1/4/07
ED

500

Learning During Search (.04)

SOLN. of PC_1 v.s. PC_2 v.s. PC_2' Problem

.18-.23
.34-.35

Jess: Sat ~ noon; maybe phone.

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

00: 49:40 : In either case, one does an exhaustive search over all cands w. $\frac{1}{PC_1} \leq T$: i.e. $PC_1 > \frac{1}{T}$ i=1 or 2

~~w. cutoff = $\frac{cc}{PC_1} > T$ i.e. $cc > T \cdot PC_1$~~ $\left(\begin{matrix} i=1 ; i'=2 \\ i=2 ; i'=1 \end{matrix}\right)$

No! I think we have to use the simple PC (PC_1 or PC_2) to decide on pc cut-off and on $\frac{cc}{PC}$ cut-off. So, it looks like we use PC_2 to guide entire Lsrch — Then, if R (.07)

.03

.04 (JN) Since PC_2 has the entire History of TM trials as its corpus, it will be changing during a srch ...
So we'd automatically have "rng during Lsrch" if Lsrch is guided by PC_2 (which it should be, I guess!).

.07 (.04) we want better PC_1, we continue the Lsrch (still using PC_2 for guidance) & look for larger PC_1 ——

This Lsrch is a bit unclear in my mind. If PC_2 completely guides the search, what is point of PC_1? .03, .07 would seem to be the only way that PC_1 enters!

.10 So we use PC_2 to get a (sat sequence) of codes for the corpus. PC_1, hvr, is used to [redacted] wts (for prodn) of these codes. The traces of PC_2's srch, that resulted in the highest PC_1 codes, will be used for **FEEDBACK** to improve PC_2's performance

So the gist of it is: PC_2 guides the search to find codes for the Corpus. PC_1 is used to evaluate (Weight) these codes. I still feel a bit uneasy about this, hvr. PC_2 seems to be too important! in the process (relative to PC_1, which is more what we're interested in.)

.18 While it is to some extent, clear as to what PC_2's "corpus" is (i.e. DATA), it is less
.19 clear as to what PC_2 is the probability of. Also unclear is how PC_2 gets computed from its "Corpus".
.20 PC_2 is the guide pc in OZ srch: $PC_2(X)$ (string), is the probability that X is the shortest code (for PC_1's corpus). The PC_2 would be obtained by integrating over density $PC_2'(X, L)$, which is the probability that X will be a code for corpus of length L.
.22 (That's a fantastic form of PC_2). How PC_2' is related to its data is still unclear. .34
.23 Not so bad after all! That's it! — see (.34)

Re: Recency hvr! This would mean that TM would have very short Memory Span! Not so good! Early ones used would be forgot if not used repeatedly! Well we have to be careful & not make the Recency "too strong" an effect. We can have many versions that are better, until we get to last few problems solved, & have much left in pc for them. We have to balance out the 2 effects of Recency!

.30 Examples of PC_2: (1) Recency (2) Use of Recognition function to reduce version. cc. — I think this is a former PC_2 — e.g. PC_1 would not recognize that a new QA was identical to a previous QA. (?).

.34 (.23) AH! That's it! PC_2 is the p.d. for the OZ problem of finding O^n of max pc_1 in given CB.
.35 That's what (.20 — .22) is about. All of the DATA bears on the computing of $PC_2'(X, L)$ d.f. of (.22)

So it would seem that we've reduced the problem of updating PC_1 by the problem of updating PC_2 (or PC_2'). I feel that we are much ahead", but I can't yet say

501

1/5/02

ID

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

ABod ABCDE

$\int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} dx$ $\frac{x^2}{2\sigma^2}$

501

00:500.40 : Just Why! ⊙→ Perhaps: Even w. a mildly good soln of PC_2', one could do rather well in optzng PC_1 & get a good [redacted] QATM.

The problem of UPDATING P_2' is an induction problem ∴ an OZ problem which gives us P_3' ... and an infinite seq of P_i' $i = 2 \ldots \infty$. Usually it is possl. to loop this infinite seq.

Formally, any P_i' problem [redacted] can be (somehow) viewed as a particular QA problem, so we could have QATM working on P_2 or P_2' or P_3 or P_3'. — The P_2' looks like an ordinary QA problem.

.08 ⊙→ My impressn of the Way I had expected to deal w. the "infinite regress" [redacted] and "T. Loop": That I would do it open loop w. only say PC_1, PC_2, PC_2' and do PC_2' "by hand" — or simple

.10 direct intuitive [redacted] induction — or do (PC_2, PC_2') the OZ problem, by some reasonable OT's. When TM had enuf training & cc in proper areas QATM would be capable of usefully working on the PC_2 or $(PC_2, PC_2')(\equiv OZ)$ problem.

.13 ⊙→ Another Way of Dealing w. the infinite Regress "is to stop the regress by making [redacted] "$PC_2' = PC_{2H}$" We do this until QATM is "smart enuf to compute/evaluate PC_2 (or PC_2'). /ie, enuf training and cc

I think (.13) was mainly what I had in mind, (?? i.e. .08) Tho also, the idea of using one or more v.g. OT's that were fairly General, was also Considered, to be quite Reasonable.

ON second thot: My present impressn. was that (.08 — .10) was [redacted] preferable to .13 — but I'll have

.20 to look into this: Both are to be followed by the Recursive form, when QATM has had enuf relevant training & cc.

.21 The Recent Discn. of PC_1, PC_2, PC_2' act. [500.18–.23, 35–36.] seems very nice! It Adds on to the purely formal, "exact" equ. of 419.. (on Maxzn of approp. opt of OM) So we have a larger formal system, & More opportunity to [redacted] properly formalize Various Heuristic Methods of induction & prob. solving.

— It ties up a bit more of the formal model in a neat way! (Also note 500.04 on Lrng during Lsrch.) → Heuristic & problem

W.r.t. lrng during Lsrch we do get closer to a complete formal Model — Also a trainto this method's being close to optimum — "Since we [?] [redacted] "Thm" that "If All mbrs in the pd (PC_2) then Lsrch o a I optimum within the factor of 2."

.28 [Woops!] What happened to the older soln. to the OZ problem in which we search over O.T.'s? This was proved to be optimum. "within Maybe factor of 2" [for all members in P.D."

.29

.30 → This PC_1, PC_2 model is for INV. problems! It would seem to work for [redacted] finding Short Induction Codes, hvr, hvr.! What about the "proof" of .29?

.28 DOES seem like a SERIOUS Conflict! ⟶ [See 502.28 — .32 for what looks like resoln. of this diffy] [Lots of Q on Validity of this]

.33 Thinking about it now, it seems clear that PC_2 is not an ideal guide for Lsrch for short codes: It is (as of new) (as of now?) oriented to finding Codes rapidly (a INV Problem) — NO! 500.22 It Does get $PC_2(x, L)$: so it can be oriented toward short codes. In the present case, [redacted] since x is unknown, $\ell(x) \le L$ is known — so Let $PC_2(x)$ be simply the proby that x is a code

.37 for "the corpus"). So a Q is: "What do we mean when we say "$PC(x)$ is the proby that x is [redacted] a code for the corpus? Superficially, this is a well-defined, partial recursive Fnct. (502.00)

1/5/02
IP

◄ HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 : 501.40 : Well, this PC(x) is to be obtained using very limited info about x.

.01 Or, Consider prob'y that x will code the corpus in time T. (see cc cp). (.05)

.02 [SN] That method in my "Letter to Levin" about how to use a General (non monotone) U.M.C. for seq. predn. A main criticism — that it, like Cover's "Extension prob'y", is very time consuming for most sequential predn. Hvr. my method could be usefully used for evaluating the "Discrete Univ. D.P." (rather than the Continuous).

.08 (OP): N.B. My analysis of Lsrch for OZ probs suggested that "usually", it wasn't very efficient because derivn of an O.T. was expensive; That in practical applicn, one

10 would prob'y find a few O.T.'s & use them every time: that the real problem

.11 was getting a good General O.T. In this case, the PC₁, PZ₂ Method of (500.18–.23, .35–.36) may be regarded as a v.g. O.T. — V.G. for Predn problems (& perhaps for OZ probs. in General).

More exactly: Say one has a bunch of OT's : Z_i (i=1…100). Would the wt. of an OT be on its effectiveness on various OZ probs thus far? Actually No! Like normal ENV Lsrch, the normal Lsrch OZ soln — does not lrn. As w. Lsrch for INV, we would need a "(conditional) P.D." to guide choices of OT's. Since OZ problems can differ a

20 Lot between them, it's hard to compare Scores: [I considered normalization to help in comparison, but I'm not sure this does a complete job.]

Comparing the cp' of O.T.'s might be best (or at least "Reasonable").

[PC' is the P (x, G, T) d.f. the prob'y that soln x will get Score G with cc = T — I'm uncertain of exactly what argt. is, here.]

.28 Actually this PC₂, PC₂' treatment was for (INV problems, not OZ problems)! False/it was for OZ probs. No! (I think)

Predn problems are OZ problems but because the Score of a trial has a particular Simple form, we

30 Can treat them like INV problems, as far as Lsrch is concerned.

With the Lsrch for predn problems, we have the usual upper bnd on cc for discovery of the particular codes.

.32 — So the argt. about PC & PC' for induction problems may be correct! So (500: 18–.23, .34–.35) May be O.K.

So 3 kinds of problems:

1) Induction solvable by direct Lsrch — also often solved as OZ problems (see 503.03–.05)

2) INV " " " " but usually as OZ probs

3) OZ solvable by Lsrch, but I think this is not usually the best way: see (.08–.11)

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00: 502:40 ! There are t. Q's of 501.33 ff specificity 501.37 - 502.01 .
Perhaps in $PC_2(x, L)$: even x, "L" may not be so trivial to compute! "L" is $\lceil -\log_2 P_{C_1}(x) \rceil$ and may not be easy to compute.

.03 Given a long string "a", : to find short codes for it, we usually don't start w. 0, 1, 01, 10, 11 etc, but we analys "a", look for correlns, digrams, ect. — a H.c. process. — T. Q is

.05 Usually, "How can we find a shorter code for "a" or for a code for a ?" i.e. t. "Code shortening problem".
[look for larger repgs in "a" or simply repgs in "a code for "a"]

So: T. problem is: Just how is an induction problem solved? In simplest case, given string "a" to find $x \Rightarrow M(x) = a$ & x is short. If we regarded

.10 this as a INV problem (w.o.t. "x should be short" cond.) we'd simply do Lsrch, & get a (usually) short x. Here we used a priyd $2^{-L(x)}$.
Now, we use a diffnt Pd : $P_2(x)$ to srch for x & it presumably, P_2 is oriented toward both short x & low cc solns.)

.15 In a normal Inv problem, we have FC) that looks at t. problem dcrn & devises a pd for Lsrch. Ideally, this pd will change during t. srch because more(truth) will become available to it, to base its search PD. on (this amounts to "Lrng during Lsrch", but it also means that x. (traces of this recent)) solns, etc

PD depends on an expanding "context".
.20 For an induction problem, FC) also looks at its dcrn & devises a PD for soln via Lsrch —

.21 A point. Pd changes during Lsrch in a way,
.15 - .21 is repetition of recent stuff: Is it recent cc correct? Is it "Adequate"? In both cases, because of t. changing "guiding Pd", we cannot know t. cc necy to solve a particular problem with a particular soln. If, t. Guiding Pd changes in a "reasonable" way, we expect that $P_2(\text{soln})$ will be $\geq P_1(\text{soln})$
In both cases we started w. $P_1 = P_2$ (IE Mind) & P_2 gets modified so we hope that will be true.

.28 I seem to have lost t. idea behind P_2, P_2' : 500.19 - .23, .34 - .35] is t. most recent start of it. One trouble was 500.22 : i.e. $PC_2(x', L) \equiv$ probty that x will be code for corpus is of Length L.
.30 If $L(x)$ is immediately known — so what is 500.22 about? Say we are looking for a code for "a"
aprip
$P_1(x)$ is t. aprip that x is a code for "a". This $\approx 2^{-L(x)}$

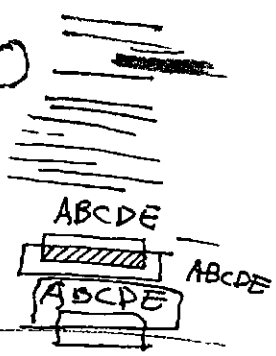
.32 Perhaps investigate t. application of Rissanen ideas to finding good O^{n+1} : We have Apriori $P_1(O^{n+1})$ d.f.
.33 [This is AZ/MI d.f.] Now want $\geq P_1(O^{n+1})$. O^{n+1} (aprip) is Max (Eq).
F( ) looks at t. QA corpus & traces of run that led to, & previous O^t (w. prec. QA corpus) : from this, it
.35 — obtains a PD over possbl. O^{n+1}. If then makes trials in a this (last PD order). Note that t. PD
(changes after each trial.) — This is a MAXIMAL GREEDY Algm. No "Look Ahead" at all ← No "LA".
PD_2 (.35)? : $PD_2(O^{n+1})$ is t. probty that a particular O^{n+1} will be t. best fit (continuum(.33)?).
$PD_2'(O^{n+1})$ is a kind of "derivative" of PD_2 — i.e. PD_2 is a kind of Integral "of" PD_2'.
PD_2' is a PD on t. Gen of any O^{n+1}. PD_2' gives for each O^{n+1} a PD on its Gen (of .33)

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 ! 503.10 : My idea was, Rest Since PD$ was an ordinary conditional probability, it should be possl. to
.01 invent a reasonable approxn. to it in some Standard way.
So: 503.32 – 504.01 is one way of "optimizing" ONTI.

