a UMC, if the repeats are in distinct branches (i.e. neither is an "ancestor" of the other).

.31

.32 12) We can take the reduced tree of 9) (.22 ff) and remove repetitions in the following way, take the reduced tree & remove any repetitions in the enumeration of the nodes used in (.08-.09) = (4). So we retain the first simulation & "left most" simulation. When we remove a node, we also remove any of its branches (below from it.) — we put a suitable marker at each pt. that we remove a node.

.39 13) If we do (12) then only the shortest codes will occur in the "super-reduced tree". — see 238.00

9.16.04

4TM

OB.OP.Alg: ORTHOGONALIZATION .08
GPS .05

00: ~~236.40~~ : **SN** (1) T. Ckt design problems Solvd by Koza's GP would seem to be as difft. as most problems of trying to find a function that satisfied certain conds.
His Goal for This problems was a set of conditions! Each which could usually be expressed in "soft" form, so it had a "gradient" — a "Hill slope" to work on.
Hvr, instead of looking at t. individual constraints that a Cond violated, & trying to fix
.05 them, it just tried a single global optzn. — i.e. I think New-Sim. GPS ideas would be very helpful in probs of This type. We do have techniques for looking at a ~~/~~ sub problem & deciding what actions are likely solve that "sub-problem" — (see 235.17)
.08 [It may be best to try to orthogonalize t. set of constraints that t. Global problem Soln must satisfy. TM should have much experience w. "Orthogonalization"!
.0 — A possl. approach to ob "Orthzn": we have this "soft" ob-op-result Algebra. It is stochastic, based on past observations. We ob serve t. system state; from this we can, for each possl op eration, get a Pd over possl resultant states. From t. Matrix of probabilities
We can compute an op that is most likely to advance t. Goal (which is some summation of functions on possl resultant states.
(2) Anyway: I want to Understand Koza's Languages for Devising Ckts. ——
It is likely that such a lang. can be applied to Many problems. (like Gabriel Kron's "Tensor Analysis"). — related to Kron-Wilkes of Kron-Rhodes?
Koza's Vol III (1999) ~~$130~~ has much on this. Hvr. He May have integrated much of t. stuff in this book, in more recent papers. Koza's Book IV (2003): 6 coauthors! Gives many recent results.
.19 .20 ——→ I also have at least 1 paper (Maybe from IEEE trans on EvComp) by Koza et al on designing Ckts, Op amps, etc.

Quick notes:

1)

.21 On P.D. for Population Modeling (a) v.s. P.D. for modeling "Populations" assoc G. (b)
T. GA community is begging to do (a). The Key claim is a sizable improvement over ~~popular~~ using population and Cross/mut; it's not clear as to just why, I should bother!
.24 On t. other hand (b) looks (in Theory) very promising if we can find good ways to do it. — Goal $=\left(\frac{\sigma_G}{G_0-\overline{G}}\right)$
234.19 gives one way to do this (Tho it is nonparametric in "G" direction) ——→ see 239.06 for a better formula
. Another kind of model is 235.25 – .40 (This needs a lot more work!)
. In Both kinds of (b) Models (perhaps all kinds), perhaps a good strategy for Cand generation
.28 is to chose a cand α_i → $\overline{G}(\alpha)$ (given by model) $+ \sigma_G(\alpha)$ is max. So ~~it~~ choosing that cand
.29 has most likelyhood of giving a higest G. Chose cands in this order (Max first, of course)
.30 How is .21 – .29 Specifically (b) .24 Related to t. activity in "Phase 2" of "Alpha" (IDSIA Report)?
In Phase 2, we find, for each PSM_i, problem$_j$ pair & T_0; a proby that ~~...~~ PSM_i will obtain, ~~for~~ when applied to Problem$_j$ for time T_0, will obtain a Goal of G.
In GA language, Prob j is fixed. (Perhaps T_0 is fixed). For each PSM_i we have a proby that applied to t. problem$_j$, G will be obtained in time T_0 .... for all values of G. So PSM_i is a cand in t. population.
T. problem because that of finding a good PSM_i. : The this seems correct, in fact, it is not. ——
.36 There are far fewer PSM's than there are Cands in GA.
Also Note: The Trial ordering Criterion of .28 – .29 seems similar to the truly optimum ordering scheme I want to use in Phase II — a real exact soln to t. WON problem! So maybe (b) (.21 – .24) is not so distant from Phase II!
Other than (.36) they seem very similar.

238, 239 are on "UMC Trees"

Spec (239.00) →

10: 233.29 : On Review: T. goal is a simple picture of project, telling what has to be done, & to some extent, what order of execution. Also, possible/probable difficulties & improvements, Bug fixes that I think of now.

How this differs (when it does) from my previous Phases, Phase2 approach in IDSIA report. I think the original idea was to simply "Improve" conventional GA/P

T. original idea was to improve GA/P in 4 ways:

Also use $MEJ = 239.07 * \max \frac{\sigma_G}{G_{max} - \bar{G}}$ .15

06 (1) Make Model of population: Use a PPM or My version of PPM w. possible further improvements such as sub-tree contexts. This should (at least) speed up GA &/or give it less "premature conv." problems

08 (2) Allow GA to use info from previous problems by (a) Using modified (updated) of previous population P.D. (b) Using a very general population P.D. & adding differences that seem likely to be useful. [i.e. Don't shift & forget after has solved a problem. Use Prior shifted slightly for new problem]

(3) The system should be able to initialize itself for a new problem. (This is really part of 2.)

In conventional GA this involves deciding initial population (sub deciding on size of that population) deciding Mut/cross rates. Deciding on what Gens to use. & the Gens is Given in the problem desc., but it may need linear translation (e.g. continuous rather than 0 or 1 only!). This extreme case turns a BNV into a 0Z problem.

When we use a P.D. to describe our "Population" we will want old P.D. & perhaps add definitions as in (2.) (.08)

7 The initialization must be "learned" by the system. It notes relations of problem descriptions to the delimitations found most useful in solving that kind of problem. The "kind of problem" can be discovered using the "Compression technique" used by Li, Vitanyi... & others. However, the compression should be better than just BZ2 (PPM): I'd also want Bzip2

0 21 to include sub-tree contexts, & perhaps EF Grammar discovery.

Later, it is not clear that Lsrch is useful in this area! See 2G1.20 to ~285. to Elements for discussion of Lsrch vs. GP

(4) Use of Lsrch for proposing new Cands: Int. system(s) of .06-.21 & P.D. merely describe the Population. Picking the Elements most likely to be in the population doesn't seem like a particular good strategy here. A much better approach is to try to model Fitness $G(\alpha_i)$. (α_i is a cand)

24 25 in a way such that it is easy to list α's of high expected G. One kind of model simply tries to predict G as a function of α_i. The predictor function will have a degree of mean square error & we may want to minimize some Risk like that. This is a special case

26 of the more general $P(G|\alpha_i)$, in which we want to maximize

29 $\alpha p \cdot \prod_i P(G|\alpha_i)$ α_i ranges over known cases. αp is a prop of Ti function P

It uses the prediction function of .24-.28, w. a mean square error criterion, this is equivalent to using a Gaussian Error df. for $P(G|\alpha_i)$ (or P(.29)) and using the criterion of .29 for our "Goodness of fit criterion"

A possible form of $P(G|\alpha_i)$ of $\{G(\alpha_i)$ w. perhaps Gaussian error$\}$ expresses $\chi(G(\alpha_i))$ as equal to the pc output of $\tilde{P}(\alpha_i)$ in some Stochastic Grammar (234.05 -.18)

($\tilde{P}(\alpha_i)$ is a stoch. Grammar).

(5) Both GA & Adaptive Lsrch are unable to watch Cands being tested. They are only told the number Cand passes or fails or what its Error was, but never "why" (see 263.34) Newell/Simon's GPS does look at why & how & Gen work

(6) When the Gen has much CC. Try to Approximate it during early phases of GA-Lsrch.

10:233.40 spec: Hvr, t. Example of 233.08–.11 should give vs prove or vice versa": A P.D. will be always modeling t. population (possibly w. some weighting funct α G or δ(G).). On other hand G or δG(·) will be trying to model some function of t. α, not related to population. Hvr, it is conceivable that t. population & δ(G) would come to some equilibrium eventually, if they were appropriate.

If we added to population via Lsrch, it would certainly Boost t. population a lot!

05 Another approach: We just use t. functional form of a P.D. to approximate δ(G) or G.

This functional form is good because it enables us to do Lsrch a/o select cands of hy G easily.

Given a CFG or CSG (stoch, stoch) one could parse each of several cands & find its pc wrt the Grammar. For continuous params one could use NonLin. optzn to find best values. Hvr, selecting discrete params of Grammar is much harder!

— One could in theory, find best grammar to fit a set of cands. T. Gove for such a run might be closeness of δ(G) or G to the pc's assigned by the Grammar. "Closeness" could be rms or max abs value of difference .... Essentially, it is a "correctability" Problem.

We don't have to use s-grammars to model t. G assoc w. a cand. We may have a way that is slightly expensive to get t. models G of a cand, we then rapidly hunt around to find a cand w. hy expected G, then we can use it as a tool if we have a true Gove that is very expensive.

17 ... If we do find t. cand w. G models hy G, is t. true Gove gives us about t. same G — } this is our way of dealing w. this problem

18 What do we do to get hyer G? We could try to jiggle around that point.

19 One way to get a problistic function of G & $\overset{\text{cand}}{\vec{\alpha}_i}$ (non parametric in the

20 "G" direction, however!). Put t. cands in Lex order as in BW xfm, but also include

21 empirical G at each point. By only using data with G > X_0, we can get

22 a pd. on future cands based only on past data w. G > X_0. } See 239.06 for a V.G. way to do this!

So we can be as Elitist as we like, by chosing hy value for X_0.

The Bad thing about this method: it is "Non parametric" in t. G direction: This is very wasteful of info & limits t. goodness of our models. It may hvr., be good enough for Phase 1.

To get beyond simple Phase 1, we give Phase 1 t. problem of finding good (better)

28 problistic Models for t. corpus (than our current (augmented) PPM). One trouble is that → .37

I don't yet have any good ideas for models to try (other than this "Augmented PPM"). However, it is a well defined OZ problem, so I should be able to devise a TSQ of problems leading to it.

This stuff from (.00) to .30 is important! (.17–.18 / .18–.19) is an impl. criticism of one of t. ideas I had. Bass that was quite good! i.e. t. modeling of true G as a funct of t. cands α_i. That as well as 233.00–.17 + ibid 30–.40 which is on using Problistic Grammars to model G as a funct of α_i (t. cand descrs) — Needs to be carefully reviewed. What t. problem was: & proposed solns. & objections to them.

37 (.28) For each value of X_0 we have a dfrnt p.d. on α_i's & a dfrnt distribution of empirical G. This $\bar{G}$ as a funct of X_0 & σ_G^2 as a funct of X_0: we can use them to decide on a X_0 to use to get good proby of much hyer G in new cand, perhaps chose X_0 ⇒ $\boxed{\bar{G}+\sigma_G \text{ is max}}$ — $\max\left(\frac{\sigma_G}{G_{max} - \bar{G}}\right)$ (thus far) is better: see 239.06

Given several Grammars, t. Goodness of fit G — has hyper value! 239.06 — Hvr. note: "interval" should be G_j − G. G is not what t. empirical population ... G and δG are for each S-grammar, and are not ... but are for S-Grammar itself!

RANT .20

032β

20: : SN On Grammar Models for GA: I think that Recent work on Pristine to Model (The Population as a S-language. My "improvement" (at least) was to model the Gene as an "S-language": We simultaneously have to find a monotonic function betw. the pc assigned to a cand by the S-lang, & the Gene. I.e. say α_i is a cand. $G(\alpha_i)$ is the fitness of α_i. $L(\alpha_i)$ is the pc of α_i as given by S-lang, L.

06 Then we want to find $\gamma(L) \ni \gamma(L(\alpha_i)) \approx G(\alpha_i)$. The data set $[\alpha_i, G(\alpha_i)]$ is used to help find a good γ and L.

08 However, it is conceivable that for certain Genes, 06 is impossible. E.g. say the cands are positive integers. $G(\alpha_i) = 0$ if α_i is even; $G(\alpha_i) = 1$ if α_i is odd. | No! see .11–.17 ?

0 – Since there is an ∞ of even/odd integers, the pc of each cand would have to be ϕ! —
If we had only a finite no of cands possible, say N integers; could assign $\frac{1}{N}$ to even integers & ϕ to odd integers. Then $\gamma L = NL$; so $G(\alpha) = N \cdot \frac{1}{N} = 1$ if α is odd.

11 In an actual GA, this problem would never arise: the SSZ is always finite, so we can always normalize As SSZ ↑ & have constant changes. We do this w. large English corpora as new words enter the corpus. So I think we should have no difficulty devising L & γ & γ-normalizing & truncating functions, will vary to compensate for ↑ in SSZ. (SO) NB Any range of G (−∞,∞ | 0,+∞ | +∞,+∞ | 0, 2π) can be easily mapped onto [0,1]

18. 22-40: Normally in GA the form of Gene (within $\gamma(G)$) can be encoded as "Extremal Contextual Ends"

0: : Note on "MASTER Plan": The idea of the Plan is Merely that I should not get lost. I think I did get lost after Solzhenitsyn & the SAAR work. (?)

The TSCQ should not ever be a Bottleneck! I should be able to write problems in several 'problem-domains in which TM could effectively take advantage of a TSCQ — or something like a TSCQ.

I do also want to review for ① Myself .... to keep me aware of what I've already done.
② To have usable review docs in case of fire or other calamity — also so I can easily take with me, useful, usable bricks for further work, when traveling.

One possible policy: Actually spend at least (1 hr/day on review: 5 pm and 1 hr (min) on project itself. Rest of time on anything that comes up (reviews, projects or housestuff). — 235:00

29

0 :17 Q: If the $\gamma(G)$ xform is related in any way to "linearizing G"? Can a linearized G go from −∞ to +∞? or 0 to +∞? Consider G = Money or Time or "cc".
To deal w. neg utilities: balance them w. positive utilities. So if B is the value of a neg (Bad) thing & G the value of a good thing. $B_1 + G_1 = \phi = B_2 + G_2$ Zero has property that any amount of it is the same. So $B_1 + G_1 = 2B_1 + 2G_1$. So, superficially, utilities may well map well onto probabilities — that have [illegible] domain(?) [0, 1].

A serious "second Plan" on GA in general & My Phase I in general. In GA, modeling the population in a P.D. is just Bad: it models Population, not merely Gene. In my "Phase I" the P.D. models past successful cands. However, I could use a P.D. to model $\gamma(G(\alpha_i))$. → ?

234.00 spec

4TM

: IMP sub. projects: 1) Speed up 2-3 Tree pgm sort works on Cons Computer.
2) (Perhaps) (later.) modify it so I can delete selected items.
3) Get a fast compression pgm: Try it on Eng. text: Compare w. normal PPM
4) Get a GP pgm running. (Note 231.36) – using Lisp/APL/Forth/assembly.
Read stuff on use of Assembly lang in GP.

SN Look at detail structure of Forth & Assembly pgms. What kinds of regs do they have that are common to related pgms? How can these kinds of regs be detected? Can PPM be adapted to do it? Would common sub-tree discovery be useful? Would use of datas (stack PmR) be useful?