.03 Another way would be to regard it as an OZ problem & use "Lsrch for OZ probs".
We start w. a FC) that looks at the problem & gives a d.f. over OT's for it.
As our trials progress, FC) can change its mind more & more on the D.f. over OT's,
.06 So we may make early jumps from one OT to another. [These OT's can use "LA"]
If design of FC) an OZ problem > I think so. FC(OT_i) is the prob that OT_i is a "best" OT for the current problem

Post LA SN On "Look Ahead": The plan was to use a first order TM w/ "Look Ahead" (= LA) = DYNAMIC PGMING
Then teach TM to work LA problems, then have TM_2 = TM_1 + (Recursion, SI).
.10 My idea was to do this only when TM was pretty smart, so it could really understand & usefully
work on LA problems (including its own). ↳ maybe vanary: (Sect. 12)
.12 Actually we can have TM_1 be able to work very simple LA problems — then w. recursive SI,
it could apply these simple techniques to its own problems. So no need to wait for
TM to be v.g. at LA! Have it use even elementary Lrnd Sbstn on SI.
Ideally, we should do SI as early as possl. so TM could learn many techniques —
for simple (or complex) prob solving that it could use immediately in SI.
.17 → Perhaps TM should immediately (at birth) be able to use any kind of problem soln.....
.18 Instead, for SI ... for solving its own induction problems.



.20 SO DO 503.32 – 504.01 in more detail to see if it's meaning full, & practical.
(→ Do Sect w. .03–.06)
In back of mind, look into optimality Things: Both from pt. of view of Lsrch &
from "All info is in PD" question on optimality: Note that latter does
not include LA theory, so it can't really be "optimum".

SN Perhaps don't try (directly) for Optimum Pgm, but Get Something that will
"More or less" run, & improve it (by recursive S.I., say) [Hunt. Main Reason I've been interested in "Optn" is not optn itself, but as a (Hopefully Easier) Method to understand & overcome the overfit problem]

In 503.32 – 504.01: How is S.I. introduced? Can
.30 I do it "at birth"? Will this give "LA" in a most General way?

I have been thinking of Lsrch for OZ problems as not being a particularly good method
until v.g. OT's were available. (→ See 502.18) On the other hand, there is the "Heuristic Optimism Conjecture"
that "If all info is in PD", it is ≈ "Best" (except for LA!) ← The OT's can use LA! (?) TM

SN Try QA in a Sequential Corpus! This simplifies several things a lot!
One, is Model of probability relation Q→A — use of 11 codes. — seems much simpler
than "Unrestricted QA" Model. — (Also easier for Audience to understand.)
Also correctness of answers in "A" may be easier (?).

→ 505.00

1/11/02 ID. #505

Recursion (.14)

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00: 504.40 ; 503.32 – 504.06 ; 2 Methods of optim $O^n( )$.
Both seem to have serious faults, but may be "good enuf" to get to $TM_2 = TM_1$ pts. (serious S.I.).
Other ways to do optz $O^n( )$: GA (491.00). ; Perhaps ANN, RANN (see .03-.06)

.03 N.B. Many Optzn techniques work OK for a small corpus (or "small problem") but become unwieldy w. "Large problems": In using L search for induction using L search for OZ probs, we'll usually break up corpus by means of "R_x" function: This makes problems much smaller, so that GA, Neural Nets &
.06 RANN can be usefully employed.
Recurrent

My impression is that 503.32-504.01 (treating induction as an INV problem); is a method into which I can express many (say) techniques. So, perhaps widen that method up in a bit more
.10 — detail: Then Go back to writing TSQ's: & see how well it does on these examples.
498.10 lists some TSQ's & interacts. It mentions Recursion as "not having details worked out yet" ....
.12 my impression is that it is worked out "not bad". (see .13)

.13 (.12) otherwise Recursion: T. recursive funct's defined in Lisp, can be of form: $f(x) = g^{(n)}(x_0) = y_0$ (fixed form)
.14 then $f(x) = h^{(n)}(x_0)$. The $f(x)$ is defined by $h(\cdot), g(\cdot), x_0, y_0$.
We can define a recursive funct by many ex, I ex 00 & the following funct is not recursive; dual otherwise).
Then $h(\cdot), g(\cdot), x_0, y_0$. The pc of 0 or 1 at this point depends on its val from
of previous use in defns. (Lap's role).

Actually, recursion has a more complex defn than .13 – .14, (even to do fine $x!$):—
.20 but the general idea in .14 – .15 is adequate to deal w. more complex recursive defns
$f(x) = x\,f(x-1)$; $f(0) = 1$. There are 2 functions in valid inv. defn : $\tilde{g}(x) = x-1$
$f(x) = h(x, f(g(x)))$: we do need n s $g^n(x) = 1$
Then if we need $f(x+1) = h(x-1, \ldots$ rather complex! I have written about this within
the last month, hrr. There is a need to look at KLerner's work for good general defn
of universal p.r. funct. Also look at McCarthy's LISP treatment
of recursive defns. Look at various defns of Ack funct. My formalism should be able
.30 to describe Ack funct. Just look at the functional eqns. & the "Boundary Values".

Try a quick review of work on this thing: ~ Since ~Dec 99/01.

P1 vs P2 ? Refs: 499.0 ff; 500.18-23, .34-.37 Does continuous to (at least) 503.28-30 – At which pt. I seem lost!

A "new" Prob: The original P_1, P_2, P_2' idea was for OZ problems: it was part of the Mind Corpus Thesis.
$P_2'(t, M(\cdot), G, T)$ is (no re stated) prob density $\left(\frac{dP}{dT}\right)$ that OT_t will, wi
OZ problem $M(\cdot)$, obtain $M(\cdot) = G$ in time T.
At the time I "solved" the Mixed Corpus Thesis, I felt that Induction problems all had well-defined solns.
— That Phase were essentially INV problems. → Now, it appears that induction problems (P_1); → P_2 → P_2' which has 506.00

IP

Jan 13 Mon 14 Tu 15 W 16 TH 17 Fri 18 Sat 19 — Winter Break

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 : 504.90 : an induction problem, ... was an infinite regress. 501.00→.20, dead end & infinite regress by (ultimately) looping... Not all induction problems have a clear "loop" soln. QA induction does, hrr. — Since any induction in the regress can be made to be a "QA".

On 501.21 I felt that this looping was fine & constituted a good theoretical soln. for QA induction problems.

501.28 Then remembers that "ordinary" OZ Lsnch is "optimum" for OZ problems — so it should be best for all induction problems. → On the other hand, the "looping method" seems to be close to the way that people work these problems. — Can I show that in some sense it is V.G. or "Optimum"? [501.18–.23, .34–.35]

.08 → I'm still not certain about the "looping" & other details of that method! 501.00–.20 discusses it a bit—

.09 → but not enough detail to write an actual pgm. → (see 510.11)

10 Dec IND — So: I have 2 methods to solve induction (IND) probs: The Lsnnover OT's has some difficulties. some difficulties. The loop method has difficulties (.08–.09). Which method is "Best" is unclear. The loop method seems to be closest to human, but has little in the way of an optimality thm. The Lsnch does have an optimality (quasi)-thm.

As a practical problem, I have the induction "problem of" an optzn: I can use either [redacted] or both s/o & new methods to solve the problem.....

A main problem! Can I devise a formalism for soln., ∋ any conceivable task that a human might be in to solve the problem could be expressed in this formalism?

.2 — ① The Loop Asmtion (in which a QA can be any problem) seems adequate.

② The "Lsnch on OT's" method seems adequate since [redacted] an OT can be any thing.

SN .22 Why [redacted] isn't "P_2 info" relevant to "P_1 problem"? During the course of the investigation, more info becomes available than just the "QA's to date". Should not this info bear on prediction of [redacted] Ans? → see .28–.29 for possible resoln

[1·15·02] Couldn't TM write TSQ's for itself? This would give P_2 info but, presumably, not P_1.

On the other hand, if TM wrote a more (1 (10^10)) Googleplex..... this could bias, conceivably, its extrapolations of many sequences, if we included it as part of "P_2's corpus".

.28 It may be that one (must) can include parts of P_2's corpus in P_1 if it was created in type 2 comb

.29 regarded as part of a code for [redacted] C_1 ("C_1" is P_1's corpus; "C_2" is P_2's corpus).

.30 .28–.29 may be the resoln of .22.

GPD .31 When I worked on the "Mixed Corpus Thm", I had a "General PD" that solved INV probs, OZ probs, [redacted] Sequential induction; Bag induction problems. This GPD was very much like (if not the same as) my $O^n(A|Q)$. I'm not sure of just how I expected to solve the IND problem at that time — Could I have simply put in a direct Lsnch was adequate? Well, I may have been

→ Thinking of the induction problems as always being incremental so that somehow the CJS was not very large.

The Method of "incremental induction Coding" that I had in mind, was rather simple— That the incremented can find Codes for the incremented corpus was to make increments on the codes for the unincremented corpus! — Was this about it?

507.00

1.15.02
ID

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 : / Sliding / More generally, if s is a short prgm for c then we try $s^\frown x$ as trial prgms for $c^\frown a$. If we can find a suitable x, then find another good code for c,

.02 (e.g. s') & ... This is "Backtracking", i.e. try $s'^\frown x$ as codes for $c^\frown a$.
.00–.02 may not work for certain kinds of coding situations — but we may be able to find coding (situations/environments) in which it does work well.

[Perhaps more generally: find $(s(c^\frown a)) = f(s(c))$ so $(s(c), a)$ is a input to f), This gives a p.d. for trials for short codes of $c^\frown a$.

.10 [1.21.02] On the Guiding P.D. for OZ Lsrch (MxtCorpus Run ≡ MCT).
In MCT, we replace (I think) the guiding PD for OZ by a Cond'l P.D. — conditional on the problem dscrp. This Cond'l PD is a P.D. on OT's — the pc that each OT will yield the Best result in given CB. So how is this (CPD) updated? In the original MCT analysis this was done by the PC, PC_1' trick: PC' (OZ problem (no CB needed), x, G, t, T) [Pcutof]
This is pc that OTs would for problem M(x), have output G after cc = T.
Updating PC_1' was an "ordinary" induction problem, & I felt I had this under control — (There is now much Question about whether this is true.) . Updating PC_1' is more generally an OZ problem (as are all induction problems) —— So it can be viewed as a recursive defn, w. various approxn.

.20 techniques used for "Boundary Conds" of the recursive defn.

.21 (RLPD) In .10–.21 (& many other TM problems) I write "P.D." — But what I'm dealing w. is not an ordinary PD, but a "RLPD" resource limited P.D. One way to define this is: The pd. one gets from each of them, using a certain CB. For each CB, the output for certain Inputs will be "Undefined," or we get only lower bnd on pd, or "Normed value of lower bnds" etc. In the case of ALP, the "pd" we get isn't even normalized for a finite CB.
(So perhaps it is if we include "undefined" as one of the alternative outputs).

.30 In .10–.21: Guiding "RLPD" for Lsrch in OZ prob.: The system really isn't so bad (time consuming): Since its a Cond'l PD, I have only a few OT's w high pc relevant to the problem! However For a "juvenile" TM, I "exercise" lots of OT's on each problem because I don't have good ideas on which is most relevant. As result of this "exercise", I get much better ideas about relevant OT's for any particular problem.

.31 A "mature" O.T. only has a few (or one) OT for a given problem. [See 507.30]

1.21.02

ID

HARDWARE .11

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00: 507.40: Also, from "Exercise" info, TM is able to [invent new OT's] that are likely to be relevant to a new problem.

.-3 [SN] In t. Monte Carlo generation of trials via A 2141: for each symbol choice, there will be a set of Legal symbols. For each legal symbol, we will store a "subcorpus" of its prev. regular corpus. T. "Subcorpus" consists of t. integer that is t. case count of that symbol. N_j is t. case count of symbol type j. If symbol types 3, 5 & 7 are legal at a point, Then we get a random no. N from 1 to $N_3+N_5+N_7$: This will chose which of t. 3 symbols to use: If $N \le N_3$, we chose symbol type 3, if $N_3 < N \le N_3+N_5$ we chose

.10 N_5; if $N_3+N_5 > N \le N_3+N_5+N_7$, we chose N_7.

.11 HARDWARE: Brewster Kahle: He normally buys H.W. that is "off t. shelf": in hy volume production: It is very cheap & because it is a "commodity" can be reliable. Before buying a batch, he will buy samples from several dif'nt batch types & test them for reliability. Examples are 500 MHz H.P. PC's: 160 GBy Disc drives ~ \$250 each. $\frac{13}{8}$ = \$1.56/GB.

Also, use of S.W. that is cheap (Pub domain if possl.) — & use of cheap

.20 pgms so they are replaceable easily.

Now, he has ~ 400 × 500 MHz HP's; uses 160 GB discs & 4 per cpu. Each cpu is ~ 100 watts. Is buying ~ 300 more discs:

20 TBytes ≈ "Libry of Congress" Books: He has ~ 5 × 20 TB of Internet: 100 TB. (It takes him 2 mo. to "sample" t. "internet")

Book copyrites are free up to 1923; To 1968 they are usually free: After that, every thing is usualy permanently copy rited (maybe 95 yrs only).

.30: 307.40: THE MAIN DATA Problem: Given eq 414.18 to make it m t. available oc — Hm., one is given all "traces" thus far & other hour info: So in this sense, it is a [conditional OZ problem]. T. problem is to use this info in t. best way to help maximize 414.18. "T. conditions" are t. prob. dcrn & all t. other info. In a "Conditional OZ problem", we want to find t. best pgm. to solve t. problem & we have this aux "helpful" info. This is a classic OZ problem, but t. P.D. on O.T.'s is historical not "unconditional", but is conditional on nature of OZ prob. to be solved, as well as t. historical

.36 info (traces) that TM has available.

So, I should watch how I solve these problems & try to put those soln's into t. form of .30–.36. "AM ANFANG" TM doesn't have many hours for problem solving (or for opt of 414.18) — but develops them as it Matures. T. Tsq. contains info on prob solving, in t. form of problems but t. trainer may insert hours directly, if it seems necy. For each hour, we want @ TM should be

→ Also @ usec. hours TM should be able to discover t. hour.

2/2/02

ID

Expo .19 IMPT idea for "REPORT"

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

:00: 508.90: A "Heuristic" is anything in TM that helps get a better soln to 4|4.18. — it's a DTer or part of an OT. T. main reasons we want TM to discover heurs, rather than have them transfer mech.

Them: 1) Ability to discover new heurs (of similar kind) is under more likely

2) TM is more likely to understand and "properly use" a heur it has discovered itself.

TM acquires Heurs 2 ways: ① Admits ② by discovery ③ later insertion by trainer

TM solves problems either ① Directly ② Viz "Hints". Hints can be very varied. Any info/advice that reduces prob solving time can be regarded as a "hint". Hints are problem-specific Heurs — but the better heurs are more general in applicn. a/o result in a soln. technique that is more generalizable. A problem related to a given problem may be a "hint".

:10 → Perhaps one of the "Great Ideas" in Mod Cog Sim (MCS) was that P.D. used for cond. OZ search was the same cond. P.D. that was used for all other things — including SI. — That there was other "Great" P.D. that was used for everything: Including SI.

This model was perhaps able to deal w. the recursive loop in the "definition" of probability! i.e. T: 4|4.18 problem is an OZ problem. This is solvable by a cond. P.D. on OT's: which corresponds to an induction problem. ($PD_2 \rightarrow PD_1$): This induction problem can be parsed to 4|4.18 TSQ (≡ Dancing).

This is a recursive defn., so it needs "Boundary Conds". It May be possible to give those "Bnd. Conds" in the form of some initial OT's that to be good — i.e. an initial # (unconditioned) P.D on those O.T.'s.

:18

:19: 2.4.02 Exposition Using Ordinary Lsearch for difficult problems is normally impractical — Lcosts too large.

:20 However, Conditional Lsearch, Lsearch guided by a conditional P.D. can be feasible if one uses a suitable tsp. — which is what QATM is all about.

A critical Q: After ("working on/solving") a QA, how is the P.D. modified so that it is pc for other Q's in suitably modified? Essentially, I want the new P.D. to be a "Summary machine" for the first QA. This "Summary Machine" is probably defined by the criteria discussed in "2 kinds of probabilistic...", (but not exactly the solns given in that paper.) (I'm referring to the Summary Machine(s) for induction on BAGS)

:24

:29 Quick Thots on this → Superficially: Lsearch for first (easy) (QA$_1$) is easy: we get "summary machine", — Then Lsearch for (QA$_2$) should be easy also — it does really has a "complete" summary machine.

:30 Trouble is one normally doesn't have a complete summary: ← (Yes, but for inadequate soln to an adequate tsq one has an adequate summary)

Thinking about induction on a sequential corpus seems about the same as trying to do this! It has same "Summary Machines" problem. The chances of data reduction much if one retains many copies (or "equiv"). So there seems to be a nice correspondence betw the seq. predn. problem & the QA tsq problem.

→ However, in normal sequential induction, we do not have a Carefully Constructed TSQ.

:36 ((This suggests that (likely maybe), for a "good" tsq, the "summary machines" would always be adequate for the next QA! This condition defines an "adequate" TSQ. [Corresponding Conditions could work on a sequential corpus, but for a seq. corpus, one normally doesn't have a good TSQ.

[Still, after an adequate initial tsq, would TM be able to continue on a less carefully designed tsq? This may be the Critical Q. I think it ought to, since it should have the proper (adequate) concs.

→ 570.00 spac

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.06 : 509.40 (509.36–.40 in particular) : 509.29–.40 amounts to successfully lrng. a conc. not for t TSQ!
A BIG Q is on t Adequacy of Lsrch to cover all possi. Heuristics. By using lrng. dury Lsrch, this technique is certainly enhanced: — but it still doesn't have Look Ahead (LA).
Perhaps only when we have recursion on search methodology (in QATM) will we get general LA.
→ In its "normal" operation hvr. it should be able to have access (for "lrng") to things that it lrnd (an tools).
I had been thinking that early (& perhaps always!) work on TSQ's would tell me how best to design TM to optimize 414.18: This may not be necy! — The actually, t. true goal is maxzn of 414.18: — so simple Lsrch is not ordinarily t. Best way: If ≡ on OZ problem & is best solved using ideas of 508.30 – 509.28: Having a frontend that choses an O.T. for t. input problem.
.08 So [509.36 – 510.08 (Noting 509.19 ff.)] may be a [adequate to solve t. TM problem] → (also 509.30 – 509.18 is useful)
i.e. 509.10–.18 may be very impt. (on recursion of t. defn of pc).
.10 Maybe Final Solution!

.11 Perhaps define TM in Recursive defn. of pc: Use simple Lsrch as t. "Boundary Condn" for recursion.
509.10 is perhaps t. best way to think about "SI recursion", & recursive defn. of pc in general.

.14 **So perhaps t. way it should work!** We start out w. 509.29–.40 i.e. we use Lsrch QA₁ ~~[struck]~~ Then we get a "summary machine" of QA₁, to guide Lsrch for QA₂, ect; recursively. This is t. elementary Model; we need a TSQ → t. successive summary machines do, indeed, have t. concs needed for t. next
.19 QA, ect.
.20 Next, we make a pd; assoc. w. t. OZ problem of optimizing 414.18, one of t. QA's:
T. method of .14–.19 corresponds to an O.T. It is given most wt. in t. problem of .20, at first, but as TM learns more about O.T.'s & how to use them, other smarter OT's will get more wt.
Note that all of t. ~~[struck]~~ pc's assoc w. OT's are condl. pd's (pc's) — conditional on t. nature of t. problem being solved.

.26 A Genzn of t. "Summary Machine" idea: Applied to t. QA corpus/PD t. idea is:
.27 a summarizing machine for t. corpus [$Q_i A_i$] $i=1/n$ is a ~~[struck]~~ PD ~~[struck]~~ (that is a good, usable) approxn. of
.28 $P(A_{n+1}|Q_{n+1})$.

Well, .26–.28 doesn't really solve t. "updating" problem. After a $Q_{n+1} A_{n+1}$ comes in (as Data)
.30 The entire $P(\cdot|\cdot)$ has to be updated. T. "Summary Machine" of .26–.28 just gives us a $P(A_{n+1}|Q_{n+1})$:
It really doesn't do any thing to guide t. search for a better $P(\cdot|\cdot)$.

.32 A possy: Say a good set of ~~[struck]~~ $P(A_i|Q_i)$ $i=1/n$ was known: — eg a set of strings that defn. possl $P(\cdot|\cdot)$'s.
We'd like to be able to search for strings that work for $i=1/n+1$, by searching over extensions of these strings.
.34 A genzn of "extension" is "Modifn". Even so it implies testing cands on t. entire corpus! (BAD!). (.345)
I guess t. way to deal w. .345 is to break corpus into "parts" that are "similarity" & equiv. classes.
More Generally: Say we have this O^n for t. $i=1/n$ corpus: for (QA n+1) how much can we localize t. change needed in O^n?

→ (511.00)

2/6/02 ID

REV: .26–.28 in particular: 510.32 – 572.31 is a more detailed discn. → 512

·HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138 9:05P 909 P and 2?

.00 : 510.40 : So we have way of recognizing any Q as belonging to a particular "part" of t. corpus, & having an assoc. "part" of O^n to give t. P.D. on A.

02 This breaking up of t. corpus is always changing as TM matures. / Some Parts are Coalized, others are broken

.03 into smaller parts. → (12)

Expo: "To deal w. a problem psychology, we do not try modifications of the fine structure constant". (Here this is an "Extreme Case"). Often it is not so clear as to what parms are relevant to a new QA. (≡ experimental Data).

Say we have already localized t. section (subdivision) of t. corpus for QA_{n+1}!
We could do a Global, ab-initio search for a new sub/stoch operator for that subcorpus; but this is usually very expensive.

10 How can we reduce search cost, "localize" "modifn." of t. old operator?

One method of Localizn: Trainer gives "indexes" to categorize QA's initially. Later, trainer doesn't give them & TM has to learn how to categorize.

.12 (.03) The changing of categories in (.02 – .03) is then lrnd using t. 414.18 Gorc: T. details of how this is to be done, are unclear!

414.18, & Breaking up of t. corpus are fairly certain to be in t. final TM:
T. idea of t. Conc. lrn is certainly impt.: Just how it's to be implemented, is not clear.
We solve QA_i: This puts soln (recognition + operator) in form of a new subop, & perhaps some new definitions. When QA_{i+1} comes in, we use t. new defns plus any "OSL" resulting from t. soln. to QA_i. — There are usually very many OSL's possl. — Also OSL's from earlier problem solns.

20 For studying & Demo of conc. nat lrng: use "d" problems: "S." problems are solved in v ways, but they are harder
determinstc Stochastic

Anyway: T. critical ideas seem to be: 414.18, conc. nets, "Problem types" (This last is least certain. Exists & maybe needs to be much modified).

But do .20 to start. ("d" problems).

Next in import, is t. idea of heuristics; Use of various techniques to make 414.18 rapidly (inexpensively)
Actually, 414.18 plus / srch Heuristics would solve t. problem! ↓ = Partitioning Corpus (17)

.26 so ("TSQ" Conc. nets) (414.18) / srch Heuristics: Of Heuristics: "Problem types" is one; & conditional Lsrch (& initially uncondl. Lsrch) are impt. srch techniques. Also OZ Lsrch: & ultimately, recursive improvement of

.28 Heuristic srch techniques, by making it a special QA. Idea of Lrng. during Lsrch.

30 → A vital (tho easily solved) problem is assignment of pc's to functions in a functional language.
.31 & That ≡ AZ414 is adequate for this. — Recursive extension to include stochastic functions.

→ Note (expo) that Lsrch using Conditional P.D. & TSQs is very impt / Very impt. OZ & NV

Lsrch alone gives Lcosts that are far too large: For 20 byte descr. of a soln, 2^{160} ~~...~~.

~~...~~ ~~1 GHz machine does 10^9 ops/~~ ~~..~~ ~~10^{39} sec for soln~~ $10^9 = 2^{27}$ Hz.

So we need speed up of 2^{133} to solve it in 1 sec. w. Moore's law, 1½ yrs to double:
so 133 × 1.5 ≈ 200 yrs. — For a 20 byte problem: much longer for a 40 byte problem.
1 byte of descr. = 8 bits ≈ 12 yrs. That's for a 1 GHz machine; 1 Ksh cost ~ 1000 times as much.
— But it reduces time by ≈ 15 yrs. To spend 10^6 times cost of 1 GHz machine now would be faster by 30 yrs.
A 2 GHz machine costs now ~ $1K.

10^9/sec ~ 10^{16}/yr. $2^{53.3}$/yr. ~~...~~ ... yrs. = 53 ⅓ bits = 6⅔ bytes so 1 yr to do 6⅔ byte problem. 1000 times as fast: 10 bits ≈ 6⅔ + 1¼ ≈ 8 bytes

This assumes an n bit problem soln takes 1 cycle time: If it takes 1000 cycles to soln time for Lsrch → ×1000.
or 15 yrs more development. [N.B. Peter Bergmann was expecting ≥ 10^{15} for CJS w.R "log" n. → 512.00

2/08/02

ID (511) ← REV : .07ff + 511.26–28

On OZ srch (opfma() .22

Note 140.00 – 141.40 on Ians 1989 thru 2001

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 : 511.90 : Actually, it may well be that a problem w. a 6 byte soln. could be a fairly big problem! I really have no idea how big actual problems are! I could just do a raw estimate of some simple problem like a simple symbolic integral, & get Lcost of raw Lsrch Soln.

Say $\int(x^2 + \sin x)$ Of course it depends on what previous info TM has!

Perhaps look at Slagle's formulation of integration.

Drop .00 ff for the present; it's nice for expo, but not critical now.

511.26–28 is T-present summary: Are there other impt ideas?

.07 REV Some of the impt. ideas ① In Optimization of / digital devices & continuous params: How to do it: This assumes that a priori of contin params is known.

.08 → ② T. "factor of 2" argt. for Lsrch v.s. heurstcs: "Call it m.P.D." ← meaning of this

Giving an understanding of relation of Heurs to Lsrch. see ④

.10 ③ "Look Ahead" How good TM will not have it, but how to get it later.

④ T. "optimality" of Lsch [meaning of optimality of Lsrch for OZ problems. — see .22 ; t. CJS = $\frac{cc}{pc}$ as upper Bnd on soln. time for a particular known soln. ← see 2

⑤ The P_1 & P_2 soln. to t. OZ distribn. problem; & the Monte Carlo method of realizing it — see 508.03 for two most. ← How to do let P.D. to OZ srch; the 2 related PDs. The Monte Carlo Method to realize this. Note ⑤

.14 ⑥ Mxd (MCT see 509.10) Corpus Run in general; & convergence Run for BAGS & for a QA set. (See 506.31 on MCT) → Also 507.10 is Good.

⑦ T. idea of RPD (Resource Bnd Prob Distribn): I've written some on this, I don't remember much, but.

.16 • ⑧ That t. P.D. for solving 414.18 (Cases in Lsrch OZ problem) was significantly different (larger) from t. P.D. within 414.18: since it had a much larger Corpus — i.e. all i.e. [Qi, Ai] i=1/n. This is very impt. in solving t. "scaling" problem of 576.04 [famous O^s's, Generic pts, traces of all trials.

⑨ T. AND/OR Net: "stats soln" technique.

⑩ Use of GA to solve 414.18. Some recent Ref(s) on this are v.g.! w. Run last month (or 2).

⑪ Parallel Lsrch is / at least 2 times faster than T ← 2T method T ← 3T is fastest;

" " " may also be faster by additional factor of 2^{d-1}; d being depth (in bits) of srch. Much More recent proof for this than first discovery of it.

20 — for this than first discovery of it.

.21 — ⑫ Ways to represent structs; t. input desc (3 i'v): How to do it using a Lisp { Actually, any method of dealing w/ structs must be accessible to t. System! → 515.00

2/07/02

.22 SN On [redacted] Levin's srch for OZ problems: Very Good Summary of ideas!

Say OT_i has pcost = p_i. Given problem OZ problem w. CB = c_0.

Apply all OT's to get p_i to problem: do this for all OT's: ≤ time CT.

" " " " " " " 2 " " " " 2T

: " " " : 2^k " " " " $< 2^{k}T$.

We can also do Run in ||: Now: this is t. nice way to Understand Lsrch:

There are no simple heuristics! For || Lsrch, (other than .31) after search time T, we will have spent T·p_i on OT_i.

30 If total time spent is $\frac{C_0}{p_i}$ then we spend time C_0 on OT_i.

.31 → For any OT_i it is [no worse than $\frac{1}{p_i}$ times as slow as that OT_i] .37

I Think .22 this is t. way to do Levin OZ srch: One simply divides t. CB up betw. t. OT's in accord w. their pc's's.

Actually "Any time" problems are commoner than OZ problems: So one just should spend pc·T on OT_i at t. end of time T.