SN We can make Monte Carlo from Sorts by randomly choosing previous to hans; We might be able to generate desirably biased Mt-Carlo pgms by randomly choosing sub-trees from the past. We'd have to do Mt-Carlo Lsrch, which is normally unpleasant. We'd have to keep pgms in lexical order or use some kind of Hashing to store them. But, I generally my Mt-Carlo Lsrch did require a lot of RAM (or clever disc Access). Perhaps I could put a normally MtCarlo pgm Generator into Lsrch order!
– [I don't see how! . . .

A trick! If a cand doesn't have a reported "call" within a certain N trials, we drop the cand from Memory. This enables us to only store hi-par pc cands.

SN Two ways to speed up (the effects of) sub-tree srch in PPM. (1) Make Definitions (2) Do ≈.14–.21

For GA: Tournaments: Pick n^2 cands at random from population. Take the n Best of these n^2; remove $n(n-1) = n^2 - n$ of them, Make $n(n-1)$ children by picking random nodes in each of the n "top" and exchanging branches. Replace $n(n-1)$ discarded cands by these children.
The degree of "elitism" ↑ as n ↑. It is most extreme, when $n^2 \equiv$ pop size!
Say one does $n^2 =$ pop size! We continue to run until it appears that the "top n" are not changing. (I forget if it would take an enormous no of trials to be sure this is true!) → (.32)
The problem is that in "top n", some of branches from distinct cands will be the same.
So we might have to check these — if they are the same choose another.
When it is becoming clear that we have "converged", do an exhaustive generation of children of "top n". (25)
How, if each of the n has k branches, that's $nk(k(n-1))$ possible children ($\equiv n^2 k^2 \equiv (nk)^2$ — if not branches over the same).
no. of branches from "other" cands.
— $n^2 k^2$ is pretty a doable number.

Or, perhaps each time one chooses n^2 for tournaments, do all $nk^2(n-1)$ children and keep $n(n-1)$ best of them.

NB Tournament use, unless GA much less scope seems to any γ(Conc) estimate! γ can be used to "roundedness of Conc so it is more continuous, but nothing else!
E.g. f.2 valued Conc of a DNF problem can be "interpolated" for partial credit . . .
In this case f γ(c) is critical. It does when an DNF limits an 02-problem. → (233.17) (Spec)

: ADMIN: Perhaps Make outline of TM project with time goals/deadlines for each Section. As I move thru the project I may be able intelligently modify various deadlines — in view of my experiences.

My Guess is that it would be best to first, make detailed outline w. refs. to how to do each parts & w alternative ways to improve &/o overcome expected BUGS.

Next, Good to get a "more or less working model" of final system — using minimally functional techniques for each section.

Next, Fix various parts — doing the most critical first.

This way I Get F.B. (Both Intellectual, & Political & Emotional) as soon as possible.

: Deficys of GA: ① slow, à ("Pravda" convergence" occasional local extrema) ② Doesn't remember previous problem solns. --- Can't do tsp's ordinarily ③ Population as Model of P.D. --- other Models can be better. ④ Lisch maybe much better than M. Carlo Srch.

By just doing ordinary GA problems, using PPM (perhaps Modified) & Lisch, I should be able to do much better than GA & perhaps be able to do tsp QS by storing Part solns in Rep B-N xform (to Bzz)

SN Tf system I have in mind for induction is oriented toward finding good functions for QA problems. Whether such a system would be good for the kinds of problems that have been addressed by GP ... is unclear. ...

SN2 I didn't analyses of Lisch, vis. GP for the "N bit Multiplexer" (or "N Way AF"). The system was able to learn à (n = 1, 2, 3 ... tsp, w. not much more cc as n↑: But I hadn't tried for recursive solns — that TM should have been able to induce from the successive non-recursive solns. Q: Does the present TM model Automatically normally look at the seqs. of previous QA solns & try to find a (possibly recursive) way to solve "the next QA"?

29 SN Each of $\frac{\Pi_i (n_i - 2_i)!}{(\sum_i (n_i - 2_i))!}$ is rather expensive: I will try various compilers, approxns.

I have to sum over many of these, But in fact, if I start w. the longest contex of size ≥ 1, & work toward shorter contexts ... perhaps I will quickly come to a peak, so I don't have to go much further! A big Q is "How much cc is spent, on average, on this particular Function Eval?" (.29)

36 SN HW/SW Note: If I want to use or Lisp: I can realize Lisp in Forth w. perhaps much ↑ speed: Further ↑ if I use only partly compiled code (in which the no. of "Next" statements is reduced by expressing very short seqs of short code "in line".

Another poss. is to express APL in Forth. [See Advances in Comp. Progg. 1974 Rubinoff ed. pp285 – 324 (2 papers) for Good ideas. I have this Book. Other Good Papers on this Book.

9:10.04
4TM

Spec.
00:227.37 → [sketch] → While the Z141 method is o.k. for normal sequential predn, I don't see how it can be applied to "tree" contexts. In Z141 I could define sequences/ngm & code the corpus using that ngm. In the case of tree contexts I can see how to define them, but I don't see how to code the corpus using them. Well: try this: If the corpus is a set of trees, we can use a particular sub tree that is "part of" "many of those trees", & as a function w. (often) several arguments, & desc. the corpus w. that function as one of the "primitives" — I think that would do it! → (.37)

.08 **SN** On using ideas from old pgms, in new pgms. (TSQ's): One way to do this for a general problem solver (like GA, say) is to transfer a set of definitions used in the old pgm, to the new program. Here we look at our new problem domain; we decide which problems in the past it is "most ~ to". We then transfer the defns used in those problems to the system that is to solve the new problem.

[struck] Other possi. ways to transfer info from one solved probl A to unsolved problem B.
1) Defns. used in A (.08) 2) ngms & subtrees used in A (as in PPM & "tree PPM")
3) General freq. of use of insts (as in OOPS "boosting" extraction) or pairs, triplets, etc of insts.

16 So: A (perhaps) better way to solve TSQ's than what I've been thinking of in the past: Instead of just adding the new problem into the corpus: Analyze it & try to find similarities w. older solved problems: Then use definitions ("tokens") w. assoc frequencies, that were used in solving those problems. If definitions are allowed, transferring freq of use of insts, includes transferring suitable defns. → [The "UNDERSTANDING" can be of any size, Depth, degree, detail, fuzziness, ect. — and of ways to "understand"...

A (sort of) objection to .16 ff: that while having the right ... Always definitions, materializing ~JS of solns of new problems, it's not quite the same as "understanding" why the old defn (function) works, & under just what conds. it should be applied. (.31)

24 **SN** Designing a ckt w. certain specs in as short a time as possi. is a Hutter-type OZ problem. — Yet it looks diffrnt from Hutter's example of Multiplying Matrices. ↑ No!
Actually, .24 looks more like a ENV problem! Similarly, perhaps Mult Matrices is best regarded as a kind of ENV problem. | Hutter's concerned w. a special kind of problem, in which ~ (norms) ... would take as long as cand. generation.
Just as Juergen took a long time to fully understand what Lsrch was, Marcus (& perhaps Juergen) don't yet know what an OZ problem is & how to solve it by Lsrch. The ~ did know how to "sort of" solve OZ problems, it's not clear that he understood why his was by no means an optimum soln.

31 (.23) One thing I'd like the TM to be able to do: It sees a problem solved by "invertible substitution". eg an integration problem. It is then able to genz. the technique to, say soln of linear, quadratic, & cubic eqns. (possibly quartic). More generally, we could get it to learn (or "inform") the ~: "try to genz. the soln. technique of any problem you solve or see anyone else solve."

37: (.07) → Also relevant: It may be good to use both Definitions of functions and PPM (either with or w.o. subtree contexts) Doing an just regular PPM would be very fast & perhaps V.G.! Defns would involve "new symbols" — which PPM is able to deal with!

4TM

.00:228.40: Li-Vitanyi use of Compression algms (just Bzz!) would seem to work a Bit in the problem,

.01 & it's very cheap. Cleverer Compression using subtrees, should do even better. (20)

228.34 ff Gives a slightly dfrnt view of the TM project! At the beginning, I can simply try to improve GA, in all of the 6 aspects. Later I can concentrate on (2) (TSQ's).

.05 SN Note that trying to model the original Gore as a P.D. does of necessity involve modeling of the Gore by a monotonic function. .055 This is in addition to extrapolating the Gore or otherwise modifying it so (a) it has higher "resoln" (>> 2 levels) (b) is more amenable to the GA idea.

In recent work on improvement on GA, one approach was representing the population by a p.d. This seems dfrnt from approximating the Gore by a P.D. (as .055) — but it may be that it is not. Perhaps the difference is: In the "recent work" they use a non-fuzzy population, (Gore info is simply (yes/no) — deterministic → 1 bit.) and they extrapolate it to make it fuzzy (probabilistic). In my formulation, I try to model the possibly entire population with its assigned Gore values, by a p.d. whose value = γ(Gore) where γ is some monotonic function. This uses the data better & is likely to give better results.

.10 (.01): Another thing we get from these Compression Algms: They tell us how to make/create new PSMs a/o initializns. The problem domain is α; the PSM's/initializns β. From a given member of β, we find a set of α members for which that β worked well. We can use this subset of α as a corpus to compress. Any new problem under (α^{14}) α_0 can be added to this subset & we can see how much extra info is needed to compress the augmented subset of α. Moreover, this "extra info" tells how close our new problem is to that class (in α) of old problems for which the β element worked.

By looking at other members of β & sets of prob domains for which they have worked, we can list the members of β in order of how likely they are to work on α_0. Also, we can use the compression Algms to generate new β members that have high likelihood of solving α_0. (possibly states fuzzy MT for large Q to A mapping, mapping.)

.10 .20 ff needs to be worked out more carefully, and integrated w. my ideas of factoring a good set of PSM's/initializations and getting the M. how to correlate the PSMs/initlzns to problems, as well as generate new (PSMs/initlzns) for new problems.

I want to write a "sort of proposal" for a sequence of projects terminating in a V.G. General problem Solver. Possibly get sponsorship thru IDSIA or (R. Holloway?) or ..... get job on staff. Another possy would be to start a co. & get "Venture Capital" (David S. J. Kotz, Sol Friedman [illegible]) Friedkin??? Brewster? Maybe not good to get "stuck" in USA!

4 TM

20: 227.40 : So! a Bit of JOY! 227.00 – .40 ; (227.10 – .24 in particular) seems to clear up a lot of all of my uncertainty of how to assign pc's in PPM & Tree contexts:
Also it suggests that Z141 is probly basicly better.

A possl. advantage of PPM & t. Burrows Wheeler method (BZ2) — That it could be much faster than Z141. If this is true, I may want to trade speed for ↑ hy pc's — & possibly for tree contexts : — but look into that later!

Another form is 226.40 – .23 which uses sticks closer to Z141 & doesn't exactly use "Context".

But do write a detaild dscn of 227.00 – .40 : As is, its kind of vague to a person reading it 1 yr from now (or maybe (MonR1)). → (.20)

0 Much of PPM is involvd w. deciding what to do when a "new symbol" occurs. In Z141, I didn't address this problem at all! In much (or all) of what I've [illegible] that about this does not occur (except, perhaps in Laplace's rule). T. PPM analysis suggests how to use Laps rule to deal w. continually occuring "new symbols" — i.e. At all times, when one has a corpus of n symbols thus far, there is a pc of $\frac{1}{n}$ of a new symbol. T. total pc of this feature is $\approx (1-\frac{1}{2})(1-\frac{1}{3})(1-\frac{1}{4})\cdots(1-\frac{1}{n}) = (\frac{1}{2}\cdot\frac{2}{3}\cdot\frac{3}{4}\cdot\cdots\cdot\frac{n-1}{n}) = \frac{1}{n}$.
— But it occurs for alternative codes, so it sort of cancels out ... The codes w. fewer → symbols for t. whole corpus would get more wt.

10 (.10) Understanding t. relationship of "Classical" PPM & Z141 is important because I want to be able to map t. "improvements" in Classical PPM into Z141.
Many of t. features of PPM are empirical approxns/averages over Z141 values. These approxns can save lots of cc & even tho t. pc values of PPM may be worse than those of Z141, t. cc saved may make PPM superior! Hvr. for Expensive fitness functions, the better pc's of Z141 & of "subtree finding" will prevail.

26 SN One of BIG diftys in adapting GA to a problem is Modfn. of t. fitness function :
In the defn. of t. problem, a t. fitness funct. will always be given. In INV problems, it is initially a Most function in which GA can sometimes be used, but almost always, a better Gone (f-function) must be devised. For OZ probs, t. Gone must also be modified. If we use a S-lang. to model the Gone, we have to modify t. Gone by some monotonic function.
0 If the original Gone is very expensive, we want to replace it by a much less expensive approxn., at t. beginning of t. "Hillclimb", & possibly other modifns during t. climb".

When we get to "phase 2" presumably, modifn. of t Gone is unnec — But I'll probly want to look into this in some detail.

34 So : GA, GP has 4 impt. diftys: We add a new one in (.26 ff) : 5 diftys:
1) GA is slow 2) GA does not remember & use ideas/solns of previous problems (No TSQ ability)
3) T. initialization of t. RUN is Big Task. Getting initial population, Lang. to describe cands, Mut/cross rules. 4) To initialization we add Modfn. of t. Gone (≡ fitness funct) (.26) ← which is really part of (3) (initialization)
(4)
5) It should use Lsrch rather than Mt. Carlo.
In all of these initializn problems, we "look at t. problem domain" & propose initializations. → see 229.00

9/8/04

4TM

.00: 225.40 : T. Q's: t. Legitimacy of t. codes for PPM of 223.32–224.20 (+ 224.21–225.40)

.06 & codes for Trees : of 220.16–19, .30–.40 + 224.28–.39 + 2.23.24.

Any algm. that gives a pd for each "next token" in t. corpus, is a legit code & a legit PEM.

At present time : I think I understand why both t. PPM code & t. Tree code are legit:

T. PPM code is legit because : At each pt. of Token predn, one has several contexts to base t. predn on. We can give each method a wt. A reasonable wt. is t. effectiveness in coding t. past corpus of that context (& context). Since we only have k dif. contexts to chose from, we can legitimately wt. them w. some function (≈ a pvip) on t. nos. 1 thru k — an optimization of t. form of Restriction on the past of t. time series.

The form can be anything we find empirically "Good": Hvr. since we know Pencidk we can use ≈ $A\beta^{-k}$ in which β is t. card of t. alphabet & A is a normn constant.

12 If $A\beta^{-k}$ is not so bad, this corresponds to t. pc of defining each context.

13 More exactly, the pc cost should be $\propto \prod_{i=1}^{k} f_i$ where f_i is t. frequency in t. corpus (up to now),

14 of t. i th symbol in t. context. We get this by simple Z141 type coding.

15 On t. other hand, we get ≈ same result by considering that a context that is (made (or concatenated with.) t. symbol "s") longer by t. symbol "s", will have an "escape cost" (in pc) of t. frequency of "s" in

17 t. corpus thus far.

.18 Hvr, t. args. of .06 seem backwards : Because in normal PPM, they start out on t. longest possl. context, then escape down to one that can predict t. next symbol. (i.e. its pc for next symbol is >0). On t. other hand in Normal PPM, they are burdened w. t. additional pc of escape: This is always an additional symbol in t. alphabet. → (Woops!) Note .38

.21 → Also, PPM doesn't multiply by t. pcost of α & ᾱ — so perhaps normal PPM should be thot of as a rather rough guess at which context to use. — (I think t pc of α & ᾱ should be very impt!)