(perhaps more; just time share in || to use time ∝ pc & some codes will not derive OT's.)

.35 → Also, remember that t. pc_i are conditional on t. problem derv. — They vary (improve, matures) as t. system "learns".

.35 is a critical Modifn. of t. system.

.37 (.31) With good "Education" $\frac{1}{p_i}$ will be small for very good OT's. This is t. closest thing we have to a "promise" (= "claim") (Not Quite: I think we have other "claims"!) → i.e. .31

513.00

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 : 512.40 : Int. list 511.26–.38, 512.07–.21;
Critically impt. is t. idea of all P.D.'s used in Tring be conditional : That for 2 NU probs & OZ probs, Lsrch is impotent w.out Learning Their Conditional P.D.'s.

Anyway, augment this list: Then Look at various apparent Bottlenecks in QATM & see if any thing(s) in t. list are helpful.

Studying t. list will refresh my memory so I'm more likely to realize I have solved many impt. probs. that occur.

→ Perhaps first have clearly in mind, what T. major Bottlenecks seem to be, while reviewing old results.

Note: 512.35–.37 : T. idea that OZ Lsrch can hang. if t. guiding p.d. is conditional on t. prob to be solvd.

→ Idea of listing a bunch of Good (cond'l) OT's & trying to make a language (Grammar) for this corpus. Then, try to make a similar thing w. Conditional OTS. But first do uncond'l. OTS, (tho it seems

.10 perhaps that good OT's automatically are "Conditional" (©)). I think t. Q is whether t. "conditional" restriction occurs "befor" or "after" t. O.T. is selected. "Befor" would seem more El., & is perhaps more easily realizable; T. generality of non-el. objects is obtained in t. O.T. itself.

I have a list of impt ideas betw. 1989 & ~2001 — also (perhaps) useful reviews

.16 Since Then: **An Impt Idea** was that t. Final Soln. to t. sequential induction problem was not to get t. "best set of codes" for t. corpus written on CB. — That normally, one didn't want to use lots of cc to code stuff in t. corpus not so directly related to t. immediately

.19 needed prediction. → 515.00 | 10/11/02: I think I finally solved this using t. "Encyclopedia" idea wherein Encyc. & TMs it costs to specify what part of Encyc. is being used to help solve a problem.

.20 2/9/02 | SN : "2 kinds" paper (1999) : P 256 § 3.2 : Summarizing Machine for BAGS: Here we end up w. a set of machines, each having own t. & a D.f. on D_{n+1}. We could modify this appr. for QA (cond'l prob. Dfs).

So: Just what is t. critical "Bottleneck" in QATM at present? → see 508.30

We want to maxz 414.18] We have available: "Acceptable" O^n's for O^{n+1} for $[Q_i A_i]$ $i=1|n$. ($P[A_i|Q_i] \equiv O^n(A_i|Q_i)$.) For each O^n we have a set of functs — They are "S-functs". How do we best get a set of O^{n+1}'s ?

.30 Assume for t. Moment, that we have a "Recognition Algm" so that (Q_{n+1}, A_{n+1}) has been assigned to a "special" sub function (set) of O^{n+1} (= "area of science"). So, (on of) this sub function mite be [illegible] $H(A|Q)$ $H(\cdot|\cdot)$.

.34 One attempt: View this as a General (conditional) OZ problem (See 512.22–.40 for good discu) This "soln." should be able to use all of t. "contextual" "extra info" in 512.16.

A Big Q is how do we update t. (cond'l) d.f. on OT's? Well, we use t. "MCT" approach to improving/updating d.f.'s for OZ problems. (512.14) — This is a Recursive Method (509.10)

514.00

2/10/02
ID [redacted]

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

125 B 1119

.00: 513:40: 513.34-.40 Seems to solve it: Does it? What are weak pts.?

.01 One, is breaking Corpus into "Parts": I Think this is always done in Human Prob. solving.
[This is One approach to t. difty of 515.21]

Superficially 513.34 ff looks reasonable! We first want to Map 414.18 : This is a sort of OZ (±Anyhow?) problem: Say we use ll Learn on t. OZ problem. Presumably, we start w a finite set of "Not Bad" OT's, & somehow work improve them as t. TSQ continues. We also add new OT's to t. original set, & "modify" O.T.'s. [There is some trade-off betw. using t. best OT's most of t. time, v.s. spending time on t. less good OT's to test them & get info useful in improving them & in knowing what kinds of

.09 problems they are best suited to.] (.27)

.10 SN ≈ Carl Smith says that learning functions that map integers to integers is an "adequate" induction problem. Tho he thinks of induction in t. "GOLD" sense, perhaps this is a good idea anyway! ([redacted] To get his paper, go to Kolmog-Sol page, got to his web p. — [illegible] t. relevant paper is on t. role of "forgetting" in Machine lrng).

One could certainly make a tree of functions (some of them recursive) & this tree would be t. Conc. net. Learning it would seem to be very simple (if t. trainer knows t. net).

T. foregg. would be true, if one didn't use heurs to cut down on search: Say t. data was always a Set of integer (x, y) pairs. w.o. heurs, t. no. of possl. functions to test, would ↑ very rapidly

.20 Actually, what is the problem seq. being proposed? kind of unclear! Is it a "QATSQ" in which Q's are (x,y) pt sets & t. A's are descns of functs? If so, then there would seem to be few heurs!
[[redacted] While each funct to be lrnd is "simple" small depth of srch, t. branching factor ↑ rapidly with no of possl. functs. to try.]

It would seem that Smith et al. are thinking of Gold's original problem & "identification in t. limit"!

This "Bad scaling" problem is one that I've worked on much. I think t. soln has to be in t. "extended Context" idea

.26 Perhaps try to find that writing & try to get into t. spirit of it! → (516.04)

.27: (.09) There was also t. Great idea of taking a bunch of OT's (I listed maybe 130 of them) & devising a grammar to favor t. set. I'd like a lrng → every thing it derived was automatically an O.T..

[There seem to be 2 aspects of an O.T. (or of O.T.'s) : (1) analytical : To look at problem and get

.30 ideas on oper by pure/mathmatical analysis. (2) Empirical: Trying various conds & devising new conds from

.31 these results: ——— (probly best combd w. analytical Approach).

I had t. idea that at first most OT's would be unchanging & little if any lrng in them — either uncondl. or condl. — but only after TM was fairly smart would we be going to have it work on improving t. O.T.'s

More recently, I wanted TM to do some SI very early, so it could pick up some rather simple, but important Heur. ideas, (Like lrng during Level 2 L.A. (lookahead))

(516.00)

2/10/02

ID

REV

515

(13 = [illegible] lined.)

(573.19 .16)

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

Spec

00: 512.21: (13) That simply ~~optimizing~~ Maximizing total pc of codes of a Sequential Corpus was not the "final soln" for the ind. problem. That each problem normally has a new assoc. sub-corpus. [Write in available CB]

One approach was to divide corpus into subcorpi. I don't know if I had a very general way to do this. (or an optimal way!). Also the Divisions were always merging, and breaking up ~~[illegible]~~ so that the Divisions ~~[illegible]~~ of corpus into subcorpi was very dynamic — constantly changing.

(14) Partitioning the corpus. Makes testing easier; usually makes soln search easier because the ~~[illegible]~~ concs. needed for soln. are common (to some extent) w/ Diff. partitions. (574.01)

(15) Making a S. Grammar for a bunch of the p. O.T.S. (574.29)

.08 (16) That almost all pc's of interest must be conditional pc's. Otherwise Lcost of soln. is too high.

.09 The TSQ is arranged so that Lcost of soln of the next problem is acceptable. (509.[illegible] is good discn)

.10 (17) Recursive extrapoln (used much by humans) (578.23) — How we can easily represent it in AZ141.

(18) Use of "Indexing" in TSQ Design.

(19) All heurs for ENV or END[illegible] can be expressed as OT's: (after we express ENV & END probs as OZ-probs.) This is more general than (2) (512.08) — which assumes all Heurs are expressible as Modifns of the / ~~[illegible]~~ P. D. Best Guide Lsrch.

(20) In OZ probs, any OT can be arbtly "Good" — so good that it effectively replaces the GOT (Lsrch) (since it got almost all of the wt. almost every time). So in using Lsrch for OZ probs (the most general & all encompassing problem type (except perhaps for LA!), we are not committing ourselves to an ultimately inferior system. [Look at last]

.20

.30

2/11/02 516
ID

HAMPTON RESEARCH ASSOCIATES HP 852X
26 Boylston St., Cambridge, Mass. 02138

.00: 514.40; [SN] Get one of the palmtops to keep index of TM topics by page nos. Th. info should be downloadable into desktop machine so I can rearrange it in "Outline" form — like "MaxThink". Try Best Pdf Best Prone people at 12 Mass Ave lout me. (Compaq?). — maybe look it up on Google: also try looking up t. HP Palm top on Google.

.04: 514.26: "Scaling" (Apparent ↓ in pc's of solns as TM matures) would seem to be a, (if not the) major problem in TM. Presumably, various those can be found to deal w. it.
→ Try to find some of my old discussions of "Scaling" — how it mite be solved by Genrl "Context".
One idea on How this mite work: Normally, in t. construction of a cond, there will be an enormous no. of functions (or operators) that can be used at any construction pt. (= large "Branching factor"). [The "Branching factor" as TM matures.] This is to be reduced by 1) Historical studies of "local context" modifying pc's of various choices 2) Larger context — historical studies: Like "Is this a Biology or an Algebra problem". The last mite be automatically invoked
.10 when we break corpus into "Parts".
Another (probably Big) reduction of t. Branching is obtained by "Logical Reasoning" about t. problem: At first we mite use standard early AI. such ideas like subgoals chaining, backward chaining, GPS (Vector Goal for Inv probs), etc.

Perhaps soln. time should → ∞ as corpus size → ∞! Soln. time mite be $(N^{-1} 2^{\log^* N})$, but — N = corpus size.
so it → ∞ but very slowly. Actually, it prob'ly depends on how Good a soln. one wants! —
.19 Also, how diff't t. corpus is. What would be conds under which $(N^{-1} 2^{\log^* N})$ would be correct? (.32)

.20 [SN] We feel that t. "discontinuity" implied by "GRUE", "Bleen" etc, is unlikely, yet t. discontinuity of water Boiling, caterpillar → Butterfly are o.k. Just how does this work? Can we apply how we do this to more general contexts? Say we don't know modern Science", but we have observed water boil, & we have some concept of "temperature".
We have observed caterpillar → butterfly, but know little about Biology.
I guess t. context is IT! We've observed cat → But; H2O → steam, xforms, but not discontinuous changes of color. (The colors could change in t. sun). (Context is a kind of "Local Gestalt"). (Rule: "Things change slowly wrt smooth changes of continuous
.30 params, unless exceptions have been observed". (T. exceptions are generalized a bit — but only "Locally"). ~~[struck-through text]~~ [Law (Rule of Continuity)]

.32 (.19) If we "context", pc's of concs → $\frac{1}{N}$ (N = no. of QA's); then for depth "d" problems pc → $\frac{1}{N^d}$ for N as small as 100 & D = 3; Dep'n factor of M.; for N = 1000 & d = 4, factor is 10^{12} = MM — very large! Anyway, it's not tolerable! (But OK to study about TSQ's). Perhaps if we "partition t. corpus", T. "N^d" wouldn't grow so fast. — But, in general, I think we have to study just what it is that humans do to ↑ pc of solns: i.e. heuristics. Actually [N] is prob'ly not t. no. of QA's, but t. no. of New concs/deductions in TM's store. — in which case t. $\frac{1}{N^d}$ becomes more reasonable/accurate.

ID

≈ GA.08 (But much more general applic than just GA)

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

00:516.40: So: T. main problem is ~~xx~~ of pt. 419.18: First helps TSQ's conc. net.. At first, t. TSQ is used to reduce cost of solns via Conc. net. Later it is also used to im. Rings That are useful in prob solving — ~~x x~~ like Math, Logic, proby, statistics.

Some very impt. methods to ↓ cc of soln: ① Partitioning corpus ("problem types") ② pc's of concs become conditional on nature of problem (or of "sub problem"), ~~xx~~ (③ of 512). Rather could pc's have as corpus, t. entire ~~xx~~ set of TM's trees (≡ history). At t. beginning, TM will be able to use only a small part of that corpus. → 518.00 | ≈ Operators; functions.

.07

.08 [SN] 2/13/02: In Organic Evoln There are at least 2 kinds of Genes: ① Proteins used to construct Being ② Control genes that orchestrate t. creation ~~xx~~ ~~xx~~ & /order of time creation of (function / operator / op code) genes.

② can be regarded as "structure", using ① as building elements. ~~xx~~

In GA, (& Organic Evoln), one would mainly want to "Mix ~~xx~~ (crossover) typ 1 w. type 1 & type 2 w. type 2. T. concept of "Alleles" makes this even narrower: so for each type of Gene, (≡ Allel) There are a small fixed no. of possys for crossover.

It is interesting That Biology uses so many "op codes". Other Than RISC machines, Computers also have large inst sets, but They are usually simple combinations of a small primitive set.

Of t. 30k human genes, > ~~xxxx~~ is k (say > 20k) are op codes. Th. control genes construct "Macros" of t. op codes. I'm not sure how t. "~~xxx~~control works: It does tell ~~xx~~ when (& in what order) and how many "op codes" are generated. T. "op codes" Themselves probably have constraints (instructions) on how They can be combined w. other "op codes".

.21 [SN] Some time ago, I was Thinking about GA. I wanted to devise a s. lang. to extrapolate

.22 t. present population of cands & Their assoc. "G's". I ~~xxx~~ was not successful in devising useful models. I will have a similar problem now w. TSQ's — in which I have a corpus of examples & I have to make a model of it — often t. corpus will include "G" data [e.g. when I am trying to improve th. D.F. on OT's (w.r.t. a particular problem type — but even in t. unconditional case ( )].

What I plan to do is see how humans seem to do t. extrapoln. & Then make a ~~xxxx~~ formal model of it.

A possy for .21–.22: Certain macros have hy G's assoc. w. Them. Also ~~xx~~ each has a pc assoc. w. it. To get t. G of a cand, add G's of ~~xx~~ its macros; to get its P, multiply pc's of macros. This Gives us a single G & single pc for a cand. We want cands w. hy G. pc (≡ expected G). ("Needs Work" — There may be t. germ of an idea here!).

2/14/02 ED   Expo: .00 .29

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00: (SPOC 517.07): **Expo**: Initially, we have this corpus (young, reconstr TM) $(Q_0, A_0), \ldots (Q_n, A_n)$ for $n = 0$ ...
The P.D. for Or & OT's is an uncondl P.D., but t. OT's are as good as we can get them.
After n is > 0, The O.T. becomes conditional on some known O^{n-1}'s. — So we want to incremant or modify O^{n-1}, if possl. We devise O.T.'s that are specifically designd to do this.

.07 As n grows, 4 things occur: ① we have many $(O^n, O^{n+1}, Q_{n+1}, A_{n+1})$ quads
.08 for t. system to use in induction. ② we have the traces of TM's behavior up to n
③ We have, as part of t. corpus, problems that TM has solved, that
.10 are usefully related to the optzn problem ④ T. trainer has devised a "grammar" for a good set of O.T.'s.
.11 Note: In .07 — .08, there is some confusion betw. the "conditions" in t. P.D.
.12 & the data upon which t. pd is to be basd. → (21)
I would do well to explain .00 ff & .11 — .12 in particular (in some detail) in t. report.

SN optn of O^n can be regarded as a n-component GPS problem!
[GPS is an OZ problem w. a vector Goal (or vector fitness function)]. T. components are not "indip", but its still a good way to look at t. problem. T. sets of QA's form m
.20 — Similarity classes: Each class can have a Goal component — & Goals are more "indip"
.21 (12) Re: .07 — .12: Also, partitioning of corpus is impt thing helping optz. 414.18 — (Helps much w GPS).
It is impt. to emphasize t. conditionality of t. P.D. Guiding t. Lsch. (575.03)

.23 **SN** 2/15/02 One impt. Heur that Humans use a lot is recursive extrapoln: i.e.
From 2 (or 1) example(s) they are able to immediately genz: E.g. ① Invertible
Substitution to simplify/solve many kinds of probs (eq. soln, symbolic integration)
.28 ② Soln. of N eqs in N unk from example of 2 eqs in 2 unks. (579.09)

.29 More Generally about QATM's operation: We give it only TSQ: After it solves a seq of probs,
.30 **EXPO** the concs (created in solving / used to solve) those probs: (either as final soln. funcs or as sub-concs) should be accessable to t. system for solving (new coming) problems. [If t. concs are not functions (is this possl?) then this should still be true] This is one of t. fundamental ideas in QATM's operation. [Also ≃ calld "CHUNKING"] → 534.00 — .05
.32
.33 Another essential idea is that t. relative pc's of use of t. concs in new cunds should depend much on t. problem (or sub-probs) being solvd. (≃ "extended context dependence") — else we get "scaling" problems.

.36 Another aspect of "scaling": if we do not "partition t. corpus" it takes long time ∝ no. of QAs.
.37 to test each cond. So time for R problems is ∝ R^2 rather than ∝ R — which is very bad. ← The way QATM works one could have an impt set of QAS to give "Quick Ans" to save much time!
It is pretty possl. to demonstrate t. effect of t. conc Net above viz .30 — .32; Then work on
.33 & .36 "superstability".

2/15/02 ID

GA (.37)

T. finds (or has and) "Curve Concave Prob" Soln. on ID 19.26-.40
∃ a later soln. (maybe simpler),

HAMPTON RESEARCH ASSOCIATES

26 Boylston St., Cambridge, Mass. 02138

00 518.40 : Another (hour) place that we can work on "separately", is TMS S.I. aspect by learning from T. TSQ. — is applying things I've learned from its own TSQ, to T. problems of Hour construction or any other kinds of S.I. — like for instance LA (LookAhead). — Also I'm during a Lunch.

02 An important idea is That we use OT's to solve interchange problems. An OT can be better than Lunch — so that is essentially "T. keeping the solution problem" — a primitive way to initially use Lunch.

03 The 518.29 — 518.32 seems (like a perhaps "adequate" application of how TM moves, the diffys in TSQ design occurred harder & got Beyond 518.29 — .32! i.e. T. conc. not a ite realization as a TSQ. T. diffy may have been that my Reference Image was either incomplete, or Not T. concs. I needed entirely correct at low pc (initially). Perhaps I should learn the more LISP, to get ideas on what Concs are really necessary!

07 So: Re: .03: Perhaps just write up a reasonable sup. of QA's for Algebra & see how it does!

09 — Tho I should have a conc. pattern in mind to calculate for CJS's.

09 518.28 SN TM could use recursion to notice many simple generalizations. — even very early in T. TSQ: Recursion might be some of the (early primitive) hours.

10 Another idea close to recursion & perhaps having simpler Hours: T. ideas of successively X thing. something until it is of "suitable form" (i.e. it satisfies certain criteria (like an IND problem).

13 The idea of Doing a X thing (or choice of many in series) to better a Conc.: (This is in direction of GPS) .18

BUT I should be able to do 518.29-.32 w.o any hours: This is T. (Pure TSQ part). see 520.29 on GPS

There is T. q. of How to realize T. "sense of conc" in 518.29-.32. — It seems "trivial" but there may be > 1 way to do it.

18 (.13) An example would be successive simplification of an Alg. expression until it becomes a single goal.

19 This would be a way of Realizing ANL. → → (520.06)

30 SN In Optimizing ¶ 14.18: If we have S-problems, We will normally have to do both discrete & continuous optim. This can be very time consuming! We pick a set of discrete params: Then finding T. continuous optimum can take lots of time. It might be well to try to get a fast, roff idea of what T. peak might be! Do this for several discrete sets of params that seem to have good results. Pick T. sets of discrete params that seem best & do more contin. optim on that set, until it's clear which are "best": We will Normally be interested in finding a large no. of "good" discrete param sets. This Seems like a Hard Problem! It is solved in GA, but

30 perhaps very slowly (inefficiently?).

A monte carlo (evolution) is not unreasonable: We just pick random points in continuous space & add up their Goncs. This will give a reasonable estimate of T. true wt of T. model. If T. peak is very narrow & very hy, we will probably miss it, hwr. — So we will not be able to do models of that type in this method.

Usually a narrow peak gets little wt, hwr, so it o.k., unless T. peak is very hy: chosen That "quadratic form" fitting method could be used to find peaks of finally set of models.

For a multidim space, w. hy dim, T. method usually gives low means, which actually correct!

37 GA: for solving ¶ 14.18 optim: If we use a good TSQ: we still have scaling-effects (perhaps) of 518.33 & 136: 518.36 taking Longer time a longer sequence of QA's to get a longer QA sequence! (518.37 R) would help much here. I don't know how 518.33 is related to GA.

(520.00)

2/18/02 ID   GA.13:(.37–.38) — V.G.   [redacted]

58   390 + 14 = 400 wpm.   ↓ 90k to 124 A.   520

[518 798 1826] Brett Blanchard.   120 .12 48w!

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00: 519.40 : On Writing TSQ's: Perhaps write TSQ so that normal human would be expected to solve it. Then look at Cost of solns. If they are much more than expected, find out why Humans are able to do better & modify system so it can do as well as Humans.
"As well as Humans" I mean w.r.t. T. concept not discovery only, not w.r.t. advanced heurs.

.06 519.19 For ANL: Teach this rather directly: eval [redacted] α (α is a string). Means a. nonassoc. that
.07 string. We can't always find that no., hvr.: but eval 2·α = 2. eval α.

.08 [SN] (HARD Partitioning of Corpus): does 2 Good things (1) It gives a kind of discrete (yes/no) context: which (2) helps ↑ pc of Cones (3) ↓ testing time of Cones.

.10 I've been thinking of ~~soft~~ Context as a "soft" partition of corpus. Can I use it to ↓ testing time of Cands? It would seem possl.! In (518.37R) we do statistical testing of a corpus. Difnt QA's can be given
.12 difnt wts w.r.t. likelyhood of relevance to the latest ~~abst~~ local change in O^{m} (≡ Cand).

.13 In GA, a (Hard Subcorpus) could correspond to a subset of the population of cands.
a Soft subcorpus " " " weighting on " " " " — a Pd.   [90k]
Hvr, it's more like a sequence of INV problems

Also in GA: The [fitness function] is a way of constructing a [TSQ] for t. problem. If t. fitness funct doesn't yield an adequate TSQ t. problem will not be solvable! Hvr, given that t. TSQ is adequate, GA can have "scaling problems" w. "large" problems. (518.33, .36; 520.08, .12; But note 520.13.

.20 So, choice of "fitness funct" is critical in solving GA pbs; It may be that we want a "Dynamic" "Fitness Funct": One that changes during t. problem soln. T. "GA administrator" must be able to recognize changes in t. problem as t. GA search continues.
T. fitness functions of any population, applied to a given problem, Give a "profile" that can tell much about that problem. Each fitness funct

| Actually I'm not sure about just what I'm talking about! — go back & fill in more details (if possl.!).
This "Profile" idea is: Maybe way to link to t. seq. of QA's? — But I was thinking of t. "Classical" GA that has a scaling diffy problem, in which t. "Fitness funct" constructs t. TSQ.

What do I mean by "In GA, t. problem changes as we approach t. soln"? One possl. meaning!
.30 ... T. "problem" is "t. population" (mainly t. part of by fitness) & its "difference" from t. soln.
.24 Say we apply a vector Gore (like GPS) (Note 519.13). "Profile" is w.r.t. t. Vector components of t. Gor. (say k dim. vector)
Our population is represented by many pts in a k dim space (each cand. is a pt. — pts. can be wtd. w. wt. ≥ 1)
If certain vectors seem to need more improvement, we use certain mutations, crossovers on those cands / (cand pairs) etc. BUT. Perhaps we should always do that! i.e., for each cand (cand pair), we look at its G vectors & decide best kinds of Mutations (crossovers) to try. Or, not to chose pairs of
.37 ((Cands at random, but Match those that seem to complement each other — to complement each other's
.38 ((differences. An [intelligent Matchmaker.] Perhaps have many offsprings of a promising couple!   (521.00)
— or many crossover pairs that can be "matched" — each making a few children. Many kids/pair seems too "Elitist" — too "Incestuous": "less diversified"

2/19/[redacted]02
ID

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00: 520.40: **SN** On Cond. pc: Consider $P_u(A|Q)$. P_u is a universal cond P.D. If,

for all $P(A|Q)$: $P_u(A|Q) \geq k\, P(A,Q)$. k depends on $P(\cdot|\cdot)$ and $P_u(\cdot|\cdot)$ but is indep of Q, A.

At first Glance, this is rather disturbing! There are many "Universal" P_u's; say we chose one a priori: Then there is No assurance that a long relevant TSQ will give a good $P_u(A_{n+1}|Q_{n+1})$.
Well, surely, that's O.K.!

Or, it screws up the proof of the invariance thm for QA corpi; On second thought, it's O.K.;
The way QATM works (i.e. 414.18), we use the universal machine to simulate various cond pc's.
It takes pc $= k$ to simulate the true cond.pc. So 414.18 is, ideally, low by a factor of k (though), k being P_u's pc of ($\sum_{i=1}^{\infty} 2^{-\ell_i}$) of simulating the "true" cond. pd.

No, No, No! OMMIT

On time needed to test Conds (GA or Lsrch): In Math TSQ's, The 100% correctness is needed, usually only a few QA's need to be tested to detect an error in O^n — i.e. in certain complicated math expressions, practically any error in O^n will give an error in evaln.
So I think "testing Conds" may not always take $\propto$ (no. of QA's thus far) —
It will ↑ for more slowly, maybe [redacted] → a Constant.
It testing Conds for $\Sigma \cdot$ QA's, Again, usually Most QA's will have their pc's invariant, but some will ↑ or ↓ markedly. → We may be able to usually test relatively few of the corpus to evaluate a Cond.

.16 Re: TSQ writing: For ANL 1 / linear eq solving; write (In English), a list of concepts that would be adequate for TM to work the TSQ: e.g.:
1) Idea of Quantity (≡ evaln funct?) 2) That "its" Quantity can be substituted for any expression within any expression. 3) That (expns) can be treated as a quantity (≡ number)
4) That numbers are always directly "Quantitys".
5) That we can treat "Literals" as quantitys: That rules applying to Quantitys are true even if we can't yet evaluate the quantitys.
6) [redacted] What a "proof" is: [redacted] If $3+x=7$, how to "prove" $x=7-3=4$."
7) That this idea of "proof" can be used to solve eqs.!

.26 .16 ff is One way to write TSQ's; Another way;
1) Give sequence of examples that a person "should" be able to "track".
2) Express Solns in some lang (Lisp, Forth, Machine lang ...).
3) Look at CJ's's of solns:
4) Modify TSQ to give reasonable cjs's.
5) The seq. of CJS's can get rather large (say 10^{18} 2^{60}) — But note that This is expected effect of "scaling".
So we next have to add theorys to deal w. scaling. at least 518.33 a .36 (525.00 spec)

It would seem that the "outline" of 518.29-519.02 would be enuf to write "The report".
So perhaps fill in the "outline" a bit, Then look carefully at what I've written & see how it should be modified / augmented.
513.34-514.09 is an earlier "outline of QATM".

(526.00 spec)

[redacted]

2/19/02
ID

Hutter: see 576.16 for impt criticism of Hutter's Method.

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 : On Hutter's method of solving (some) Inv. problems:

.02 He is interested in solving problems like Given $M(x)=y$, and given y to find x;
But he wants a general pgm ~~that will~~ $F(\cdot)$ $\ni x = F(y)$, and $F(\cdot)$ shouldn't take too long to execute.
~~[illegible]~~ Marcus considers that finding f & proving it will work for all y, will take time T; & if $F(\cdot)$ was an efficient algm, it could find $F(y)$ in poly time (or whatever).
Lsrch would find $F(y)$ & run in poly time, but w. a large ~~cost~~ Mult ~~[illegible]~~ f
— given by the recost of $F(\cdot)$ — This cost would occur for all values of y.
Instead, Marcus hunts for $F(\cdot)$ & the proof that it is the inverse of $M(\cdot)$ (.02). ← see 576.16 for impt criticism of this view
~~If~~ If finding the proof, takes time T, then his ~~[illegible]~~ algm take time of $F(\cdot)$ plus the time to find and proof correctness of $F(\cdot)$. So Marcus has ~~[illegible]~~ (if he does solve the problem) ~~a~~ large
.10 ~~[illegible]~~ additive, u.s. a large Mult constant for regular Lsrch.
∑ Actually Marcus does both Lsrch & his srch "simultly" by timesharing, so if he can't find an ~~[illegible]~~ f & proof, he will get a soln via regular Lsrch.