.24 So .12–.14 (which is ≈ Z141) may be v.g. & perhaps significantly better than PPM. — This appears to be MAIN advantage of Z141 over PPM

Note 226.16–.23 on using ≈ Z141.

So: Z141 would seem to give a good rigorous model — a way of understanding how to improve PPM.

It does seem to give a very reasonable way to do my modifn. of PPM.

It gets rid of t. (tome) confusing idea of coding by means of postfixes — since in that case, the entire (ll) codes of t corpus are not stated. My more recent (.10–.24) analysis makes it possl. to get ≈ & probably much better than PPM using the normal ll codes & pc's of definitions, of Z141.

33 Also, it clarifies t. assignment of pc's to tree contexts. We effectively have a pc of defining each tree used for predn. T. Nov will give no. of symbols, and β^{-Nov} (β = card. of alphabet) is an approxn. $\prod_{i=1}^{k} f_i$ (in .13) which we use as

.37 a suitable normn factor to get a good pd of t. various subtrees used as "context". → Diffy In trees, hvr → 230.00

.38 (.21) In normal Z141 we also have to add a new symbol to t. alphabet — namely α — & an assoc pc for it. This does correspond to t. pc of "escape"! Whether it does so quantitatively is unclear.

0: 225.40 : In view of 225.35 – .40: Perhaps drop this problem & return to Tree Compression coding.

.1 T. "Soln" Pro ~~term of~~ PPM is 223.32 – 224.020: (w/obj'ections on 224.21 – 225.40). (possl.)

.2 T. Soln to Tree coding w. PPM: I think ~~[illegible]~~ Bibl. outline is on 222.25 – .36 i.e.
i.e. ⟨220.16 – .19 : 220.30 – .40⟩ (Main discussn.) ~~[illegible]~~ 221.28 – .29. (How to get Apri of Nov.) (Diffty of 221.30 – 33 is resolved on 223.24)

---

0 **SN** **IMPT!** I have been using Contexts to do predn. This is in t. spirit of MDL, hvr, & not much Good for OSL (≡ One Shot Lrng). In PPM (& prob'ly in Tree version of PPM), we can easily modify our techniques by simply considering alternative continuations of t. corpus, & ~~each~~ comparing t. pc's of these

3 alternative corpus continuations! .133 Here, we do not consider contexts as ~~having~~ inducing proby distributions on t. following Tokens. The calcns may be slightly simpler than in t. usual PPM way.

Hvr, t. Mechanics of .133 ~~[illegible]~~ — how we do predns — is not yet clear!

One way: ~~[illegible]~~ An ~~[illegible]~~ item of length k includes last k ~~+1~~ symbols of known corpus plus 1 / symbol to follow. (predicted)

We express t. corpus as a .... No!

I think I would have to use t. method of AZ 141 in my G. Wolff letter.

Perhaps use ~~it~~ Both and t. previous version PPM for comparison.

(See 227.00 – 227.40 (22 lines) for a continued discussion of Z141 v.s. PPM. & their close relationship)

~~[illegible]~~ BIG advantage of t. AZ140 method is that it enables (eventually) ~~[illegible]~~ many definitions & reparsings

0 Many reparsings of corpus & definitions of Aruses → PSG discy.

~~[illegible]~~ One possl. advantage over other methods is that we just use BW r.t.u. to ~~[illegible]~~ find good items as in ~~[illegible]~~ usual PPM (i.e. ordinary PPM) — Then common subtrees are then discovered (probably) by PSG discovery

3 techniques — that don't ordinarily involve direct search for sub-trees. This may be much less expensive.

SM Consider ~~[illegible]~~ pc of corpus if context of length k was used every time (as a fn of k).
This should give t. P(k) — rel. apri wt. for contexts of length k. We have to be sure that each context gives pc > 0 for every symbol (viz Laplace or Dirich).

9/4/04

4TM

.00 Perhaps express it as a sum! Say we have a specific set of $[m_i]_{i=1}^{k}$, $\sum m_i \le \sum n_i$, $i = 1 \dots k$. get the redundancy ($\equiv$ "degeneracy") of wt. of then sum them. Maybe simplifyable by factoring, by various identity, approxs, etc.

.03 For a given m_i, n_i, there are $\frac{n_i!}{m_i!(n_i-m_i)!}$ ways to chose m_i subjects from a set of n_i. $\equiv \binom{n_i}{m_i}$

.04 So for entire set of symbols: $\prod_{i=1}^{k}\binom{n_i}{m_i}$ is no. of possl. ways. $\equiv$ Degeneracy $\equiv$ Redundancy

Say N_i are the cose counts for the entire corpus;

.06 For the α corpus, the pc assigned will be $\frac{\prod m_i!}{(\sum m_i)!}$ ← Null (for precorpus)

.07 For the $\bar\alpha$ " the pc assigned will be $\frac{\prod (N_i-m_i)!}{(\sum(N_i-m_i))!} = \frac{\prod (N_i-m_i)!}{(\sum N_i - \sum m_i)!}$

Reciprocal of $\left[\binom{x}{y}\right]^{-1} = 1/\binom{x}{y}$

.10 Total wt for $[m_i]_1^k$: = $(.04)\cdot(.06)\cdot(.07)$. → Maybe .04 will cancel many factors(?)

.11 The D.F. bag wd will be $\frac{m_j+1}{(\sum_i m_i)+1}$ ← is this for null precorpus?

So, the rel pc. of symbol j will be

.14 $$2^{-\sum_{i=1}^{k} n_i}\cdot\sum_{m_1=0}^{n_1}\sum_{m_2=0}^{n_2}\cdots\sum_{m_k=0}^{n_k}\left(\prod_{i=1}^{k}\frac{n_i!}{m_i!(n_i-m_i)!}\right)\frac{\prod m_i!}{(\sum m_i)!}\cdot\frac{\prod(N_i-m_i)!}{(\sum N_i-\sum m_i)!}\cdot\frac{m_j+1}{\left(\sum_{i=1}^{k} m_i\right)+1}$$

$2^{-\sum n_i}$ $\equiv$ cost of defining each code. See .30

(.04): Wt. $\equiv$ redundancy; Also this $\equiv \prod_{j=1}^{k}\left(\sum_j\right)$ of (.31) $\equiv (1+1)^{\sum n_j}$

(.06) α; (.07) $\bar\alpha$; (.11)

.18 The $\bar\alpha$ factors mite be simplified by simply considering the no. of symbols it has to code & using mean entropy for the entire corpus: mult. by the fraction of symbols $\bar\alpha$ has to code.

.20 Other factors mite be approximated by Gaussian, Binomial, or Gamma distributions.

(.04) $\equiv$ wt.; could be a 1 dim Gaussian

.22 $\overset{\alpha}{.06}$ & $\overset{\bar\alpha}{.07}$ could be multidim Gaussn or Binomial dists

Otherwise it may be best to neglect these 11 codes! However, I really don't know if they can be neglected w. impunity!

It mite be best to drop that "trim" for a while! Just see it works well w.o. it in PRTY. The real improvements that are probly more worth spending time on is the search for common subtrees. Having found some useful context subtrees, we can also try them as cands for 60 terms (to be predicted).

.29 The "total wt" portion .14 is $\sum\cdots\sum\prod^{k}\frac{n_i!}{m_i!(n_i-m_i)!}$ and this should sum to $2^{\sum n_i}$ or $2^{\text{some sum}}$, since we divide .14 by $2^{(\text{some sum})}$ because that's the pc of decn of each of the terms.

.30 "Some sum" $\equiv \sum_{j=1}^{k} n_j$

.31 $\left(\sum_j\right) \equiv \sum_{m_j=0}^{n_j}\frac{n_j!}{m_j!(n_j-m_j)!} = (1+1)^{n_j}$ So I guess .29 is the product of the factors (.31 L) $=\prod_{j=1}^{k}\left(\sum_j\right) = (1+1)^{\sum n_j}$

The effective wt. of .14 is one code (It's the wtd mean of $(1+1)^{\sum n_j}$ codes), the

.35 Q is, How does it compare w. other codes — (Having an a priori wt of "1")? We can use the approximations of .18 – .22 to get estimates.

Another way to try to answer .35: The codes of .14 are all codes of subsets of α. It maybe that they always get << pc than the single direct code for α.

226.00

and 227.00 for → spec. Corry matter

4TM

10: SN Just how does the precorpus entering into the coding of the sub corpi? Is the precorpus shared by α & $\overline{\alpha}$? If so, how is this done?

View ①: That α & $\overline{\alpha}$ are represented separate corpi: each has its own precorpus.
View ② (Occam): That the whole $\overline{\alpha}+\alpha$ corpus has this precorpus — so α & $\overline{\alpha}$ interact somehow in using it.

Since α can be a rather small subcorpus, the precorpus can be a big deal. We may want it to only have $\frac{1}{2}$ a symbol for each of the alphabet — (which is a common typecast Dirichlet distribn.)

07: 223.40 So: we code the "entire corpus so far" by assigning its symbols to either α or $\overline{\alpha}$, and we obtain a pc for it: $P_\alpha \cdot P_{\overline{\alpha}}$. Say this context is of length ℓ. Then we have $P_{\alpha_\ell} \cdot P_{\overline{\alpha}_\ell}$ as the pc of the corpus wrt $\alpha, \overline{\alpha}$: The prob distribn over the Tokens to follow, will be those of α_ℓ.

Each of the contexts α_ℓ ($\ell = 1 \ldots n$) (length of corpus thus far) will have a wt. $P_{\alpha_\ell} \cdot P_{\overline{\alpha}_\ell}$ assoc w. its D.F. on the next Token. The final predn. will be a wtd. mean of all of the D.F.'s assoc w. all of the contexts. The wt. will be $P(\ell) \cdot P_{\alpha_\ell} \cdot P_{\overline{\alpha}_\ell}$. $P(\ell)$ is the apriori wt. assigned to contexts of length ℓ.

13 $P(\ell)$ can be of various forms used to assign apriority to integers (see 220.30-.39) — but ultimately the form of $P(\ell)$ will be empirically adjusted so that in the past the pc of the corpus had been

15 maximized (see 221.28-.29) — re: the f.n. of (221.24-.27)R ← my bias is to start the apris at $\ell = 0$ .... rather than pick the longest ℓ w. cnts > 0 & use it for that listing — but I think this has to be empirically determined — which assignments method gives the highest pc to PCKT corpus: (Note $\Sigma_\ell P(\ell)$ being also a function of: n, the length of corpus to date)

} see 227.18-.24 for an insightful discussn of this Q. !

18 So: 223.31 — 224.18 is a complete down of how to do PPM — except for recommendations of 224.00-.06 and 224.15-.18: also the diddy of (.21)

Of course there is the Q of how best to realize or approximate the technique described.

21 SN Note $P(\ell)$ of (.13) will be a funct of n (the length of corpus to date): If n is small, then large values of ℓ can't have large P's. Perhaps $P\left(\frac{\ell}{n}\right)$? Perhaps assume functional form: $P(\ell, n) = P_1(\ell) \cdot P_2(n)$?

SN While 223.31-224.18 may be a good Bayesian approach, the recent "improvements" on PPM may be expressable in Bayesian form & be integrated into my approach — so I can optimally incorporate the recent improvements into my Model.

SN A possl Theoretical Criticism of the entire PPM approach is we can gain lot more

11 codes by treating some of the α cases as $\overline{\alpha}$ cases. This was done in my "Letter to Gerry Wolff, on ~ 2141. ← see Folder ΦΦTM 2.141 (2000 yr) for hardcopy letters + corrections.

One Q is whether we have to say which parsing of the corpus we're using. If we use a parallel parsing then the tally which one we use very very big pc = ∞. This seems correct: So, given the compressed case counts & the precorpus case counts, one should be able to calculate the distribution for the next token etc.

Considering every subset of α as a possl subcorpus: code. If α has k symbols omit those that are just 2^k distinct sub corpi (including null sub corpus). Each sub-corpus & β has its assoc $\overline{\beta}$, so code the entire corpus.

Ominity Considerations of Precorpus: If $n_i]_{i=1}^{k}$ are the case counts of the k symbols (types) in α, — then the subcorpi will have between 0 to n_i case counts of symbol i.

9/2/04 ~ Dyno. #246
4TM

00: 222.40 : T. point is, t. thing I derb is a legit (PEM / pdn. method): for each prediction, I use a function on N_{ov} ($\equiv P_{Nov}$) to assign pc's to sets of Pems. (Each N_{ov} has a diffnt set of PEMs). Assoc w.
02 each Nov is a "single event PEM," that assigns Pc's to "next token" on t. basis of previous known corpus.
03 The "PEM of .02 consists of a bunch of sub-PEMs in ll. They are t. standard α, $\bar{\alpha}$ pems assoc w. Lap's rule
04 (or Dirichlet's rule). T. sub. Pems of .03 can be anything I like. I could assign one to every possl. context including
05 → contexts that do not occur at that point) ← Hvr, I chose to not include those contexts. If I did include them, it would seem to (be close / or identical) to get consider. context ... but this is not exactly true; only the next occurs.
07 [redacted] — each context has a α, $\bar{\alpha}$ and: (a pc for corpus) = wt. and [redacted] a pd. over t. next token to occur.
— But I think we want to only consider P.d's assoc w. contexts that did just occur. — If we did not, we would be summing over almost same set of PD's — indep. of t. current context.
10 Now say we do .03 & include just Pems assoc w. contexts that just occurd & we use their assoc wts (.07)
~~This do~~ We then sum over all N_{ov}s (sum since total P_{Nov}'s sum to 1). This is a legit. P.E.M, since
12 it assigns an (already normzd) pc to each possl. "next token". — T. Q is is it any good?
13 To better legitimize t. dern. of (.00 – .12) express it as ll codes of t. corpus.
— Well, any PEM has an assoc. code, so ~~this~~ the (.00 – .12) is legit to that extent.
15 → T. Q's are: ① are t. codes dcs finite? ② Are they a (reasonable / good) approxn to t. Dmrl. dist.?
Well according to .13 – .15 we would then want to include all of t. ll contexts (.04 – .07) (most of which did not just occur).
→ Hvr, all those "ran dead out" pems are (identical / t. same) ... its not clear that they have diffrnt codes.
i.e. if pc's are diffrnt, they have to have diffrnt codes: If pc's are same, they may or they not. diffrnt codes.
19 For each context (any derivable context), there exists a α, $\bar{\alpha}$ code to t. entire corpus: Since
20 all these context derns are diffrnt, they correspond to diffrnt codes! Hvr, if we define contexts that are not occurring now, they are very expensive to describe. To describe a current context, you have to tell where t. symbols are (2 dimensions): For non-current contexts you have to tell position & Times identifier & types of all of t. symbols — which lead is r cost of $(k)^{Nov}$ (k = no. of diffrnt symbols).
24 → So, for non-current contexts: If there are N of them, they have each a pc of $\frac{1}{N}$ so total wt is 2^{-w} (only.
So: (24) very solvent! All of the non-current contexts have total wt. of 1 so we can probly simply disregard them, since each current code has a wt. of
To get pc's of current contexts, for each N_{ov}, ~~and no. of symbols~~ (indep of No. of symbols), (depending on functions used & their "arity"), there is a number of possl. contexts. (see 220.15 for a few values for binary functs only) — we can use the reciprocal of this no. to get pc of each
30 current context.
→ But see 222.00 – .40 for impl., now corpus ideas on its relation to Z141.
31 So: "FINAL soln." to context weighting for Sequencial (≈PPM) corpus & for prdn in which contexts are sub-trees. → **223.32 — 224.20** → See possl. improvement(s) of 224.21 — 225.40
32 ① Sequencial corpus: (≈PPM): ~~However~~ At each point in t. corpus, we have a nesting (overlapping) sequence of contexts (≡ post fixes) that extend ultimately to t. first symbol in t. corpus.
For a corpus of length n, there are n contexts, Assoc. w. each context, there are 2 sub corpi:
ⓐ Tokens preceded by context ⓑ rest of corpus in which tokens were not preceded by context.
α $\bar{\alpha}$
α & $\bar{\alpha}$ have diffrnt case counts for each of t. symbols; Using these case counts, we can use Lap or Dirichlet to assign pc's to each of these sub-corpi ⇒ P_{α} & $P_{\bar{\alpha}}$ { Note 224.00 – .06 on t. Lap – Dirichlet diffy } → 224.07

9/1/04

4TM

00 : So ① Describe problem exactly ② Give possl. solns. ③ Tell what's wrong w. possl. solns. | Bad : √ (Min x)

④ List alternatives w. pro-con Argts. ⑤ Is there a way to avoid the problem?