.14 It ~~may be~~ is possl. to do ~~[illegible]~~ TSQ's for both searches in Marcus' method.
In general, either component of the search is impractically time consuming — unless one uses condl pc to guide the srches, — So one needs TSQ's for both search components. It is probly possl. to use the same TSQ for both, if Inversion of $M(\cdot)$ is anything like theorem proving (which is an Fun problem). I Guess searching for both $F(\cdot)$ & the proof, together, constitutes an inv. problem.

.20 Don't Hsrch — In a regular TSQ (e.g. for QATM), one could use ~~[illegible]~~ Hutter's srch method (Hsrch) instead of Lsrch. It would be a waste of time unless the TM had adequate heurs for finding $F(\cdot)$ & proving it was correct. → 523.21

.22

[SN] On use of Lsrch for INVerting functions! Often, once TM has found way to xform $x = \overset{Binary}{1001}$ into $M^*(x)$, & & it is given $x = \overset{longer\ Binary}{1101100}$, it will simply try the new value of x in "The same function" if this is meaningful. This will be ~~[illegible]~~ equivalent to ~~[illegible]~~ Hsrch, but wo. the need for proof. Hvr., H's method is useful when

.30 Verifn. of $M(x)=y$ can be very time consuming. — The "proof" makes verifn. unnecy.

Note that Hutter meant Hsrch to be used for some problems in which Lsrch was inappropriate:
e.g. ~~[illegible]~~ Multiplying large Matrices (presumably by decomposition). (Say it found the "nondecomp" method

.33 first ??).

~~[illegible]~~
Could H's "LA" (Look ahead, Dynamic Pgmg) system be worked into the TSQ in a similar way?

Re: .00 – .33: We can think of the "kind of" (good)? Hsrch as consisting of 2 ll (time shared) ~~[illegible]~~ (1) regular Lsrch for soln, (2) regular Lsrch for thm & proof | 2 INV probs in ll. If (2) is solved first, then it is used as soln. — If not, (1) is used.

spec 523.00

2/21/02
ID

Hutter (cont.)

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

(Spec 522.40): On the other hand [illegible] (2) were solve first, then regular Lsrch. would find it first & use it as a soln!

A poss! reason why not: It may take Lsrch too long to verify solns for Lsrch.

What we do, is solve $M_0^{-1}(y)$ for a large and "random" value of y $\ni$ $M_0^{-1}(\cdot)$ generalizes. Then for subsequent values of y, we use the same function. So: I it took time T_0 to find M_0^{-1} then using this "general" soln will take Time T_0 to find M_0^{-1}.

Trouble is, M_0^{-1} need not be particularly optimum or fast for |y| values > $|y_0|$. For many |y|, hvr it would do better than $M_{optimum}^{-1}(\cdot)$ mult by $2^{+\ell(M_{optimum})}$.

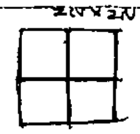
This brings up a possl. Criticism of Hsrch! It requires that TM find a $M^{-1}(\cdot)$ and prove not only that it is $M^{-1}(\cdot)$ but prove that it is the best possl.! Maybe not! — Say it only proves $M_0^{-1}(y)$ works for all y. (Not optimality of M_0^{-1}) If it got a soln before regular Lsrch, it would still be faster than Lsrch.

Hvr, If Hsrch fails to find a prove correctness of $M_0^{-1}(\cdot)$, then the time spent in that aspect of the srch would have been wasted — i.e. straight Lsrch would have been far faster. It may be that Hutter however spends more time on the Non-Lsrch than the Lsrch part, so he never gets worse than double time of Lsrch.

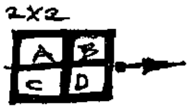
So Hsrch is sometimes faster than Lsrch (for long |y|), but never worse than twice as slow (or some other factor, or function — up to the user to decide — it could be an arbitrarily small fraction of the total cc, — but if it is a small fraction, the likelihood of finding M_0^{-1} & proof before regular Lsrch succeeds, becomes smaller.)

12(: 522.32: From a practical pt. of view (in QATM), (522.14–.22) TM could decide how much (what fraction) of its time to use for searching for a M_0^{-1} & proof. Just as a human would, it would spend more time on it if it felt it had "good ideas" in this area.
In fact, H's method is $\approx$ what a Human Mathian would do.

SN on Matrix inversion 2Nx2N matrix.



2x2



What is inverse of a 2x2 matrix?

Hutter said that his method is applicable to solving specific problems like we have a 1000x1000 matrix to invert. That if [illegible] it can be proved that a certain method inverts it in time $kn^{2.7}$ & his method did the proofs, it would take his method $k\,1000^{2.7}$ + time for proof to do the inversion. For large enough n, this would be faster than Lsrch, & would be close to $kn^{2.7}$, rather than $2^{\ell}\cdot kn^{2.7}$ which normal Lsrch would do.
(ℓ is ℓ cost of Lsrch).
I'm not sure that's exactly what he's saying: He may be merely looking for proof that a certain method actually inverts the matrix (no proof for time). The time thing need not be proved. T. time for 1000x1000 matrix need only be known.

Anyway, & Normal Lsrch for OZ problems: Does it automatically include H's method? —

2/23/02 524

ID

Hutter

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

00:523.40 : I usually don't have a clear concept of what OZ Lsch is!

If we use Conditional pc to guide selection of OT's: we would eventually discover Hutter's method (if it really works) & an OT would use it whenever it was appropriate. ([illegible]

My impression is that H srch is not a variety of OZ Lsrch, but is closer to INV Lsrch. [illegible]

.05 [Actually, it would be better to apply OZ Lsrch to INV problems! The 6one would be yes/no. (Most), but the O.T.'s would recognize that & use appropriate srch

.08 methods — they would use INV Lsrch if appropriate, or Hsrch if appropriate. → (528.28)

So The big Q is: "Is OZ Lsrch an Efficient way to find good OT's?" (The INV's are usually (always) "Anytime" problems)

10 Well, I think it's recursive so if it's good, it's good. Remember this is Cond'l Lsrch.

Seems strange, because OT's are assigned on basis of nature of problem — Then each O.T. 1st looks at "nature of problem" before doing Anything.

(SN) In [illegible] modifying the Pd for Guiding OZ Lsrch: One way to do this is to [modify old O.T.'s] & gen't new OT's by pc. The exact mechanics of how this can be done, must be worked out.

H srch can probably be applied to OZ problems. If we can prove that a particular O.T. will converge (or ↑G) at a certain rate (I'm not sure this is the right way to think about it).

Conditional Lsrch for INV. probs. Would this not eventually discover Hsrch (if Hsrch was "appropriate.")? In QATM we only (?) solve Open-problems, so I'd have

20 to adapt Hsrch to it — it couldn't be used as it is.

[NOTE] I think the Origin of Hsrch was to find fast way to do Matrix Mult. Normally Execution time is ~ N3. Finding a faster way to do it would then require ~ N3 to verify.

So, for a particular problem, OZ Lsrch would not help because evaluation of the Gen'l would take too much time.

Can we gen'z this idea (of hy cc evaln/verifn) for OZ, INV probs? — Maybe even IND probs.

In the case of INV. probs, we have the Corpus to code: We have a particular code; hvr, it takes a very long time to verify that the code works for the corpus. If we were able to prove that it would for the corpus, we might use the "extensions" of that code for predn.

30 Of course we do know a way to get all short codes — that takes much too much time. — But superficially, this fact is of no value in predn! — It doesn't give us the code — it just tells how to find it.

A possibly "better ms [illegible] ways" Solution to Hsrch inversion type of problem than HS: It may be faster & work when we have no proof:

To invert very large matrix of edges, N: Find fast gen'l way to invert matrices using regular Lsrch, specific diagonals of large sizes << N w. random elements. Such progs are very likely to work w. much larger edges. This seems close to way humans do it (?) (see 557.10)

2/24/02

ID

HAMPTON RESEARCH ASSOCIATES, [illegible]
26 Boylston St., Cambridge, Mass. 02138

(SPEC (524.40)) This will be attempts to ~~write~~ TSQ's using ideas of 521.16 & .26. ← Actually [illegible]

[SN] T. derivation of how machine lang. was used to realize [illegible] "GP" is of interest.

2/24/02
ID

GNP ; (GDP)

525½

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

: GNP: A computer life type game or environment in which the reward for an individual (player) is ∝ ↑ in GNP of all: this encourages ~~[illegible]~~ personnel, "laundry", bringing up of children (costly but useful in long run).

—As opposed to games that are Zero ~~[illegible]~~ sum

Also, this game (simulation) could be good at ~~[illegible]~~ evolving rapidly into ~~[illegible]~~ a soln. to a great all-over problem. This "problem" soln/oore becomes the measure of GNP. — the "fitness function".

Its more like "[illegible] God" than "Artificial Life".

So, what I want is "A Life", but in a Game that encourages ~~[illegible]~~ Cooperation and Lookahead.

2/24/02 ID [Report] Abcdefg ABCDef $\int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} dx = \sqrt{2\pi}$

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

[Report] : "Outline" : This is like an introduction

.00 : SPAC 521.40 ; First describe QA problem: Give many examples. Also, how to pd on A can have various forms.

The examples can show how general the QA formalism is.

.02 [SN] Consider the Q "Find $x \ni M(x)$ is max in 10 seconds." (or some "anytime" problem):

- How can this be expressed as a [illegible] QA problem?

.04 Well, we can give examples of probs. of this sort, using exactly the same language to describe those problems.

.05 (Ideal case) TM would try to guess the mechanism that generated the answers, rather than try to get a max in 10 seconds.

It would seem that this is a very impt issue! — Since we want TM to do S.I.

On the other hand in Mx corps Term, an OZ problem becomes a kind of "induction problem" (a QA problem of the kind that QATM normally solves). So while QATM can't be directly asked to work optzn problem, we can xfm an optzn problem into a set of QA probs. that QATM can work.

10 QATM has internally a way to look at any OZ problem & quickly decide which OT('s) are appropriate. It gives "pc" wts to all OTs w.r.t. that (" ").

[NB] If we use cond pd's for OZ learn, [illegible] "Mature" TM will assign a large pc to a [illegible] O.T. w.r.t. any particular OZ problem — So it may get close to optimum & in the available time.

→ Still, QATM's language facility isn't exactly like that of a Human: We can Tell humans directly to "work on an optzn problem". QATM's language facility doesn't seem able to deal w. this concept.

On the other hand a RTM would have no trouble in this area: Humans are close to RTM's.

I might mention that RTM's are, indeed, better in this (and some other ways), but the diffy in controlling

20 a very intelligent RTM was beyond me! By "control" I mean getting it to do what I consider to be useful work. Yudkowski would disagree. He welcomes "independence" of TM's.

"If they are smarter than us, should we be telling them what to do?".

[redacted] QATM & the Q of .02: It mite take some training to get QATM to understand Q's like (.02). Perhaps give TM the traces of a machine trying to solve a .oz-type problem.

One trouble was that .04–.05 would occur: we can avoid this diffy by the ordering TM examples of other machines answering optzn. Q's. Perhaps get TM to understand what "optzn" means — perhaps by watching other machines doing it. Then, when we ask TM to try to work an optzn problem & we see if it understood, & if not, in what way did it misunderstand?

From this we mite devise more examples to correct its misimpression(s) — but we don't

30 — give TM a pc for its answer! It leaves about optzn "open loop"!

.31 → + "Fact" that QATM could directly learn optzn, suggests that there may be other things it can't try! (.40) note .335

.32

Conceivably, we mite give QATM a hy pc "reward" if it really uses all of its available OT's & heuristics in the best possible way to get a soln. (.355) Another way: TM has access to various

.35 subrtns within itself: Among these are a general OZ problem solver (w. or w/o CB bag Gram).

If QATM refers to this srtn. as a soln. to a OZ problem, then this is only & correct (p=1, say)

.39 We can, indeed, compute the Lcost of such a soln. — What sort of learn could it have?

.40 (.32) Can .355 be related to the Q of .31? SPAC 527.40

2/26/02 ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

spec. 526.40 : Superficially 526.355 seems like a soln. to 526.02–.03

Considering that 6. "A" that says "can't ~~~~~~~~~~~~~ optimizing routine" is absolutely correct, this is essentially (often) a MTM problem for QATM!

So Let this go for a while: Do discuss various kinds of QA's & mention that they can solve OZ problems.

Next discuss the ~~~~ formal soln. using a universal cond. d.f.

Then give 4(4.18 ~~~~ & the assoc. convergence thm & in particular the "~~~~" "if your pc is better" thm — which Occam's razor is one "vague" way to describe the thm, but note that it is obtained independently of Occam. Occam is based on it, not the other way around.

So $\frac{1}{2}$ ~~~~ problem is OZ. of 4(4.18.

Discuss Direct soln via Lsrch: give "1 byte = 12 yrs" rule. 1 ~~~~ $4 \cdot 10^9$ ops: 2^{32} = 4 bytes. = 48 yrs. from 1 op/sec. or: how long to work problem (Lsrch) 10^{8} bits/sec → x2 in $1\frac{1}{2}$ yrs?

Discuss TSQ's: "Incremental lrng" (refer to sol 86, 89 at website)!

2) Discuss "Scaling" problem w.r.t. TSQs & cons & concinnity
Mention this & the empirical result of Sainb paper (w. Paul). to $\frac{1}{n^2}$ effect. | Tell how to solve w "extended infoset"

3) "Scaling" due to fin. Time to best seconds. (or N = no of QA's). | Tell about Pathfinding works & about "types of problems" "Recognition functions".

Mention possl. substitution of "Hsrch" for Lsrch & that Humans solve probs this way: They do spend time looking for general method & proof that it will work.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 : On optim operators: In 526.335 — QATM has an optim op. within itself.

.01 Just how does it work? We have a function H(x) that looks at a problem x (have no will consider only "anytime" probs — no CB given.) H(x) then gives a P.D. over OT's {OTi}, (Pi) that work in ll on x, w. time-share ratio = Pi (= no. wt.)

At most times, each OTi will have a "Best soln thus far". The main idea of H(x) is obtained by the technique of MIXED-INT (MCT). It uses QATM as a subroutine to obtain the p.d. of OTi operating on x, having a certain Gain at the end of time T. (so this is not a duplicate problem!)

Suppose H(x) is given an "anytime" prob? — (no CB). well, it can learn to assign a "typical" CB to x. QATM can give a D.P. for the CB of the problem, x — i.e. can give expected value or Median or whatever.

If this "guessed CB" is much exceeded in the anytime problem, H(x) can be given a larger
.10 CB value.

So, in general, any QATM can solve OE probs w. or w.o. CB given by any Given.

The problem of improving the optim method, however, seems more complex than just improving H(x)'s ability to assign appropriate OT's to a problem, x. We must also invent new OT's; this is perhaps best done via a O.T. "Grammar". Grammar Construction is a natural QATM operation used for induction. Of course, I will probably devise a first Grammar for the set of Initial OT's that TM will be Given.

The optim method of .01 with H(x) is itself an O.T.: so shouldn't it be considered with "all of the O.T.'s"? Well, it could be, but it is (perhaps) relatively complex; it would get small wt. — But eventually, it would get a fair amount of wt. The general, it always takes longer than any convergent O.T., so it's
.20 never in a best O.T. — so eventually it should get small empirical wt. ($\to 0$ as $N \to \infty$)

since its subroutines to be used by other O.T.'s, they are used by other O.T.'s plus administrative etc (overhead)

In General, I haven't much considered just how the "Grand O.T." of .02 evolves. — how it decides new O.T.'s (using the O.T. Grammar) & how it decides how O.T.'s by improving the old ones. Initially, I will endow TM w. a fixed set of "Good" O.T.'s & I will try to improve Procedure 4 the set as we continue to TSQ. Presumably, TM will be better able to work on this problem after it has worked on optim problems that build up to the difficulty of inventing the "Grand O.T.".

It appears then that QATM has 2 kinds of problems: (1) induction problems (2) OE problems It uses Ind. problem routines to solve OE problems — in spirit of M.I.T. (Marvin Minsky).

.30 Initially, F(x) will look at a QA problem, decide it is an induction problem, & use L search to find a set of "characteristics". Later, when it is shown a way it will represent the induction optim implied by the induction problem to take "just another OE problem" & solve it in a great variety of ways

2/28/02 So it is now an induction/OE problem solver! Better, it is a INV/NOE problem solver. How have things improved? How does it solve INV probs? A INV prob. has only one or several solns: The "quicker" (lower CC) soln is "better". — So in this sense, it is a OE problem.

.38:524.05 I recently (last wk or 2) wrote about INV problems being OE problems, but in the obvious way (not the usual method of solving INV problems by using a fictitious criterion & NOE). This 529.05-.08 does seem like a better way to do INV probs. than I used!

2/27/02 There is a 673: ID, 6/30/02! does wrong; this should say 528½ 528½ (673) This is a dup. of 673 The date suggests 528 if should be 528½

IO / "say SOY (Spurious Optim Yield)

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

: when one picks the peak for a individual sample, the peak is spuriously high, but it is ≈ the largest expected individual (??)

Well, I suspected "No": here's a simple case: The population is in 2 parts. (Set 1 / One) has mean ϕ & $\sigma = 1$; (Set 2) mean .5 & $\sigma = .1$.

One would usually pick a peak from Set 1 - which is bad!

So it might be best to pick the peak of $(x_i - k\sigma_i)$, where k depends on parameters of the ensemble.

If σ's of individuals are unk. one can't do this. The one can put individuals are in subsets of known σ, it might be possible to do so.

Does there exist a function $f(x_i, \sigma_i) = \alpha_i$ → picking i w. peak α_i has expected error of zero.? (or some such criterion)

For 552 between ~ 2 or 3 & 100 SOY doesn't change much; it's, say ~.76 or .88

So picking i w max $x_i - .76$, say, would not be bad in for many samples.

I must first show that picking peak from subtracting proper SOY, gives expect expected value; $f(x, \sigma) = x$ is the default.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 : 528.40 : Induction problems (mkt. old 1986,89 TM) would be OZ probs. INV probs (in Rule TM) would be solved by Lsrch — not always the best way.

.02 So, the latest model: 3 kinds of problems: ① OZ problems, ② induction problems, ③ INV probs. They are all solved by ~~Opt~~ O.T.'s. So we have a "frontend" F(·) that looks at the problem, & puts it into form of a ~~LP~~ opt'n problem.

.05 ~~Or~~ INV problems may be recognized as "special" by F(·) — which may decide direct Lsrch is best for some, & assign an O.T. to them. Or we may just automatically express each INV problem as a OZ problem w.c. MESA Gore. The "Grand O.T." looks at it & may decide to assign it to OT's: Some of these will devise "soft Gore's" for it & solve them as regular OZ problems — they may even modify the Gore, as is the case in Lsrch continues. If a good "soft Gore" is not available, regular Lsrch for INV will be used.

So, it looks like Opten is the main thing!

In .05, F(·) is simply program the P.D. that guides Lsrch or the O.T.'s: F(·) is a Conditional P.d. — the cond. being x, the argt. of F(·).
Inputing the Grand Opten method involves inputing F(·) the cond'l pd on $\sum OT_i$ — Inputing F(·) automatically involves inputing/creating both kinds of O.T.'s.

The 3 kinds of problems in OZ are always "reduced" to OZ problems. We do, thus have a "Mixed Corps" so we have to define a good way to pool the 3 kinds of data — perhaps this is what the MCT actually did! AH! The Grand P.D., F(·), unifies all problem types: pools info from all 3 types of Problems

There is certainly no dearth of O.T.'s! There are dozens & scores of them in common use, & many more seem to be/~~are~~ ~~significantly~~ many significantly different types. (31)

.23 SN If I do Not Mt. Carlo the Grand of Lsrch for OZ problems, i.e. GPD changes during the Srch, it will not work out very well! The GPD's give the rate of work on each O.T. Hvr, a O.T. selected can get zero work on it if it is "above threshold" — this would help (perhaps a lot!)? — Pseudo yield
Hvr, trouble is that in this MESA search method one selects (and of layout "pseudo yield" & one doesn't actually know what the policy is!

.31: (22) So, most general way: OZ problems are the only search problems — all other problems are convertable to them: To start, hvr, we have 3 kinds of probs: OZ, INV, IND (including) In early TM (& sometimes later for special cases), INV & IND can be solved by direct Lsrch. (Lsrch ~~can~~ can (perhaps usually) be done as well or better by H srch).
Hvr. If an INV prob is done better by H srch than Lsrch, then doing it as a OZ problem is uniformly better than either.

.34 In OZ Lsrch, the strategy to improve the system (improve the F(·) conditional,) Space
if an OZ problem it is encountered as an induction problem — so we can have a Lsrch as boundary (and the 530.00
the recursive depth of OZ & Lsrch.

3/21/02 ID Quick Abort (25)

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 : SPACE 529.40: ~~[illegible]~~ 529.38–.40 is v.g.! It gives a complete ~~[illegible]~~ recursive defn. for the OZ problem! — & hence a complete defn. for solns of all INV & IND probs.

Also, any heuristic (which by definition, is good at search for soln of any problem (OZ, INV, IND)) can be regarded as an OT, so it can always fit into the model of all 3 prob. types being

.06 kinds of ~~[illegible]~~ OZ problems. (an OT, in the sense of enabling new probs to be solved faster.) (25)

While solving the ind. problem by direct (rather than OZ) Lsrch has been used for a long time, the applicn. of direct (INV-type) Lsrch to OZ probs had not been considered. It is not a v.g. way, but for ~~[illegible]~~ simple problems, it is o.k. & it is adequate ~~[illegible]~~ as a

10 Def BC BC (Boundary Cond.) for all OZ Lsrch.

If we want to insert a Heur, we can do this by inserting a new OT or modifying (generally hopefully) an existing OT.

Hvr Re: OZ: I still don't have a good way of getting TM to ~~under~~ learn to "understand" what "optimization" means! E.g. I don't know when one ~~OT~~ OT is better than another, in a particular applicn. I can know ~~[illegible]~~ that an "improvement" in an OT will work over a certain range of problem types.... etc......

20 Well, TM could develop a "property list" for "optimization". "Understanding" ~~[illegible]~~ has many (too) components. TM will ~~[illegible]~~ have ~~[illegible]~~ only one defn. of Optzn. but it can have many OT's & know how to improve them, etc. — so perhaps in an effective sense, there is no limit on how well TM can "Understand" "optzn." — Tho, it's sense it doesn't seem able to understand optzn the way I seem to!

.25 (.06) Can we get "Quick abort" "abort" into this formal way to look at heurs! It looks like we should — since it is oriented to best soln in given cB, rather than simply getting good P.D.'s.

So w. 529.38 ff. we have the OZ problem being the main prob. that also solves INV & IND probs.

When we start out, hvr, instead of using a large set of OT's, we use one simple Lsrch OT for Ind probs & another simple Lsrch OT for INV probs. This simplified form of OZ will continue until TM is smart enough

30 (has enuf experience) to use fully worked out problem of ~~[illegible]~~ devising new OT's, improving old ones & improving the "choice function" of FC:) for O.T.'s. (FC.) is a pc why funct for its use 3 + (optzn problems)

.32 Or, more exactly, it will initially be unable to work on general ~~[illegible]~~ OZ problems — unless they are of the sort that can be ~~[illegible]~~ solved by direct Lsrch. Given X ≡ the prob term : {M(·), cB}.

Say $[X_i]_{i=1}^{n}$ are the set of OZ probs in the corpus. ~~[illegible]~~ $A(S_i|X_i)$ is a PC assoc. ~~[illegible]~~ we string S_i given X_i — S_i is the final soln. to X_i. We want $A(\cdot|\cdot)$ to be as good as possible. So when is one A better than another?

Well say $[S_i]$ $i=1/n$ was the best set of solns we obtained for the set of $[X_i]$ probs.

.40 ∴ The best A is the one with largest $\prod_{i=1}^{n} A(S_i|X_i)$. (see 531.02 for correction!)

3.3.02 ±D [CHESS.14] Types of OZ probs .06–.13; .12 → B in particular; Error Correction for binary Prods (.32) 531

.00: 530.40: The last (530.00) says nothing about how long it takes for A to compute $A(S_i|x_i)$ — Given [redacted]

.01 enough time A.... Or, simpler: A could simply be a table giving the largest S_i for each x_i — Natural

.02 O.K so $PC_A \cdot \prod_{i=1}^{n} A(S_i|x_i)$ ← should be max; PC_A is pc of A.

Back to .00–.01 & the cc of A's outputs: A could simply test S_i cands in [redacted] lexical order for as long as it liked, then output the best. The original criterion seems to be unrelated to CB_2. It would give PC_A to that "best"! but the S_i's in the (.02) would often get zero pc! so .02 = O!

.06 [SN] There is a great variety of OZ problems: Noise free, time invariant; Noisy [redacted], time varying.

M(·) can be open, closed or only partially open. M can be constant in time, varying noise or

M can be time varying w. constant noise or ... Some of these are not solvable by Lsrch.

.10 Also there is the Gen: Ξ $M(s) \cdot F(t)$. F is a funct of $\overline{CB}$ used: Unsolved by Lsrch.

.12 I think if M is "open or closed", but time invariant & the noise is time invariant, that Lsrch is usable.

If M or noise are time varying, I don't know Lsrch to be usable.

.13

.14 [CHESS]: For TM to learn to play good chess. Say the problem is to get a good fast Board Evaln funct. To be able to tell if Bd_1 is better or worse than Bd_2. Since the ordering is linear (complete) TM can associate a real no. w. each Bd.

The data used: Black & white positions in master play: in which we know which side won.

I'm not sure how to compare boards! Say Black won from "Bd_1" by many moves to "Bd_2"

.20 So "Bd_1 is better for Black than Bd_2 is for white". As the game continues, we get more pairs of Bd's to compare.

.22 Also if $Bd_1 \to Bd_2$ by master play, we can assume all Bd_2' bds obtained by other moves are

.23 worse than Bd_2.

Using these .22–.23, we might be able to simulate the play of a particular master! — By using all of his bd. transitions in all of his games, as data.

To beat a particular master, use all of his games as data to simulate him — but since TM can play several ply into future, TM is considerably better than the master — even the TM simulates the master it is "master + k ply".

How to do this induction to get ordering of Bd's is unclear! (Well, the Bd evaln funct tells how to

.30 describe transitions (Moves) "economically". If we have 10k pairs w. knowledge of which is better, this is 10k bits. We want a (min length) pgm that gives the choices, w. penalty of 1 bit for each error (?). — (Not exactly)

.32 That stuff I wrote G. Wolff is relevant — at least 2 ways to impose penalties for errors. How much info needed to correct k errors out of n bits? Consider no. of configurations possl. $\binom{n}{k}$ perhaps — so $\log_2\binom{n}{k} = \log_2 \frac{n!}{k!(n-k)!}$. Here, I had more complicated expns!

— What were they? They may have involved pc's of coding n & k. Given n, rest of k is ... (and rest of n [illegible])

Do we know n? I think so. So total bcost = $\log_2\left[\frac{n \cdot n!}{k!(n-k)!}\right]$

Code k using radix n, code the error no. using radix $\frac{n!}{k!(n-k)!}$. $\ln \uparrow = \ln \frac{n \cdot n^n}{k^k (n-k)^{n-k}}$

$= \ln \left(\frac{n}{k}\right)^k \left(\frac{n}{n-k}\right)^{n-k} \cdot \sqrt{\frac{n}{k(1-k)2\pi}}\, n = \left(\frac{n}{k}\right)^k \left(\frac{n}{n-k}\right)^{n-k} \sqrt{\frac{n}{2\pi}} \cdot \sqrt{\frac{n}{k}\frac{n}{1-k}}$

.40 $\ln \left(\frac{n}{k}\right)^{k+\frac{1}{2}} \left(\frac{n}{n-k}\right)^{n-k+\frac{1}{2}} \cdot \sqrt{\frac{n}{2\pi}}$ if $p \equiv \frac{k}{n}$ this is $\frac{1}{n}\left(p^p (1-p)^{1-p}\right)^n \cdot (p(1-p))^{\frac{1}{2}} \cdot \sqrt{\frac{2\pi}{n}}$ (See 532.00)

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 531.401 In Prose 10K (key) choice pairs! Certain pairs are of much more interest than others (≈ "Near Misses"). E.g. comparing 2 bds of much different value, is not of so much interest.