221.30 seems to be the counterargt.

Perhaps list some good features of proposed soln.

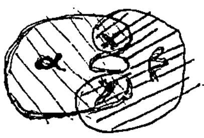
Consider case of ~~nested~~ sub-corpi (sequential predn., standard PPM).

Since each "deeper nesting" is a smaller, more restricted sub-corpus, each such nesting costs more (bcost) to derb. ∴ lower a priori. Tho, in the present case, third cost seems much smaller than usual! To derb. a 10 bit prefix requires at most $\log 10 + \log\log 10 + \log\log\log 10 \ldots$ bits, rather than 10 bits. (i.e. $k(10)$ [at least]).

i.e. $k(10)$ v.s. $k(2^{10})$. A similar economy occurs in ~~[struck]~~ 'sub-tree descn' (in the "tree-compression" problem)... We just have to give the positions of the symbols in the contexts, not their values — in the "nested" case, this is particularly "cheap." — Th. contexts themselves, hvr, are not "nested". | 1010 = 10. 4+3+e(1)? = 10?

10

12; 221.40 Perhaps consider the more general case of several corpi that have some, but not all, elements in common.



So corpi α & β overlap at x & y. Ideally, to do induction on x & y, just consider the total corpus.

.15 This approach doesn't seem to help much in sequential PPM, since / in that case, α overlaps β completely!

First note that as ssz ↓, the distribution need not get narrower, tho it could: [i.e. contexts gets "narrower"]

Case ①: it gets narrower: Symbols are A B C. narrow corpus has 10 cases of A predicted; no others.
Broader corpus has 10A, 10B, 10C. Broader predn.

Case ②: narrow corpus has 1a, 1b, 1c. Broader corpus has 100a, 1b, 1c, better pred.

Possl. [predn. rules] pick context w. smallest varc? or combine contexts w. wts = $\frac{1}{\text{varc.}} = \frac{1}{\sigma^2}$.

Tho. ssz would seem to be very impt!

25 Another possy is the α, ᾱ approach at 220.03, but if N is length of longest context, (so there are only N contexts); one uses these N contexts in ||, w. wt. ∝ pc of corpus wrt each α, ᾱ.

26 While .25–.26 seems not too bad for sequential predn. for the "subtree" contexts we have a much larger set of () codes — But for reasonably sized corpus & reasonable compression,

The many codes that we don't use except contexts may have small wts. (?) The actually, the no. of contexts that have never occurred, is really enormous! So for the time being, just consider contexts that have occurred, that have a string of symbols linking them to the token to be predicted.

30

So this is essentially the ~~[struck]~~ soln. of 220.16–120 & 220.30–.40 & (221.28–.30)

→ See exactly why I discarded it the first time. I think it was 221.30–.33

The idea there was that the null context was by far the commonest context — so if we summed ~~[struck]~~ the PD's of all contexts in ||, the null context would win because it had such high redundancy.

.38 A possl. soln. is that the null context only occurs once. } → See 223.24 for soln. to this difficulty

See if this soln. applies well to ~~tree~~ induction. If we only consider contexts w. one or ~~[struck]~~ more cases, it may work! Also think about (.12–.15)

But it looks like ~~[struck]~~ (.12–.15) is a nice approach to the problem described there!

.10 : Hvr, in t. seq. predn., The overlap properties are "special" & may bear additional

.32 Scrolling! (TT)

[SN] Can I adapt lossy Methods to N dimensional Images?

[SN] How does lossy (≈ Dictionary) method compare to other compression methods?!

For first char of sentence we use char freq. from large English corpus.
" second " " " " " 2gm " " " " " (for first 2 chars)
" third " " " " " 3gm "

for forth & subsq., we used dictionary to narrow possys, Then 2gm frequency or 3gm freq's.

[SSN] For comparing speed of 2800 Pent vs. 500 Pent., use 32bit compiler! See if I can get some insts in "Consol Compiler" as in regular Power Basic Compiler.

1.02 Consider sequential Case, P_{Nov} is what?! (see 220.39 for reasonable defn.) If we make actual preds for only 1 context each time, the P_{Nov} is t. long term past pc of t. choice of that particular context.

Consider 2 solns soln: 1: if t. token that will occur has never occurred, we give zero pc. & use a diffrnt predn. method.

Soln. 2. In all cases, all possl. tokens are implicitly considered. If N tokens have occurred up to now, pc 1/N is always reserved for new tokens. (This is usually "wasted"). Using Laplaces rule we/here always precorpus of 1 case count for symbols that have not yet occurred.

Soln 2 will always give a true predn of pc > 1 for any future symbol.

.19 Soln 1 will give pc = 0 for a prediction, but this will occur very infreqly for a long corpus.

0 T. discussion of .11–.19 is about "escape" Problem: It will not occur in certain kinds of corpi — in which we a priori know t. alphabet (say for certain English texts), or when we know set of primitive elmts & no definitions are used. Let us discuss this case first (i.e. no escape problem).

.24 — So we have many contexts; each can be used to define (make) a code for t. corpus. We want to use these coding methods in 11: how best to assign aprips to them?

[In all cases of predn methods, we have to assign an apri to each method, & this

.27 apri will be somehow related to more dirctly past of t. corpus.

.26 In the (default case / worst case) (Bad in t. sense of available info), we use t. language of discourse to assign aprips. Using data from t. past, we assign aprips so that pc of corpus is max. {Note footnote of (.24–.27) R} — This can influence apriod much — We may

.29 — want to try both ways.

We can start by listing contexts from $N_{OU}=0$ — Presumably here Nou's; pc/Ortw. Nesmum; Nou available & worth down to Nou = 0. See discussn of 224.15 / 227.18–.24 is more insightful!

.30 → Alt, perhaps use all contexts for all predns! (in theory, probs) Once uses either α or $\bar{\alpha}$ each time. — This is o.k. — Seems to cause troble but fixes it — A trouble, perhaps! The many $\bar{\alpha}$'s will sum to much wt.! Hvr, $\bar{\alpha}$'s overlap a lot!! See 223.24 for resoln. of this diffy

In view of .30, I'm beginning to be uncertain of this present "Break Thru" An impt feature of

.33 "Context" is that it works well in a particular situation — not merely t. whole corpus.

If we have 2 predn systems: each w. mainly its own corpus, but some overlap: How do we wt. t. 2 systems when they both are usable.? The normal sequential PPM system is a peculiar one in which t. corpi always include one another! → 222.12

4PM

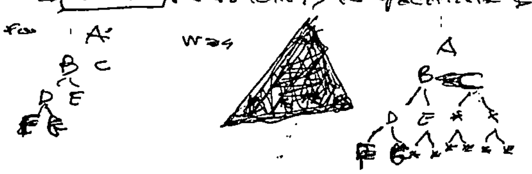
10 : 219.20ff à .31–.32 do point out how PPM (for sequencial) à for Tree predn. ought to be ideally done. | Dirichlett d.f.

I'm not so sure that t. alternative parsings of 219.31–.32 are relevant to 219.21ff.

.03 Say one defines a context, α, (for "t. corpus up to now"). This automatically defines $\bar{\alpha}$, t. complement of α: whenever α doesn't occur, $\bar{\alpha}$ occurs.
So defining α ($\alpha\ \bar{\alpha}$) gives us 2 sets of case counts, à each case count can be assigned a pc viz Laplace or Dirichlet d.f. (See 219.31–.32 for exactly.). T. pc of corpus is product of 3 things! ① pc of defn ② pc assoc w. case counts ($\bar{\alpha}$) ③ … This product of 3 pc's gives t. wt. of pt.
.08 predn of context α ($\alpha\ \bar{\alpha}$). (.13)
So a big Q is how to get pc of defining α (A good soln. of this problem: 223.24)

0 An assoc. Q: Given a corpus, what does it mean to ask for pd of next token"? what is criterion of "correctness"?
One answer is that t. present known corpus is a basis for t. a priori pd of its contents.

3: (.08) For .03 ff. Think about sequencial (non-tree) coding to start.
Specifically at each pt in t. corpus parsing, for each context size (corresponding to N_{ov}) there is only one possible context. For all other contexts, α_i we use $\bar{\alpha}_i$ at this point, and $\bar{\alpha}$ is t. "no context" case, in which t. pc's of t. tokens are simply their (total) raw/case counts in t. corpus – thus far". This is t. "default d.f." — pure Bernoulli (Laplace / Dirichlet d.f.). For each "N_{ov}" there is an assoc no. of possl contexts. — for seq. coding it is $K^{N_{ov}}$ (K = no. Tokens). (for Tree coding No of contexts K^1 $2\cdot K^2$ $5K^3$ $12K^4$) [N_{ov} = 1 2 3 4]
.15 16 My impression of soln. Say $P_{N_{ov}}$ is t. a priori assigned to $N_{ov} = k$.
Then for all t. contexts w. of that N_{ov}, one has a code for corpus (viz α à $\bar{\alpha}$). We take t. "pc of corpus"
wtd sum of all those contexts à this is assigned (multiply) $P_{N_{ov}}$. — This result is indep of
9 how many contexts (of $N_{ov} = k$) one sums — t. more, t. better, — The usually says a few how much wt (.30)
0 SN Some new thoughts on notation for functions to facilitate subtree search!
Say t. notation for 'A' was … where * means Null.



8+4

Notation 2.5 summary:

So this subtree occupies much more RAM than strictly needed, (A tree of depth d, needs 2^d bytes!)
So: t. string , A|B C| DE 00 |FG 00000| .. — points."
The value of worked many is that it makes it easier to get N_{ov} of 2 d[ifferent] symbols in th corpus.

0 (.19): The idea is: for each N_{ov} value, we have a prediction à wt. à apriori $P_{N_{ov}}$ • It looks like a good soln. to t. problem. I'd like to get my mind in a state so that t. soln. seems obvious — so I would automatically suitably Gen it. [d.f. on tokens; à (pc of corpus w.r.t α à $\bar{\alpha}$.)]
$P_{N_{ov}}$ (P as a funct of k) should be obtained weakly empirically: Some possl forms are t. a pri pd of t. integer (= overlap number) (Rissanen approx., for reals); (Rissl approx.), my approx for integers only, Normalized $\frac{1}{K^\gamma}$ choose γ; ($\frac{1}{K}$ Normalized: finite no. of values of k used) } This presupposes t. form of many of t. "kernel" I've used for PPM

SO .16 – .19, .30ff seems to solve t. PPM for seq. à trees Theoretically: Also clarifies exact form of t. kernel, for both cases:
39 [Actually, t. exact functional form used should depend on our experience in t. past. T. exact function that relates Proby to N_{ov} is selected so as to Max pc of past corpus. (See 221.28)

8/29/04 2.19

4TM N_{ov} ≡ OVN might be better to use number.

Def: N_{ov} ≡ ovn. N_{ov} ≡ OVN ≡ overlap number. N_{ov}

10:218.20 : [redacted] On pred on Basis of "subtree overlap no.". I had that that I would "pool" all data from a given "overlap no." — But this is not so good. It is likely that certain subtrees would be v. good for defining & predn & others not much: "OVN" (≡ overlap no.) could be an index of pc of defining subtree.

Def Also an index of how much benefit is saved by defining it. So say we assign pc = pc to overlap trees of OVN = k

..Hvr, for coding, we have to specify OVN, then which of the many possible subtrees of that OVN, we want to use for predn. In many versions of PPN of sequential corpus, one predicted token is an automatic way to choose t. contexts. In ours, it's t. longest context that correctly predicted t. actual following token. Instead of picking one context, one can use all contexts & wt. them — so no "best choice" need be expanded. This is t. method I was planning to use for sequential corpus using a "kernel" to rapidly implement this [redacted] predn routine. (My memory is a bit vague on this; I better make sure (soon) that I write it up. a/o hours old biblio references)

0 T. way t. sequential prediction worked; we did prediction in t. L domain (Lexically ordered contexts). T. predn was a wtd mean of preds of contexts on both sides of t. "present" context. ← seems most impt. this seems irrelevant. Wts could be a function of (either or both) ① length of common match ② How far away along L t. assoc. context occurred. T. wts would be a funct of what t. predicted symbol was. But not merely t. actual symbol that occurred). ← [I'm not sure this last dependency is nec].

For each token, there is a 0,1 sequence along L, telling whether that token occurred at each pt.

" " " we have a smoothed prob'ty of 0 or 1 being at that pt. We have a "smoothing kernel" to get these pc's.

→ For each entire (actual) context we have an assoc prob'ty vector for t. possible predicted tokens.

②

Consider sub-tree contexts: Assume t. N_{ov} OVN, t. no. of distinct contexts will first ↑ then fictitious then → 0. Hvr, t. ssz's for t. contexts w. small N_{ov} will be very large & tend to ↓ w. N_{ov}.

.21 I did often want to wt. predns according to how much pc ↑ of corpus, t. assoc. concepts, considered. part. I'd have to use a main entropy per symbol that existed w/o this special code & to this special concept.

O.K. If a token occurred m times out of m' & there are β tokens altogether; then I had this factorial value formula for a Bern seq. (or Dirichlet seq) that gave t. pc of coding. We have to throw in t. pc of defining t. context. Also divide by t. pc of coding t. corpus w.o. t. defns.

This last done exactly, would involve this info: ① case counts for all tokens in corpus up to now.

② Case counts of tokens in corpus following context of interest.

If we use each context to code t. corpus, then ∥ codes can only ↑ t. pc of t. corpus. (never ↓)! I was afraid that t. ∥ codes would "swamp out" any u.g. code... but this can never occur!

30 but this can never occur!

31 T. details of t. foreg. (including alternative parses of t. corpus wrt t. new defn) is dealt w. in

32 my "Letter to G. Wolff" & t. TM notes of about that time.

The considerations of .21 ff should apply (.21.31-32) (more simply) to a sequential (non-tree) corpus.

Well I'm not so sure about t. Direct relevance of .31 -.32 to .21 ff! In .31-.32 we used defined a gram itself, to ↑ pc of corpus. In .21 ff, we use the definition of a context to modify pc's of following tokens so as to ↑ pc of corpus. Hvr, much of the M&R of .31-.32 is relevant to .21 ff, since we should consider ∥ parsings and the effect that defining a context modifies t. Base counts of tokens not (predicted by / controlled by) t. context.

8/28/04
4TM

$\sigma \cdot e^{\frac{n}{T}}$ = min.

Akaike: $AIC = \frac{1}{2}\ln(\text{mserror}^2) + \frac{n}{T}$ ← no params, T ← no. datapts.
minimize AIC. = ln($\sqrt{\text{ms error}}$)

10:217.40 : The idea of "Value" of some numeric (or non-numeric object) being assoc. w. many objects, is a very human heuristic idea. One way to get it into TM: The idea that many objects will have a number assoc w. them: Or, they will have fuzzy no's assoc w. them. Also that objects correlate w. one another — The correlation being via proximity (~ concept) or functional relations. So a certain function will tend to take certain argument types. If the funct has several args, they will tend to "correlate", So [redacted] knowing one arg. gives "info" about the other.

: [SN] I'm thinking about ngrams, branches, subtrees etc: In a sense of PPM, these subtrees would give a PD over tokens at an adj. context pt.

———— How, one big use of ngrams is that they are often predictable as following other ngrams. This seems like a different (& important different) way to look at things! If we define objects & make tokens of them, then these generalized tokens can be both predictors (context) as well as the things predicted.
We can achieve this by making defns, parsing corpus several ways w.r.t. these defns. & periodicly reparsing & perhaps periodicly removing old defns & well as creating new. Reparsing is not such a big deal if we can reparse O^j each time we modify it. We could also reparse previous O^j's.
We can also have several || parses for a given O^j. → (219.00)

[SN] Re Oops: One big problem in Phase 1 is that there is a O^j that has to work for all QA's! So if we make a modifn, we have to test it on all past QA's.
If we are able to work on each problem anew (as in OOps) we'd not have that problem....
but the amt. of transfer (ing) is then very limited.

[SN] Re: GP Complaint was that "GP sometimes broke up good branches"! This was perhaps because the selection rules were poor. A better way would be to keep top k percentile of population & make cross/muta from that only (replacing, when better cands are found). This is close to what is done when the population is modeled by a p.d. . Or we may want to keep all or almost all of population & use a selection of parents based on G (or = fitness)

[SN] Use diff't notations for function trees (e.g. the one I derived in: Then use PPM on corpus: compare random-ness of various notations. There should be a nice way to combine the 2 codes here (Backwards & forward) and the 4 codes of Look at King and Cover for good way to combine them. ~~Design of TSQ will ... deciding on wts.~~

[SN] A big Q is how much to wt. each O^j in the corpus (for updating): One (seemingly "natural") way, is to use the no. of times it was successfully used w.o. modifn. So Designer of TSQ will end up deciding on wts.

AIC = $\ln(s_m^2) + 2m/T$
m = no. params
s_m^2 = sum of error²

or AIC = $T\ln(RSS) + 2k$
k = no params.
RSS = "Residual sum of squares"

$T\ln\sigma + n$ = min
$\sigma^T \cdot e^n$ = min
$T\ln\sigma + n$ = AIC
Seems
Dimensionally incorrect.
We can change σ by varying scale factor: But this merely (mults) adds a constant
$\sigma^T \to \sigma^T \alpha^T$.
So it's ok.!

8/27/04

4 TM

00: 216.40 : updating : (overlap no. limits / Update expenditure time/cc) can be adjusted so that we get best cc for corpus for
01 given Σ cc. viz considerations of this kind that led to eq. 124.33

02: 216.37 : Actually this tree node notation (≈ 2·3·trees) is not so expensive. w. a corpus of ≅ 10^6 tokens
I have 500 × 10^6 bytes of ram — so 500 bytes/problem: certainly adequate: My 2·3·tree used "only" ~ 36 bytes per token of corpus.
This tree notation does make comparison cheap.

A possl. relatively "rough & dirty" way to get a lot of "points in corpus w. large overlap no." wrt a particular prob pt:
Say α is a pt. of comparison: Get all pts in corpus w. "hy" overlap on (left, rt., or +?) using (PN, RPN) and PPM, {X_i} Then
go over [X_i] with complete overlap comparison to get the more complete set. <Note 216.38 - 217.01 on cc of tree matching">
It may be that usually the RPN, PN matches will give overlap nos of > 8, say, so we need not srch for too. Only if they do < 8 will
we pull out the "Big Guns" of complete tree matching of (.02).

0: 214.09 : **TSQ's in English** : For ANL (≈ SAARBRews): Need to decide just what the problem is ...
what needs to be done. So, to some extent, we must start to choose a language/notation.
So list a set of possl. notations, for Algebra: e.g. RPM, PN, infix (of various kinds: APL uses a mixture)
What notation does APL use for algebra? Maybe infix makes the commutativity of × & +, easier
to deal w/ take advantage of.

Then there is the Machine Crafting methodology. It would seem that all of these choices
must be made before the ANL problem is definable.

Are there some heurs that are useful (adequate)? indep of which of these choices is made?

Some "indep" heurs: (1) Tn l. early + − × / problems; The idea that + − × / are simple "controllers"
They are "ob" outputs that tell us what to do. [how does this differ from 3, 5, 7 ... that also "tell us
what to do"?

10 ABstractly Stated: we have a TSQ of $[Q_i A_i]^n$ pairs: To devise a sequence of machine
symbols (≡ instructions) O^j such that $\left[\sum_j \prod_{i=1}^{n} O^j(A_i|Q_i)\right]$ is max. $\equiv f(O^j, [A_i, Q_i]^n)$

TM is just Given the TSQ $[Q_i A_i]^n$ & for any O^j cand, the value of

I guess that operators like + − ... & numbers are "different" in ways we can decsb. A small no. of operators,
a large no. of numbers. Numbers are all similar in certain ways.

The initial lrng: TM sees "3 + 7 → 10" figures that 3 & 7 are "nos" and are
acceptable args of Binary functs. "+" is an operator name. we/ learn to associate it w. the
operator sum(x, y,) . [Polish]

We could initialize TM w. the knowledge that in the instruction "sum x, y," x & y must be "numbers",
and "sum" is the name of an operation on x & y that gives a number as output. We then "tell" or "teach"
TM what "numbers" are: how to recognize them etc.

A Big Q is how much we want TM to know at the outset.

It may be that choice of lang for defining probs, & choice of inst set for ref machine is mainly
important for ANL: that for problems past that point, discussions in "English" can be made that
do not depend on these choices. E.g. we should be able to solve alg. eqns. equally
indep of these choices .... derive heurs in English indep of these choices
Hvr, a value of having TM know what numbers & functions are, is that later these techniques could be
useful in lrng. new, novel concepts — quite unrelated to nos. & functs.

(218.00)

(somewhere in PS! 2004 I think)

: [SN] on Cramer's early formulation (1985) of GP. Seems he mentioned a lot of good heuristic tricks that were not taken up by the GP/GA community --- Look at Ream!

.12 : 215.30 [SN2] I have a recent paper written as a MS Thesis in Eastern Europe (Now Geyer Maybe at Stanford). In searching for continuous eqs (regular eqns or ODE's) to solve a problem, he restricted such space by limiting forms of functions considered: This is one form of HINT that I may have to use in training TM!

Some restrictions he used: (1) eqns of max degree 2 (2) He gave it the function $f(x,y) = \frac{x}{x+y}$ or $f(x, const) = \frac{x}{x+const}$ (leaf)

This put the pc from $\sim 10^{-15}$ to $\sim 10^{-8}$. Since the fitness function was very expensive (he had to step out the ODE's a/o optimize params of functions derived), ∴ This was very impt.

? { $pc = 10^{-15}$ would be very expensive, if pure Lsrch was used, but perhaps quite accessible if GP were used. } ??

I found .02 while looking for papers on common subtrees in Google, MSN. Much work on largest common subtree. Perhaps repository on other such engines

0: : On searching for common subtrees: A possible (expensive) way to represent trees: The way I described the 2.3 Tree program: Each node has a list of nodes that are its kids. They are arranged in an Array: The Nth row of the array was first: Content of Node N: followed by node nos of children of N.

Or Nth row has contents of children of node N. each content followed by node no of its child.

24 → This may make it faster to find similarities of nodes. To "compare" 2 nodes (to find how large their "overlap no." is): First we note that they have same "content". If they are the same, we examine children. (However we consider commutativity & other symmetries.) We do this by putting one extra "word" (as pert) children onto stack.

("tracing out of identical nodes")
Actually, we may be able to do this w.o. the expensive representation of .20 & 24.

0: E.g. look at the graph of (201.02.L): we trace out "forward": A B C .... P: This traces out the entire Graph. Suppose we want to get the "overlap no." of 2 different graphs at point α. We trace along as "A B C ..." the nodes at which the 2 graphs have the same symbol at α. We trace along as "A B C..." the nodes at which the 2 graphs are identical. If they don't match we drop that node & go on to the next. We keep track of the number of common nodes that the 2 graphs had as we traversed them, we only "traverse" the common nodes that are "connected to α".

The mechanics of how we cut out a non-matching node/branch, is unclear: How we do it depends on notation: Straight RPN (or PN) is cheap, but cutting out non-identical nodes is expensive.

37 The notation of .20 & .24 is expensive, but omitting non-identical nodes is cheap during comparison → 217.02

38 In the forms, we may not want to trace out beyond "overlap no ≤ 8" — In PPM, they didn't get much better by considering much larger contexts .... (This may be because they didn't do it correctly!) This "limit" can be adjustable — depending on how much (time) we want to spend on

217.00

8/26/04
4TM

0:21:4.39! / for Radix 3
In the sequence of ~~integers representing~~ n, $\log_3 n$, $\log_3\log_3 n$ .... The leading char. can't be 0. ~~So we start out~~ When the leading char. is zero, then we know the next integer coded will be "n" the no. we want to represent. Say f(n) is the no. of ternary bits in the expansion of n.

$F(\frac{2}{1}) = 1$ $f(3,4,5,6,7,8) = 2$ $f(9,10 \ldots 26) = 3$ .. etc.

So to code 25 ~~so we~~ need f(25) (=3) and to code 3, we need f(3) = 2. to code 2 we need f(2) = 1

so for 25 we need $f(25) + f(3) + f(2)$ = 6 symbols, + 1 extra zero

.07 Note the leading symbol can't be 1 or 2 because we get into infinite loop. So only number 3 and 1 zero can be coded this way, so this is unnecy; but this still is ucy! So to code 25 we need 3+2+1 = 6 symbols.

Since we can't code 0, 1, 2 ~~...~~ — if we want all 3 we have to add 3 to ~~number~~ an integer before we use this code.

I'm not sure of .07 ff. we would have to code 2 = f(3) if we wanted n = 3.

~~Say~~ / usually The first 2 symbols always represent 3, 4 .... 8. If the first symbol is zero, then the next 2 symbols represent n = 3 ..... to 8.

If the first symbol is not 0, then the ~~...~~ first and second symbols represent 3 ... 8; but these are the no. of symbols in the next number (which will be n if it starts w. ϕ). (or, if the next symbol is ϕ, the next (3, 4 .. 8) symbols will represent n.

So the Q is, will the prefix of any ternary sequence ~~...~~ always represent an integer.

1 2 = 5 2 2 = 8
1 1 = 4 2 1 = 7
1 0 = 3 2 0 = 6

There / may be an awful lot of infinite nos! ——— at certain points in the generator a zero ~~...~~ will start a termination. But zeros of this sort occur less and less frequently, so it may be that ~~almost all infinite...~~ the prefixes of almost all infinite ternary sequences are of length ∞ !? However, lots of long sequences would make $\sum 3^{-l_i}$ converge! — it may make the $\sum < 1$!)

·6:21:4.05 **On choosing betw. LISP & Forth:** In Forth, we generate the output sequentially, so when the cand. is only "½ derived", we have implemented ½ of its "test" (say for INV probs). (However in OOPS the prgm did work the problem "on line")

In many cases, this is imposs. — that the test requires the entire cand. Probably if we used (as Rozz did/does) the SPICE prgm to test cands, we usually need the whole cand before we can begin testing. If we can't do this, then Lisp does not have that disadvantage.

The second Q about Lisp vs. Forth is whether PPM or other regularity ~~detectors~~ (e.g. PSG discy; common subseqs etc.) could more easily find regys in Lisp code vs. Forth code (Also consider "Machine Code" & APL and ≈ BASIC, Fortran, C // what about Object Oriented langs?)

The third Q in choice of lang. is speed — Tho it may be possl. to modify slow langs like "Lisp" — by compiling them a/o by making primitives close to Machine lang. Another speed point is to arrange lang. definition so parallelism of modern chips can be used. There is a Japs chip in development at \$15/Gigaflop — very parallel, but one must properly design search tasks for it.

4TM

00 : "Prospectus" for TM: A path of work for future. Give various Alternative choices. List various problems &, if possible bibliography of where I've done work on these problems. Put this on Computer, so I can insert, Modify sections more easily. — Also ideas from ALP.

→ Perhaps a MAIN idea: that I want to get started on TSQ as soon as possible, because many

05 other problems (like choice of Ref. "Machine", ≃ "language" to use; "Forth" vs. "Lisp" (i.e. sequential v.s. each input is member of prefix set), perhaps form of a priori on "integers" (212.20) — might also be solved by the TSQ's & their Heurs. → (210.11) → 215.26

Describe in some detail the "Integrated Approach to TM" of 105.18! How are phase 1, phase 2 approach and the TM, = The approach described in Abstract of the IDSIA Report.

— I think the main idea is to start w. TSQ writing, in English. Get all of the Heurs I want

.09 to use, for each problem, And an idea of which corpus I'd need to discover the heurs. → 217.10

10: 212.32 SN The system gives a certain a priori to the integers — (I can adjust the amt. of wt. it gives for short v.s. long R's, by using different α or by other means). There is, hvr, the Q of All other inputs to the 3 input mc — That the tokens all have different pc's .... How can R deal w. this? Well, perhaps

12 → "No problem". If we are writing the token using R, we use pc in the same way as we would for any other mc input — we just multiply the pc's together. — Hvr, my mind is not altogether at ease about this question! Perhaps Considerations of Prefix set properties & Normalization are relevant — — That .13 is adequate for the problem!

Well, the Q input: We don't care what its pc is. — But the O's & R inputs are "self del".

An Aside: Leonid said that the $\lg x + \log\log x$ ... seq. diverges for base e: (& presumably for base > e.) Can I devise a system (like 211.14ff for base 2) but use Base 3. — it has to converge!

.19 → But it may not get the exp ... → 4

20 SN Re: The PSG-discy problem. It may not be so dfclt! The diffy may have been that ① people really didn't believe in the proper Gore ② If they did, they didn't try many parsings. ③ They may not have considered parallel parsings as part of the Gore! i.e. They tried to solve the identification problem rather than the ~~prediction~~ problem. ←⊙

Also note: I'm not really in PSGd. I mainly want to find common sub functions (subtrees). There may be work on this: try Google. Also other search Engines.

SN Look in Google: { Phrase Structure Grammar Discovery / Context free Grammar Discovery }
Also try Yahoo, Excite, Altavista, MSN.

Also Look in 2 citation indices: for forward ref to work by Chen (?) or others. Also, Rorian of ~1996 at Harvard will have Bibl. that I can use for forward refs.

I tried Common subtrees in Google: lots of pages,

30 — usually "Maximum common Subtree" .... See 216.02 for Best SN. Try other Srch Engines.

Tree Compression Google: Got a review article! (See D:/ ... PS/8.28.04)

): 212.40 : I could use R to "Modulate" the effects of the past corpus, on the present problem.

In OOPS: It is possible to have a problem fixed "Oj" pgm in a Qi objective memory, or on a stack, & have a R pgm that Generates P candidates, sequentially. } R is a Universal TM on Ai of choice of Oj & Qi.

05 DEP Going back to [204.39-.90; 206.00ff model (which I'll call "Model M"), we can regard Each Ai as being a function of its Qi: But by having Rs "properly list", we help the search for good codes for Ai. Say we had many properties & n-tuples of properties & we correlated them w. the freq. of use of various Tokens & Macro-tokens in the correct Ai's. Even w. best properties & N-tuples of properties — all we get is a vg. p.d. of ~~the~~ expected use of tokens in the target Ai. Even if it were sharp enough to give us exact freq. of use of ~~the best~~ tokens in best code of Ai, its still not = Pc=1, (Pc of would certainly be rather good — ..., it would still be extremely "fuzzy".

This [expected freq. of use of Tokens is simply a generalization & improvement of OOPS's "boosts".

It's a good heuristic, but it is certainly not entirely adequate.

A possibly adequate "Model M" (.03)" would have lots of Macro tokens — so the final solution would, ideally, use only a few of them.

Another (IMPT Bit) Implementation of Model M: Model M finds the best pd's for tokens — "best" in the sense that they give the highest Pc for the corpus. However, the P.D. of a token at a particular pt. in the generation of a cand, should depend not only on properties of the Qi, but on previous tokens that it has used: This is in direct contrast to PPM — ... How can we "mix" PPM w. the effect of properties of the Qi (∈ Model(M)? Or the pd of tokens should depend on "order of the token (how far back) a token in the generation of the pgm for (the target Ai's Cand appears? (This last is "similar" to previous local "previous history" as opposed to corpus "previous history".

The Model M "token probs" are like "unconditional" probs: Conditional probs would be based on ~~recent~~ local recent history. So the conditions should be both corpus previous tokens & local previous history.

8/24/04 Note that this OOPS method has crossed above $Q \to A_{x'}$: R is used by Oj to implement the P.D. Would it be OK to put the Continuous info about a P.D. into "R"?

0 : 3 pd's? (1) Original pd prior before TSQ: (2) a pd of Sum after TSQ has been Run.

(3) Guiding PD. Actually, the GPD is a pd on {Tokens/Macro tokens} — Rs it assigns to a pd on the sum. of the last problem.

.10 : [SN] (≈ R 210.32): Some time ago, I considered using Lisp w/o AZ as a 3 input UMC (≈310). While Lisp & AZ / Monkey have some difficulty about "Asking for more bits",

.02 (i.e. we don't evaluate them until all bits of args are known.)

If we can tolerate .02: a way to get a 3 input UMC def on output: Consider AZ or Lisp: We have a function w 3 inputs. The input alphabet is 0, 1, Δ (Δ means "end of args.")
So any string consisting of k 0's & 1's followed by Δ is a k+1 symbol string. & these strings form a complete prefix set, since any string of 0, 1, Δ can be uniquely broken into items in the prefix set. The total pc of the set is

$$\sum 3^{-l_i} \quad \text{where } l_i \text{ is the length of the } i\text{th member of set.}$$

.0 The first member is Δ; next (0Δ, 1Δ; | 00Δ, 01Δ, 00Δ, 11Δ etc.) [1, 2, 4]
There are just 2^{l-1} strings of length l.

$$\sum_{l=1}^{\infty} 2^{l-1}3^{-l} = \frac{1}{2}\sum_{l=1}^{\infty}\left(\frac{2}{3}\right)^l = \frac{1}{2}\left(\sum_{l=0}^{\infty}\left(\frac{2}{3}\right)^l - 1\right) = \frac{1}{2}\left(\frac{1}{1-\frac{2}{3}} - 1\right) = 1$$

$\frac{1}{2} \cdot 2 = 1$; $(3-1)$

.14 Another way to do it is to use the "complete" $2^\wedge(n + \log_2 n + \log_2\log_2 n \cdots)$

.15 Every string is has a member of the set as prefix i.e. it is a complete prefix set. → (212.06)
The first few bits of each member tell how many bits contain "data".
We can then feed random 0's & 1's into a "front end" that decides how many of the bits are legal code, & then stops the input.
I've forgotten how my original .14 worked. This it did have property .15, it may have assigned > 1 code to each integer!

-0 A way to do .14 w.o. this diffy: we have 2 kinds of (binary) strings.
type (1) first bit is 0 : to get value of string, convert the 0 to 1 & evaluate as binary no.
type (2) " " 1 → " " " "
Index string Type 1 tells how long the following string is, ("the following string" can be an index string or a data string)
data string type 2 is a data string.

.26 The data string of binary no. n has $\lceil \log_2 n \rceil$ bits in its representation.
.27 The index string preceding it has $\lceil \log_2 \lceil \log_2 n \rceil \rceil$ bits in it.
.28 " " " " " " $\lceil \log_2 \lceil \log_2 \lceil \log_2 n \rceil \rceil \rceil$ " " etc.
.29 ~ So the sum of .26, .27, .28 .... is the total no. of bits needed to say "n"
.30

1000 = 8
101 = 5
100 = 4 5,6,7
10 = 2, 3

$\lfloor \log_2(n+1) \rfloor$?
4567 → 3   100
23 → 2   1
1 → 1   n·1
so maybe $\lfloor \log_2 n \rfloor$?
10 = 2
11 = 3
$\lfloor \log_2 n \rfloor + 1$

So say 17 uses 5 + 3 + 2 + 1 = 10 bits. (data)

3 needs 2 +1 (data)

00|11 → = 3 (index bit, data bit)
0 0|10 = 2

10 = 2   no way to represent 1
1. = 1

Maybe not 01 → 11 → 3
00 → 01 → 2

00 always ??

00 is silly so say it means 1
01 is 3
11 → 3
10 → 2

o.k. so 00 means 1
10 " 2
11 " 3
01 → 11 is index 3.
so 01 100 means 4
01 111 " 7
2 3

4, 5, 6, 7 all cost 2+3=5

2+2

say $f(n) \equiv \lfloor \log_2 n \rfloor + 1$

8/23/04

4 TM

.30 : 211.40 say we want to find how many bits we need to represent n

$x = n \quad s = n$

Do

$x = f(x)$; $s = s + x$

until $x = 2$

OD

.05

s will be required output if $n \geq 2$

If $n = 1$ then $s = 1$.

I think I compared this to normalizing const.

$A(\log_2 x + \log_2\log_2 x \cdots)$ ... but I may have compared incorrectly.

$f(x) = \lfloor \log_2 x \rfloor + 1$

$= \lfloor 1 + \log_2 x \rfloor$

06 : 211.15 } Anyway Returning to the sheep! In the 3 input case:

One input is O^j, another is Q_i, & third is R. We feed O^j, Q_i & R into the Machine & it evaluates the standard function. For the R input, we can use binary integers of fixed length.

We can assign the P's to the integer n via $x = \lfloor \lg_2 n \rfloor + 1$, $pc = \frac{1}{3} 3^{-x}$ ($x = 5$)

or via : algm .00 - .05 2^{-5} ; (.20) (Compare for $n = 17$: Algm .00 - .05 gives $2^{-11} = 2048^{-1}$; $x = 5$; $\frac{1}{3} 3^{-5} = 3^{-6} = \frac{1}{27^2} = 729^{-1}$

$x \to 5$; $2^{28} = 2.68 \times 10^8$

say $x = 17$ $s = 28$ ($= 17 + 11$) 2^{-28} v.s. 3^{-18} $3^{18} = 3.87 \times 10^8$

So for small n, 3^{-x-1} is larger, but for x as large as 17, 2^{-s} is larger!

But $x = 17$ means $n \sim 2^{17} = 128k = 1.28 \times 10^5$.

Well, this pt. is not critical.

I think that for large n, the $2^{-s} \approx 2^{-n}$

but for small n, it's much smaller

hvr $\frac{1}{3} 2^{-x} \approx \frac{1}{9} n^{-1.585}$ ($\doteq \frac{1}{9} n^{-\lg_2 3}$)

$3^{-x-1} \approx \frac{1}{3} 3^{-\lfloor \lg_2 n \rfloor + 2}$

$x = \lfloor \lg_2 n \rfloor + 1$

$\frac{1}{9} 3^{-\lfloor \lg_2 n \rfloor} \approx \frac{1}{9} 3^{-\lg_2 n}$

$\lg_3 n = \frac{\lg_2 n}{\lg_2 3}$; $\log_2 n = \log_3 n \cdot \lg_2 3$

$\frac{1}{9} 3^{\lg_3 n \cdot \lg_2 3} = \frac{1}{9}\left(\frac{1}{n}\right)^{\lg_2 3}$; $n^{-\lg_2 3}$

$\frac{1}{9} \frac{1}{n^{1.585}}$

$2^{1.585} = 3$

.10 : (.10) We put in binary integers for R, 0, 1, 2, 3, 4, 5 ...

= 0, 1, 10, 11, 100, 101, 110, 111, 1000 ...

1 2 3 4 5 6 7 8

and we see if the output = A_j

.22 we want the smallest R for that output. The pc assigned to that R can be either of the above methods. (.28)

— Or other methods; Normed $\frac{1}{n^{1.585}}$ would be ok. We can use a normed $\frac{1}{n^\alpha}$ for any $\alpha > 1$.

(The norm constant for $\frac{1}{n^\alpha}$ is related to Riemann funct of α).

(SN) for normed $\frac{1}{n^\alpha}$ (A) v.s. (.00 - .05) (B), A > B for small n. — but for each α there is always a value of n for which A = B : for n < that, B > A, for n > that A > B

So for very large n, B always wins: for small n, A is always better.

.28 : (.22) While this works fine in theory it is quite wasteful of cc : Each value of R has to be tested individually — For 2 different R's, there is no sharing of common tasks, as would be true, if the ref. machine worked along R sequentially (as in OOPS or Forth).

[I think the difference amounts to a factor of d, where d is the depth of the tree = the length of the coded corm. [This may be indep of Radix ... but I'm not sure] → (214.10)

.32

N.B. One of the Ref. Machines I must consider is APL

Hvr. dropping that for a moment, suppose I had the O^j that summarized previous corpus using the default ideas of 204.34 - .40, 206.00 - .40. This gives a default pd for A as a funct of Q_i.

should this be the PD for null R? If so, how does R modify the pd on A?

Well the default D.P. bases the pd's of all tokens & macrotokens on the previous corpus.

A more general pd could base these things on the corpus this far. Also there is a sequentiality in the ideas of 204.34, 207.00 ... that's quite different from the Lisp/AZ approach.

This is a non Lisp/AZ approach!

(213.00)

8/22/04

4TM. Conclusion (protem) .11 ff

00:208.40 to review old pgms: we just want to vary wts so as to max t. product of t. pc's of solns of all of t. pgms for t. A_i.

Also in "Updating" we can try to find new Obs (features) of t. Q's & ways to relate them to t. pgms for t. assoc "A's". These "Relations" need not be simple wts.: they can be any functions.

T. forgg stuff: we have a Q_i & obs on it & as functs of these obs, pcs on tokens & "Macrotokens" that give a pd on a assoc A_i. The only way we can vary this pc on Q_i's is to change, add any new obs, change relation of obs to tokens & macrotokens. Add removes, augment macrotokens.

We want more flexible ways to vary the relation betw Q_i (& its obs) & t. final pd for t. associated A. The "present" way is by varying pc's of tokens & macrotokens — This is pretty not t. most General way: The "pc's of Macrotokens" is a big step toward Universality.

Conjecture: Heurs for QA. When I have Heurs for QA problems & — a TSQ of them. — I expect that any attempts to implement these heurs will suggest solns to forgg questions.

T. heurs will suggest solns & perhaps ways to implement those heurs using t. forgg. models & **Modifns of these Models.**

AH: Th. Discussion of t. R funcs (≡Obs≡Properties) in t. early discussn of QA probs. in t. **IDSIA report, is** very closely related to forg. — The R funcs are meant to deal w. d Funcs (i.e. NMTM). But Actually, R ~~functions act "Beg t. Question"~~ Actually t. Pz funcs "Beg t. Q": R funcs are pure OBS: they map into Yes/No. They then control t. $P_z(A|Q_i)$ functions, which are sf funcs of t. type that I'm trying to {Define/Create/Engineer}.

If t. R^z funcs were narrower, they would arrange $P_z(A)$: i.e. t. P_z need not look at Q_i — R^z has already looked at Q_i .... We have 3 cases: ① No R at all in which O^j look at Q_i & give pd for A ($O^j(A|Q_i)$) and ② $P_z(A|Q_i)$ ~~in which there is a set of~~ in which R^z recognizes a set of Q_i's such that a certain $P_z(A|Q_i)$ will work for this set of Q_i's & ③ R^z ~~etc~~ will look at Q_i & decide Yes/No whether its associated P^z will give t. rite $P^i(A)$ for that Q_i....

So I still don't have an adequate set of ways to (realize / implement / parametrize) $P(A|Q_i)$. 207.20 ff is still looks like t. beginning of a good way to get More general $P(A|Q_i)$'s

— So far I have been using only a few ways to "Modulate" $P(A)$ (= pc of A: a sf funct) — Obs on Q_i correlate to pc's of tokens & Macrotokens (programming) of A.

---

32 In 207.00 — .19 I had some "special situation" ways to characterize sf functs:
1) Fixd funct form, but set of real params functs of Q.
2) Fixd discrete set of funct forms w. funct & params funct of Q
3) Universal set of functs; functs of Q; also params of f_i are functs of Q.

Limits of Human Reasoning/Induction: 00

:SN On HUMAN (vs, induction): T. w/ human languages are created, found: Humans are born w. certain built-in deduction, induction facilities: Human langs are designed so they are learnable by these facilities. These facilities are probably not universal, so certain langs synthetic langs would not be learnable by a newborn.

Also, humans may use Post-non-universal reasoning facilities for all Domains — so we would not be able to ever discover certain regularities of kinds of physical (or other natural) laws.

On the other hand, we may be able to design machines/progs that can do reasoning beyond that available to Humans.

Turing/Partial Rec functs also able to express all Human ideas of Functions. Could there be Functions not accessible to Humans — perhaps accessible, but perhaps not accessible to Turings.

As it is, Human induction does seem limited to ≈ eg. Phrase Structure langs.

Pos = CFL's: Those we can include context sensitive langs — which perhaps makes them universal (CTAG lang is ≥ Cont. Sens. Regs. [very "simple"] investigated by Post, as is universal).

It may be that Humans can only deal w. very limited Context sensitivity. The ability of humans to understand the gen (non CFL) concept "recursively" probably does not generalize to all context-sensitive (even rules).

→ Another possy: That humans can get beyond their inborn reasoning facilities, but they do this awfully slowly ≈ as by TC — so machines could be way ahead here. Humans may be able to transcend their newborn reasoning by writing things down, by drawing pictures or their logical equivalents.

ATM

0 :207.40: The D.F. on A_i depends on t. previous solns, but in a fixed, standard way. So it's a fixed function of past. A (Perhaps) Goal: To make it "search" around that past.
→ One kind of search can be over "properties" (Obs) of t. Q_j's.

A possibly related problem: we have a UMC which on random input, gives a certain (universal) d.f. on its output. How do we modify that UMC to generate an arby p.d. on its output?
Since it is universal; ~~[illegible]~~
∃ a finite initial input string, that will convert this UMC to any other ~~[illegible]~~ True Output DF.
07 Since t. pd's on Tokens induced by 207.20 ff are all "nondestructive" (i.e. They give ~~[illegible]~~
08 $pc \neq 0$ for all possl choices) I think t. resultant D.f. has to be Universal. ??

I'm not so sure of (.07-.08) "$pc \neq 0$ for all choices" means that no item has zero pc.

10 A machine w. [illegible] pd's at each instruction point is equivalent to a deterministic machine w. a pd on each input symbol. If this pd on inputs is "invertable" (i.e. ∃ a pd on strings that converts it to t. uniform d.f.), then the code for that inverse as prefix, x-fms t. original machine into a Universal machine. — so "invertable" prob'listic machine must also be univ'l.
If t. Machine of .10 simply has t. same modif. of pc's of tokens i.e. uniform pd = $\frac{1}{k}$
15 $\frac{1}{k} \to pc_e$ for all t. tokens; we can invert using Normzd $\frac{1}{pc_e}$ ‖ $pc_e\,\text{new} = \frac{1}{pc_e\,\text{old}} \Big/ \sum_{e=1}^{k} \frac{1}{pc_e\,\text{old}}$
If t. pc of a token, T_e is a function of values of all previous tokens, we can still use t. Method of .15 to invert t. d.f.

If a p.d. assigns p.c. to ngams as well as tokens, we can also express it as a pc on t. next token only,
18 by summing pc's of all ngams that begin w. a particular token. (? — What about t. pc's of t. second (say) token in one of t. ngams?!
20 In .18-.19 There is a parsing problem: If t. ngam (defined) αβγ occurs we don't know if t. first char. was obtained from th. ngam αβγ or from t. token, α.

Perhaps a quick: Say we have a grammar that assigns pc's to Tokens and to ngrams.
At first consider grammar:

$S \to N\,S$
$N \to a \mid b \mid c \mid abbabc.$
$\quad p_1 \mid p_2 \mid p_3 \mid p_4$

(a,b,c are terminals, S, N nonterminals.

This gives a pd over all strings of a,b,c! Is there a grammar that can, w. these strings as inputs
26 put out a p.d. w. pc of a,b,c = $\frac{1}{3}$? Well, that's not what I want.
To get t. uniform abc is not hard: For any input symbol, output pc = $\frac{1}{3}$ for a, b, c: !!
What is it I'm really looking for?? A deterministic grammar?
30 — I could get that by listing all possl. outputs, as a pd. & assigning $\frac{1}{3}$ of them to a, $\frac{1}{3}$ to b, ect.
— More precision by considering seqs of 2, or 3 or ....

32 Go Back to 207.20! We have this set of OPs on t. Q's. We have t. set of associated A's. We have the PGMs, that have created the A's. The pc's of Tokens & subfunctions, acts of these pgms have been obtained as "correlated" (or other functions) of t. obs on associated Q's.
When we get a new Q_i, A_i & t. PGM assoc. w. that A_i, we try to adjust t. relations betw. t. Obs & their correlated tokens, ngms ect, to be "Better" i.e. so that t. Total pc of t. corpus is Maxd.
Updating will consist (at least) of optimizing continuous params (wts) in view of t. new data pt. Since t. pgms for all of t. solns are invariant, we don't have → (210.00)

SFUNTION realization, Option for induction ...

: Memo on Sfuncts Generation, representation, inducing:

I can use in any different forms of Sfuncts, for different kinds of problems:
I can give TM t. form & have it find continuous/discrete params for t. problem.
E.g. for hC) problems Give it Tier Gamma D.F. or t. Gaussn D.F.
& esp for t. 3 params.

See, if I can use t. "EM" (= expectation Maximization) method for any of my param optim problems.

In other cases, I can ask for a pretty much not D.F. & one or more of new moments of t. D.F.

For Discrete D.F.'s ● one (or more) € / cluster centers, and a "distance" rule w. assoc continuous a/o discrete Params.

For some problems I may want to use t. Input time model, but I really don't know how to do this effectively. — Tho t. OOPS "Levin" model would work.
I have worked out some details on how to do this, here. How practical it could be w.
OOPS Fortn, is unclear. There are 3 input params: O' (the general model)
Q_j: tho 1st question is R t. "random" no. We could arrange so TM knows it has to read
O_8's first, then a Q_j then try to make a prgm to get A_j next as short as poss!. ... Tho we will
be interested in several other prgms as well. — (see .20-.23) ↓

T. idea of an "Exercise" TM that knows it can used t. easy to get info; perhaps
This would enable (.12-.15) !?

● If I had some hours for realizing t. soln to the stochastic QA problem — then, I should try to put t. 3 input time model stuff into that / those routines

.15 A v.g. way to get [Universal] Sfuncts ! 204. 34-.40; 206.00-ff ! ← This considers the QA problem:
A bunch of Properties (= Obs) on tho Q's are (found/given/tried). These Obs & pairs & triples of them induce d.f.s on t. individual tokens & on functions, branches, etc. The output does correspond to that of a 3 input time. How this routine does it (each of the details of Q can be ...
● I think it gives a very fuzzy answer, but perhaps I can modify it to "do t. right thing"!
Also: I want it to give rite answer in deterministic (Math) case. T. methods of 204, 206 don't seem to do that, but perhaps I can get them to. !
.32 of Say Q_i has occurred before & TM has obtain rite answer. If Q_i has same answer, [For a time, no problem] — but if t. answer is diff; All properties of t. Q are to some sort will try same instructions & functions that worked last time, then "move out" to less likely conds: t. lang, of course, will still be universal.
.29 Say Q_i is "by" Q: lots of properties bits: But only a few differ from the last (correctly answered)
.30 Q_j. Depending on properties, Q_j & Q_i may or not look "identical", but for Q_j, TM will usually try the tokens & functions used on Q_j.
What this system does n't do well: in .29-.30 say we have very large functions w. only 1 numerical argt. differing from that of previous Q_j. (Like $Q_8 = \sin(\Gamma(3+\cos 7))$
— Tho. 5 was a "4" in Q_j. Would like TM to be able to localize t. difference between Q_i, Q_j & make a "narrower" trial set.
Trouble w. .24 as an improvement on t. 3 component sum of Sfuncts: The property list of t. Q_i contains a complete description of t. Problem & t. answer to Q_i. There is no mechanism to search or to try new kinds of Sfuncts (via D.F.s on A_i's). It is not a D.F. on D.F.'s on A_i's, but a single D.F. on A_i.
I think I need a Universal Lang to describe t. kinds of P.D's on trials that .24 ff obtains
Modifying Property list a/o relations of pairs of tokens & sub-trees, would not seem to be "universal". 20800

8/20/04

4TM

00:204.00: Also, for singlets, pairs, trips ... of properties, we can get probs for [Subfunctions of various pairs,] n-tuples of Tokens. This would be a v.g. way to find subfunctions.

T. subfunctions (or n-tuples of Tokens) could also correlate w. previously generated parts of t. Cand.

So, t. function/leaf chosen at a pt. in t. Cand generation will depend on (one or more) properties of the problem also on the previous choices in t. Cand generation.

Both of these choices will suffer from small ssz so we will make t. pc of t. token selected, depend on whatever "property(s) of problem" we have data on — t. ssz of the data: The recent context in t. cand, & its ssz (since there will be many contexts — each w. its own ssz)

The "properties of the problem" can include External Context (e.g. Chemistry problem, Dynamic Prog, linear Algebra ....) as well as Context based on studying problem descr itself.

We can use "Encyc" to occasionally decide what kind of Prob it is:

In all of t. forgg, TM does not have much "understanding" of what t. problem is: The properties & their corrlns are, indeed a kind of "understanding" — but not as good as a person "understanding" t. text of t. problem. Hopefully, after TM has worked lots of Algebra (say) problems, it will have found lots of useful properties of t. "problem descr". Would it be able to look at a Partial Dif Eq & decide what "kind" it is so it would know what soln methods were feasible? ..... Well, this sort of thing can be lrnd via TSQ.

18 One kind of Understanding of Q: It will lrn to understand "solve" for various args. Maximize (for Discrete, continuous params)

At first, we only give it Induction, INV, OZ probs. Later, only INV, OZ probs. Later, it finds out how to convert INV to OZ. Each problem type has a standard form, so TM really knows what is required. APL (xJ) (for APL in, just past, trys)

Hvr, even after a lot of trng: Consider QA problems: TM could have lots of trouble "understanding" t. Q's — But part of t. QA problem TSQ is learning to understand what t. Q means, e.g. in 18-19 w. "solve", "Maximize"

Also, I expect to teach it to understand English about Domains of its Competence

30

0

8/19/04

4TM

Ranz. — Husband — Bladder cancer. 2 wks ago.

00 : 203.22 : Also, continuation ~ 199.32: in this next situation: GPD is able to learn:

.01 At first it adaptively modifys pc's of fishing. say Lapsor Dirichlet's rule. Also Methods by G. Wolff & M.Li on Chunking in DNA, Protein Analysis

.02 Next PPM is usd, Next trims on PPM, next PSG discy — using Stolcke a/o my improvements

03 (3) on next level, say I look at TSQ, & seqs of solns, & I notice regy(s) that could be usd for predn, but they are not accessible to TM thru PPM ··· PSG disy ect.

05 → I could modify code (say so that this (these) kinds of regy would be accessible via

06 PPM, extended PPM + (201.00 — 202.60 ··· 203.00 — .20) PSG disy, ect.

If .05 — .06 fails, I want to look at finding regys in a corpus of ppms as a problem. It is solvd by Lsrch (say) over a random set of regularities defined by a universal lang.

— So essentially, we'd like Phase 1 to be able to work on this problem. (TM₁ = TM₂)

10 TM₁ looks for solutions in the primary TSQ, using a GPD that is at first, updated by methods like .03 — .06. When these fail (or even if they still work or less work), we try to view

12 TM₂'s problem as being "improving the GPD".

13 .03 — .12 looks like a return to the original IDSIA report i.e. the system that was dscbd in the Abstract of that Report. It does not mention the Phase 1, Phase 2 system. When I wrote that abstract, I didn't know about the ideal soln to the OZ problem using h(·) functions ... and the [slope diagram] α slope param. (Ideal solution to Gamb. House problem ···· better than Lsrch ··· truly

17 optimum

18 It is most desirable/good idea, to integrate the old "Abstract" approach, w. the Phase 1; Phase 2 approach. In the "Integrated" method, we start w. Phase 1 & we use the methods of .01 — .02 for updating the GPD. We eventually begin giving TM problems in induction like those needed for GPD updating. When it gets mature enough in this area, we have it work on its own GPD updating. In this last problem, there are many methods TM is already familiar w. { like the methods of .01 — .02 } If gives certain wts. to these methods, & it has already had experience using the h( ) methods (.13 — .17) on OZ probs & it can be usd for the IND prob. of GPD updating. [Note that in general, it will look at any problem, & use obs (properties) of the prob. form to help solve it (as 204.34 — .40, 206.00 — ff)]

: How to get most PPD's: Discrete, Continuous; Using various — standard forms

4TM

00:203½.40 Recomputation of top of 203½: using "add on" only: (Prechosen)

(2) 21+4 = 25 = no insts $\frac{(25)^6}{2\cdot 2\cdot 3\cdot 2} = 1.6\times10^4$

(3) $\frac{(29)^5}{3\cdot 2\cdot 2\cdot 3\cdot 2} = 2.8\times10^5$

(4) 15+21 $\frac{(36)^8}{2\cdot 2\cdot(2\cdot 3\cdot 4)\cdot 2\cdot 3\cdot 4\cdot 5} = 4.9\times10^8$

(5) $\frac{(43)^7}{5(4\cdot 5)(5\cdot 6)(3)(2)} = 7.55\times10^6$

These are $\frac{1}{pc}$ values: They should be ~ no of insts tried.

Inst 5.2
P30 : 50
31 12
32 6
Σ 68
#69,70
P33 2 (1999, restart) + unnec (untraceable)
P34 3 (21,72,23)
Σ 73

---

0 Discussion of Inst. timing / Traps, etc: 830; Top

P20 1 to 2 M steps/sec on 1.5GHz cpu : so roughly so ~ 1k clocks/step.

so .5 to 1 μs / time step.

For (6), Its not clear how to Calculate! It may have been done as a completely separate ppm:

5 instructions : But c1 & c2 had been "Boosted".

$(21+2.5)^5 = 23.5^5 = 7.17\times10^6$ which is rather small.

Ppm (10) Tower of Hanoi, P21! 10 insts $(21+5)^{10} = 1.41\times10^{14}$!

Comparison: 5×10^{11} for OOPS! The disparity is due to the very serious affect of Boost for this problem. Boostq(2) uses 5 insts of Problem #6 (how this was called "2" is unclear)

(Also note, in my "21" insts, several were never used before :. they should be even lower pc!)

Also Not in (10) (ToH)! Several "New" insts were used. So they prob'ly would be of very low pc (in My formulation)

0 **GENERAL** Discn of **OOPS**!

OOps was able to solve Tow of H. Because

1) It had set of instructions that gave a short (10 inst) soln to it.

2) By using boostq it was able to get large pc's for 4 of 4 insts: for 4 of these c3, c4, c5 & it had a "Hint" boost: only "dec" & "boostq" were un boosted.

3) As I pointed out, the amount of milage one gets (in CJS ratio) depends critically on the no. of insts used. W. 73 insts, he got factor of 1000

Only 21 insts used for both problem types; The improvement ratio is ~1 for maybe 14 insts. It maybe a 4th power affect, since 4 insts are involved.

so $\frac{21}{14} = \left(\frac{3}{2}\right)^4 = \frac{81}{16} = 5$ so he'd just get a CJS improvement of ÷5

30 if he only used 21 insts.

4) His version of **FORTH** & his method of getting recursions is of interest: (The Using a DO loop is faster.) It's not clear that either is of larger pc. One Q is how it fits in with other functions, problems, hours. e.g. How PPM treats them

34 5) Boostq could be a very useful idea: We look at each problem & we have a property list for it. The properties are correlated w. prob. solns. When a new prob occurs, we get its properties. Each property gives a wt for all previous solns. We then use a wtd Σ of all insts used in all correlated problems. — This gives apri wts for pc of Tokens in new problem.

For more precision, we'd need, make Correlations betw pairs of properties, perhaps triplets. Actually some recent work on how to do this → "Opt. Dependency Trees..." (S. Baluja, Scott Davies 1997a) (Hard Copy I have a hard copy)

40 for each pair, triplet ... of properties, we get a probability for each Token. → 206.00

SN Do experiment w. 26 Assembly insts.! See how recursion can be implemented.

Criticism of OOPS problem set-up: Oops has no way to recognize that it has been working on ToH rather than 1H2M. Except: Re setting of pc's of tokens; Also was started start of ppm each time or end of last ToH successful trial. So OOPS can't build up variety of Properties to describe Problems.

→ only one of 3 factors listed in 202.29–.32

0:202.40 : In view of 202.38–.39, it would be easy to drop 202.26–.32 & use the simple mean of the PPM predictions of 202.38–.39

Cover and King had a (to me, non-intuitive) method of combining prediction systems — perhaps look into it.

[The PPM has a way of wtg. [~~...~~] contexts of diff. lengths. I had a method of doing it using a kernel in Σ space. The idea (perhaps incorrect) was that the wt. should depend not on the length of the context, (202.29 — this is usual PPM — maybe is this) & not on how much its length differed from the longest context; but instead, on how many "contexts distant" it was from the closest context.]

← This may be close to the "To the Front" method of Borroughs & Wheeler.

.0 It would be interesting to try 202.26–.32 on simple sequential PPM as a method of combining context predns. It might be v.g. & it might take less cc that the "kernel" method I had in mind.

What makes 202.26–.32 not so hy cc is that data for all contexts of same "overlap no." are pooled.

In the 4 methods of 202.38–.39, it would be well to try them individually, to see that each had about the same predn capacity! (The 2 reversals of 202.39 would seem to make any change in predn capacity unlikely! 202.38, however, could give signift. different (compressed) entropies.

.7 Another ~~4~~ (2) types of context (in addition to 202.38–.39) can be obtained if context is toward Root v.s. toward leaves! So this gives ~~8~~ (6?) context types that are easy to find & keep track of.

→ Conversely, only 2 more contexts, since each node has only 1 parent. (Tho I think we can still invert order of Binary functions, non trivially! so Maybe 4 more context types ≤ 8) (.27)

.0

SN This may be out of our conventional "Phase 1": Go Back to 199.20 –.26 I really need to

.22 Work out exact details of it. → O.K.! — Say we got to 199.32 : we have a TSQ → 205.00

SN Continuation of 203.20: Improved methods of PPM & then PSGdizg viz ..... & my own tricks.

SN Early version of the Conversion Theorem: Since 1 or 0, or U will always be an output, the Norman factors are always Exactly 1: {The output set is a Complete Prefix set.}

.27 (.20) SN In the "4 methods" of .17–.20: "Double Counting" is not a problem! It is a "Feature"!

i.e. ~~when we~~ The "double counting" is (I think) always a legit example of alternate coding.

e.g. when "sum" has its args commuted. If it is really an alternative code, it should be counted.

.0 [A Counter Argt: That any self-limiting code can be written backwards & still be a legit. self-limiting Code. Since this is true for All codes, considering the backward versions leaves resultant pc's invariant.

SN In deciding on/designing a ref lang: Look at successful cond. progs in each lang. Would PPM be able to find regys? Would PSGdizg help? Langs to Consider: LISP, AZ, OOPS (other forms of FORTH e.g. the lang I used in SHARB for ANL) Assembly lang (26 msts). Fortran/Basic/C.

8/18/04 — 205 1/2

TM — Usage of insts in OOPS

OD

4 Problem (3) denom : 21+3 = 24 — $\frac{1^3 \cdot 2^2}{24^5} = \frac{24^5}{4 \cdot 38^{10}} = 2 \times 10^6$ — 50

" (5) " : 38 — $3 \cdot (3 \cdot 4) \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 1.09 \times 10^{13}$

← p's of solns for problems 2, 3, 4

Prob (4) denom = 47 — $\frac{47^{18}}{4 \cdot (4 \cdot 5) 3^3 \cdot 2 (3 \cdot 4) \cdot 2 (2 \cdot 3) (2 \cdot 3 \cdot 4) \cdot 2} \approx 4.2 \times 10^{22}$

"add on"

Its clear those "add on" solns would have not been obtained w.o. special reference to previous soln.

Problem	no. insts used	Σ
1	2	2
2	5	7
3	10	17
4	18	35
5	25	60
6	5	65
8	1	66
9	7	73
10	10	83

7

(crossed out:)
8 add 30
9 mul 30
10 Power 30
11 sub 30
12 div 30
13 inc 30
14 dec 30
15 by2 30
16 [illegible] 30

(crossed out:) 59 c0 c30; c1; 60 c2; c3; c4; c5

ONLY 20 Insts used (of 21 target insts) — of 73: 2 insts are simply examples of how to write programs in FORTH

Census of inst use! (Problem #) — 1 thru 6 — $A^n B^n$

	Inst								
0									
1	defnp	1 2	3	4	5	6		10	
	2toD	2	3 3	4 4	55				
	grt	2	3	4	5				
	c2	22	3	4	5	6			
	endnp	2	3	4	5	6		10	
	boostq		3	4	5			10	
	Dot D		3,3	4,4	55				
	bsf		3	4, (4 4)	5,5,5 [5]				
	from D			4,4	[5,5] [5,5]				
10	del D			4,4,4	5,5,5 [5,5]				
	by2			4	5 [5]		9,9		
	cpnb				[5]				
	c1					6			
	calltp					6		10	10
	mvdsk						8		
	c4						9	10	
	c3						9,9	10 10	
38	cpn (= cpnb)						9		
	Exec						9		
	dec							10	
40	c5							10	

boosted for prob 1 thru 6

boosted for 8 thru 10

No Problem #7 exists.

8 thru 10 Tower of Hanoi

4TM ⑨ Scirus.com/espappl Comparisons and Convergence Rates

00:201.18: AHA! This Partial ordering Net can be used to summarize data (corpus): just as I linear ordering did in PPM! We can devise a rpm ("2-3 Tree") to put new nodes in the Graph!

Not so fast!! ☺: Each node has a "region around it" (Tree) for a given fixed node, R_i's common regions w.r.t. each Node are partially ordered.

.04 Given a node N_i, every node N_j is partially ordered w.r.t. i (including i — i has a trivial graph in common with itself). So we have relation $N_j \overset{?}{\geq}_i N_k$ ($\overset{?}{\geq}_i$ means ordering w.r.t. Node i's context) for all i, $N_i \overset{?}{\geq}_i N_k$ if $k \neq i$. So this partial ordering is a ternary relation.

Nor, if $N_j \overset{?}{\geq}_i N_k$, that tells us little or nothing about $N_j \overset{?}{\geq}_l N_k$ for $l \neq i$.

e.g. Say N_j's V_R has nothing in common w. N_i (i.e. they have distant tokens from N_i) Yet N_j's N_R could have a common token at their nodes.

.0 So, to approximate all of $\overset{?}{\geq}_i$ info for all i, whenever a new node occurs in the corpus, we have to place it on each Graph of every node in the corpus: i.e. each Node has to have its own graph. || Actually, we only have to insert into Graphs of Nodes that have same Token — other insertions are trivially uninteresting. || Also, we are only interested

.14 in the "down" (toward leaves) direction from the Node.

While .04 - .14 refer to $\overset{?}{\geq}$ type partial ordering's assoc. graphs: There is one partial order graph (i.e. the Graph) corresponding to the functions in the corpus.

A Possl. way to use the info about contexts. Say we have a $\left(\frac{N_i}{\text{node}}\right)$ in its assoc. (toward leaves) context. (Or, one can go up toward roots, so context is toward Root)

.6 Any way, we have the Node & we want to know P.D. of nodes just (below / toward) it.

Each context from the root of the corpus has a certain amt. of overlap to N_i. (it may be zero)

.22 The "overlap" can be the no. of Nodes that are ① same as those of N_i contexts ② are connected to N_i via "same" nodes.

Each Context has a "set of tokens that it has" followed by its Node point.

Each context has an "overlap No." w.r.t. N_i. Assoc. w. each "overlap no." is a "universal" "utility function (Must be obtained by correct trying out past).

.26 To get a prediction at Node N_i: we take a wtd sum mean of the p.d.'s induced by each of the contexts. The wt of a context will be the product of 3 factors:

.29 ① Some (universal) utility function of "Overlap no." (.22) ← "Overlap Number"

.0 ② A Laplace or Dirichlet's rule involving the case counts of each token w.r.t. that context (includes "pre-corpus")

③ α^{+N} : N = total no. of tokens predicted by that context (including, perhaps, pre-corpus) and α is the (Geometric)

.32 mean pc per symbol of the corpus up to now.

The product of ② & ③ is the amt. of pc saved by using that context (saved w.r.t. ③) Σ'm not quite sure of this ... we may simply have to find α by "correct trying out past".

N.B. we can pool all data from Contexts of Σ Overlap No.

↑ Saves a lot of time.

At present, I don't know a practical way to get the overlap nos. of all of the contexts relevant to a particular prediction at a Node.

I can think of 4 exactly obtained context types, however. They are:

.48 1) PPM with normal (RPN. 2) PPM, but RPN corpus is run backwards.

3) ① but order of binary functions in stands is reversed 4) ② but order of binary functions in stands is reversed → 203.40

} see 203.17 for 4 more context types

9

8/17/04

4 TM

0: 200.40! By doing PPM for RPN ~~functions~~ — both forward & backward, one would get somewhat different branchings

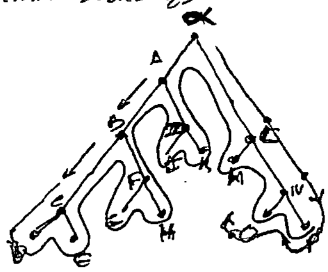
So forward its w/ alphabetic order (α for.) α A B C D F

Backward its α ~~Ben~~ L N P O M A I K J B F H G C E D

So its quite different from Backwards of t. forward RPN

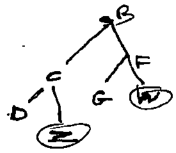
So it looks like using Both forwards & backwards drawns will ↑ no. of predns found

It will ↑ no. of apparently similar nodes



A different approach: Consider node B, we can "match" it w. all other B nodes in t. corpus to some extent. ~~Each B node in t. corpus will match~~

A node



will match up to t. pts (Z) & (W). So for every B node in t. corpus we can say how far out we can go in any branch & still have matches.

So assoc w. any 2 B nodes there will be a common match region, consisting of Branch, but usually parts left out. These common Branch will be a function of t. parts not assigned. In t. above case we have a function at $f\binom{E}{Z}$, $\binom{H}{W}$ positions

Suppose we had a way to group, relate, each subtree w. every other sub tree that had same top.

How could we use this structure/relation to assign pc's to cands?

6 ~~Start at top nodes, or~~. This constitutes a "PARTIAL ORDERING" of the contents of any Node in t. corpus. AS such; It can be represented by a Boolean Graph. (Also Note equivalence by commutativity (or any other symmetry) of functions. → 202.00

18 Woops! t. discussion of .00 – .10 may be wrong! The modification I have described is not running t. dcmsetry backwards, but inverting t. order of (binary) functions! ← which is certainly an interesting xform on which to use PPM!

0 But tree backwards would seem to have less good contexts (i.e. good for predn),

22 P O N M L K J I H G F E D C B A α (.30)

(SN) On trying Existing PPM progs on various corpi: I think standard pgms take ~~each file~~ any kind of file, expressed as a sequence of Bytes, — So I should be able to test various ideas w. ± 4 bits precision) E.g. I could try it on ~~Eng~~ Backward English, Backward Lisp, ect. I prob. want to remove formatting instructions – like line feed, ect. Perhaps represent each Lisp instruction as ~~a~~ a different 1 byte string.

0 .22 ? In looking for Matches, t. search could "take advantage" of Commutativity of certain functions. (Also associativity, distrib. etc... perhaps).

N.B. The Magnitude of speedup of such is Enormous! Going from raw 8 bits/token to 4 bits/token (of L.Z.) is square root of CJS. ~~Going~~ 8 bits to 2 bits/token (v.g. "PPM") is 4th root. For search of 10^{10} trials; → 10^5 → $10^{2.5}$ ~~Nevertheless~~

Hvr, a factor of 10^{10} (Jurgens OOPS) does involve some compression ... just how much?

Also 2 bit compression involves long Corpus (which ~~we~~ we don't have). ~~we~~ raw Bzip2

I could start w. pure rote search for OOPS $\left(\frac{1}{23}\right)^{10}$? factor of 3.3 $\left(\frac{1}{73}\right)^{3.3} = (1.6\times10^{6})^{-1}$

$(23)^5$ (factorof2) $= (2.07\times10^{9})$ which is ≈ what OOPS does. So we get $73^{1.66} = 1.24\times10^{3}$

8/16/04
4TM

00:199.40 That we can go very far. The system will learn to do induction in this particular way. — by concatenating useful ngms. It will discover ways to express many (perhaps all) regys in this form.... So PPM would be a v.g. update system!

To work hard problems, it would have to have an adequate set of cones. of this type. TM could develope a set of this sort, but it would perhaps (probably?) involve a set of heurs difrnt from those used by the Trainer (?)

For heurs that Trainer is conscious of: Could they always be put into the form of concatd. (cones / ngms)? Perhaps best way to understand this Q is to write various TSQ's & see what heurs they have & see if I can always put them in form of concats of cones.

My Guess is that it can't always be done: That a "most general" PPM is to be formd of compositions of functions (Primitive Rec functs perhaps).

0 Can one always express any funct of k vars., so that it is in the form:
$f(\ \ ), x_1, x_2 \cdots x_k$: i.e. the args all follow the function dcn.

Well we can have a function on expressions that does just that: It looks at a (large, complex) function tree w. args. It has a way of refering to any no. of nodes of this tree. Say it refers to k dif'nt nodes:

15 If Tr is a tcon of the original function tree & h( ) refers to k of the nodes of Tr,

16 then h( ) Tr defines a funct of k vars — i.e. that tree, w. those k vars replacing those k nodes of Tr.

18 So looking at a large corpus of function trees & finding common subtrees is equivt. ← Finding "generalized" commutativity would also make many functs identical a:: ssz. (.31)

19 to deriving functs in the form h( ) Tr (.15, .16)

20 Is the problem of .15–.19 the same as finding CF Grammars for a corpus of function trees?

21 So there are 2 interesting kinds of "functions": One is h( ) (.15–.16) & the others + DO loop

First takes a real & integer & a function & a function as inputs & creates a function:
$F(0) = \alpha$ (real) for i = 1 to n $F(i) = \varphi(F(i-1), i)$: output is $F(n)$. (See (4TM: 196½) or Kleene)
~~[illegible]~~ There is 1 funct (of 2 vars) i.e. $\varphi(\ ,\ )$ — this is generalized as Kleene.

So, given a 2 input, 1 output (real) function, a real α & an integer, we get $F(n)$.

→ This suggests that it may be possl. to get a Universal induction using PPM & a "common subnet search" a/o some kind of PCG-disc Algm. | Also, I may need the 2 special functions of .21

On the other hand, it would be well to continue the discussion of the Goal of 199.20–.26

0 An alternative, more intuitive (?) way to get Universality (& Recursive).

1. (.18) AH! Use GA to find Algms for fast detection of common subtrees !! First do it by hand, to get good set of "primitive" operations. — Perhaps the system can BOOTSTRAP itself in this problem! Perhaps use a lot (WEB) computers to constantly work on this problem. This problem may have had others work on it. I think it's known to be "NP complete", but still short approxns may be available.

If there were many people working on the Phase I or even PSG disc, "common subtrees" is a major sub-problem & it would be possl. to get some fraction of the community to work on it.