Anyway We have a (eval) function that orders Boards. The pc of the function is (1) the pc of kern of the function mult by the pc of the errors (≈ 531.40) ~~needed to evaluate~~ made in the coding of the corpus. So we want max product.

Here, we do not consider cc after (Bd.) each eval. funct. — which is quite unfortunate. If we used Lsrch to find the "best" eval. funct., cc would be consid'd. — but is Lsrch a practical way to Srch in this case? Also, does Lsrch give best way to decide w. cc in the case of Chess (12)

.08 Eval. functs of this sort could be used to decide whether to bet long or short in GM,

.10 but I think this doesn't records much info. The choice is which stock to bet on, & how much — not just whether to bet long or short on a given stock.

.12 (10B) Note that cc is restricted (Q) in real chess. Tournament —

→ On the other hand, Chess (& Games in general) is good example of use of "Look Ahead" (LA). The in many (maybe most) problems, the "opponent" is not trying to be "w/m". (e.g. the "Cure Cancer" problem). Also: Giving TM a bit of Bd, More pairs than Masters (≈ Q,A) could be a nice way to get TM to "Emulate". But devising a good "eval function" seems Less "El." We gotta learn to work on both problems simultaneously & share concepts for 2 problems.

.20 : **SCALING** (scaling) in QATM: OSL (no) seems to drastically decrease importance of corpus size!

— Since in a large corpus, the chances of a 552 = 2 events "by chance" is rather large. On the other hand, in a large corpus, one must consider (n-tuples of concepts) of high n.

This is a most serious form of scaling!

.25 I don't see any good way to get around this. In general, as SSZ ↑, there will have to be more cand. Models w. adequate SSZ. We will have to use heuristics to decide which of these models to test. This amounts to using "extended Context" to decide which combinations concepts to consider — Previously, context served to give pc's to concepts & not a particular problem & not. — Now, we will also

.29 use context to decide on which combinations of concepts to Test.

.30 The problem occurs (much) in SM prodn.

Perhaps after one has a reasonably large corpus, the problem doesn't get any worse — it's just that one has a finite cc to be used to search hypoth space, & the hypoth space ↑ in size w. corpus size, but for a given C B, the amount of the hypoth space one can search is limited & ≈ indep of corpus size.

Recursive functs: definition: I had this datum in which $f^n(x_0)$ gave scores $x_0, x_1, x_2 \ldots$ from $g^n(y_0)$ gave a series $y_0, y_1 \ldots y_n$: Then the rec. funct was $x_s \to y_s$.

I did this by defining Pair Test function in one definition.

It would be easier to define $a(s) = f^s(x_0)$, $b(s) = g^s(y_0)$!

Then this funct of $x \to y$ becomes: $b(a^{-1}(x))$ $(a^{-1}(x) = s \ni f^s(x_0) = x)$

Now, $x!$ is more complex. $a(s) = f^s(x_0, x)$ $= f^1(x_0, x) = f(x_0, x)$

$f^{s+1}(x_0, x) = f(f^s(x_0, y), x)$

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00; 532.40 : 530.32 Considers a young TM. ; Other than Inv & Ind probs; what kinds of OZ problems can it work on usefully? It has a certain set of OT's in its repertoire.

Well, say it started out w. a good set of primitives to construct O.T.'s. We give it a TSQ of OT's [redacted] and it uses t. primitives to construct OT's for that TSQ.

One way: start out w. a set (large or small set) of O.T.'s that I think are reasonably good. Given a TSQ of OZ probs: At first I give TM a p.d. over t. OT's to try, using OZ Lsrch. From this, it gets enuf data so that it can assign pc's to t. set of OT's, for *any* input problem.

So TM gets fairly good at deciding which OT's are best for a given OZ problem.

→ Eventually this OT selection will be applied to t. OZ probs assoc. w. Inv & IND probs. [N.B.: in t. case of INV probs, "good OT" will devise soft G function for t. problem]

[Much earlier, I had this idea of TM "rug hours": Now "hours" correspond to OTs; How can I reconcile these 2 views? It is impt because it seems, that most all hours could be expressed as a (cond'l)

.13 modification of t. Guiding p.d. over t. srch space. In t. case of Lsrch for IND problems, t. P.D. modifications over

.14 the "context cond'l" P.D. of t. srch for cases (& numbers of cases 532.25-.29). Modification of t. Guiding Pd for a

.15 IND problem can be regarded as an effective modifn. of t. OT. (.55) While it's (very probly) not t. most general

.16 way to modify an OT, it is often very good!

– In t. Lsrch OT that is used for IND problems, we have a Guiding p/d that is conditional on t. problem descr.

Modifying this P.D. (as by a "kind hour") is a modifn of that OT. Note (.155), hvr. (i.e. its not nearly t. Best way to modify t. OT.)

So t. "classic" hour → modifn. of Guiding p.d. is a variety of OT modifn.

.21 Def **To Summarize:** T. Grand OT (≡ GOT) is t. Lsrch over "all" OT's:

in (final form) (Mature form), we will probably have a small no. (maybe only 1) v.g. OT's. We will have a F(·) that looks at t. problem (OZ, Inv, or Ind) & decides on rel wts of O.T.'s for Lsrch. T. O.T.'s will have pretty good ideas on how to work each problem given to it.

The F(·) is (in large) part of t. GOT. Some OT's (like t. one used for Lsrch on IND probs) will look at a problem & have an assoc. P.D. for Lsrch. If we have "long during Lsrch" this p.d. will be modified during t. Lsrch.

At t. beginning, when TM is young; we may have a large set of OT's in GOT (or just a small set of OT's)

To start, we will solve INV & IND probs by Lsrch. T. P.d guiding t. Lsrch will change in response to "context" (long) (generalized) (.13-.14-.16) (spec 534.00)

we want X ≡ M(X) ≡ ... we want this quickly as short

[Right margin: → Have F(·) (look at t. problem to give a "rough soln" ≡ assignment of OT's wts) — then have t. OT's work on it: is a way of "breaking down" t. problem soln. — A "rough" start, then refinement.]

.32 SN "scribble": In either Ind or Inv Lsrch: say t. problem is "X" (includes M(·), ...)

we look for a pgm that will take t. problem descr as input & give soln. as output. This is a very General pgm & t. same one should work for all problems! I think this is t. form of Lsrch soln devised in OSS 85.

In this form, if we look at t. pgms that solve various (many) problems, these pgms should have many common features. — So one should be able to develop them via a TSQ!

Of course, "one" pgm that will almost always work is "Lsrch"! — I should think, hvr, that something faster could be available — to take advantage of regularities in solns, obtained via a TSQ. → 534.13 spec

(534.00)

3/5/02 ID

"OM": t. Optimizing Machine.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

519.29

.00 (533.39) (513.20) SN I recently (Cumm[illegible] wk. or 2) had a very nice way of characterizing t. need for "chunking" & t. frisbility of t. elements of a TSQ. Also that t. "Parts", t. "new concs" created/used by past solns. should be available for combining in to new trials/cands. T. sub-chunks & chunks used in previous solns. are available for use in constructing new Cands.

When a "creative spark" occurs in t. soln. of a problem, t. results of that spark should be available in a "routine way"

.05 for deriving trials solns for subsequent problems.

.06 Having these 3 kinds of probs, makes it easier to find & express simple uses of desired/needed concs. I Now how was MCT to be applied in t. early stages of TM? Well, there is t. GOT that is shared by all, i.e. There is t. GPD (General P.D.) that is where t. "Magic" occurs. GOT is derived from GPD by integration (so GPD implies GOT). GPD is a general conditional P.D. It can be t. $O^c(\ )$ function of $O^c(A|Q)$ (QA problems) — essentially t. General induction P.D.

.10 Now: what about INV probs? — How do they relate to t. GPD? I may have written about this in my MCT work of 1998. I may not have appreciated t. complexity of INV probs at that time! — I think of Lerch only being used on them. Perhaps 533.32 ↔ is relevant! T. input (conditional) to GPD is t. .13 (533.40) (INV prob. descn) $\exists x$ plus ideas that amount to a soln "mean". T. [illegible] cond PD is then a pd over progs that map x into cand. solns.

.15 In General, Inv probs are very ~ to t. QA(IND) problem! In a tsq. of INV problems (say trying to solve eqns), we use t. same function to map t. prob. descn into t. soln. If we have a greater variety of INV probs, like eq. soln and symbolic Integration & Theorem proving, we expect that t. same function will have

.20 2 parts (like in QA probs): A (ob, sensor) recognition part & an (motor or action) part that solves t. problem. As in QA probs, this "Recognition" part, saves a lot of cc in checking that t. same function works for t. whole corpus.

.16ff makes the QA IND problem look much like a INV problem! — Also t. "same function" that we are hunting for, in t. QA (IND) case looks a little like the OT's in t. Lsrch for OZ solns! As in the INV & IND problems, we'd like to have a single O.T. but at first (at least) we will try to find f(.) a recognition function, that assigns OZ probs to OT's in a probabilistic way.

.29 .30 So .16 – .29 Make OZ, INV & IND probs look very ~, but not (yet!) exactly t. same. It may be that by modifying one or more of t. 3 problem types, I can make them all about t. "same"! Review t. behavior of t. 3 prob. types

.34 & see to what extent I can "Unify" them.

SN If I seem to "get stuck" in this TM project: Look at t. List of "ways to solve probs" ([3.1.88] ... "Gen prob solving (mainly for Humans) — 16 tricks). Also, make list of Main unsolved probs in TM, in order of (diffy / criticality).

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

00:53:40 My impression: That Main Problem in TM is "a good overall scheme of Revisions of heuristics".

537.16-.34 seems like a good approach to analyze: Look at it carefully!

As I mentioned before, INV, IND probs are best viewed as OZ problems because the solutions are less restricted (I think), than trying to solve them "other ways" — (or standard Lsrch ways)

AH, note also: In normal OZ problems, one need not have a "fast" evaln. funct. In the case of Matrix Mult, Proving that a method is fast w.o. actually doing executing the evaln funct. can be a big economy in finding a optzn (a "Max"). [illegible]

In general in Lsrch, if one can obtain a proof that a particular cand is "good" or satisfies INV criterions then this could save lots of time in evaln. An example would be finding best set of coefs for linear (or N.L.) regressn. We can "prove" mathematically that the Coef will have a certain value:

.10 This was the approach I used in the end of "The discovery of ALP". That ALP was the best lower bnd that one could prove show for $\sum 2^{-\ell(s_i)}$.

So for induction at least, one should allow stuff like proofs (Hsrch) for (cands) solns.

And in all OZ, INV, IND problems, proofs should be one of the legal ways to Evaluate Cands. [Actually, any evaln method that can be proved to be equivalent or an approxn, or a "lower bnd" — can be used. — for an Maximum problem]

So I will want to mention use of Hsrch rather than Lsrch in all 3 kinds of problems.

12/5.17 [SN] I've studied "scaling" — an effect of corpus Grows.

1) Another type of scaling is (perhaps) assoc. w. large data set for a problem.

2) " " type " " " " Long (hyper search) [Much more time spent on S.I. as part of the problem soln.]

.20 Example of ① — say the data set is an encyc., or WWW: Making a good index is one way to help deal w. this — E.g. Google.

Example of ② (I'm not sure this is example): As a child, I wanted to solve TM problem: I first educated myself in what seemed to be related fields. As I studied, my idea of what was "related" changed.

b) Read works of others on attempts to solve the problem or related problems.

c) See the list of ~16 ways to solve probs 3.1.88 for some tricks.

d) Study fields that seem to be Analogically related to the Target Problem

.26 3) If it looks like TM will have many problems of a particular type, It would be well for TM to study that area of (ing to complex) General Text, data, in that area, to get Good set of Concs.

NB In .27 ff on new kinds of "scaling": These are all more or less Ad hoc — That "A real RTM would not have such problems (?) — i.e. a RTM has a single, well defined Goal.

.29 RTM Perhaps study RTM to get ideas on how RTM deals w. Scaling & apply it to QATM

.30 [Digress] The INV probs differ from other OZ's in that in a regular OZ, the [illegible] CB is fixed & we want max G for that CB: In INV, we have a certain G that we want to reach & we want to do it in as little cc as possib. — A bit closer to a "Anytime" problem.

Say that instead of OZ Lsrch for fixed CB, we simply do all Lsrch of OT's, for whatever time (≡ cc) is available. This would solve INV problems (eventually) if they were solvable. In a practical TSQ, so that about all (if not all) INV probs presented, will be solvable in an acceptable time, using the currently "Best Genl. Algm".

One problem w. this: we have the GOT that is common to all 3 types of probs — yet the GOT is used somewhat differently for INV probs. — Just how do we go about attempting to "optimize" GOT ...? — what is the Goal? We had that GOT_1 & GOT_2 formulation.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

00: 735.40 : Can we use or modify this, so it works w. INV as well as negative OZ search? — It would seem that we would have different GOT's for different (types of classes)/(applications) of GOT.

Well, say we have a GOT_1 whatever. Given a INV prob def'n X, it gives the pc that OT_2 will solve x in time T (or fewer $< T$) — "solving" many a certain "G" level.

For a INV problem Y, $GOT_1(Y)$ gives a P.D. that OT_2 will get a given pc in time T.

From GOT_1 we can integrate in various ways to get various forms of GOT_2.

Since "various forms" need: T. pc that OT_2 will have to beat pc for Y of time T_0

T. probability that OT_2 [illegible] will have to [illegible] smallest T (allot's) for solving INV problem X

It may well be that from a suitably General GOT_1 we can get any format GOT_2 we like by suitable "integration".

There is, however, the Q of "Anytime Problems". It's not clear as to what the exact goal is (or Gen is!) [illegible] Parallel Search for OZ problems seems like a good kind of soln. — but still the Gen was not really well defined. Originally, the Gen'l OZ problem had a fixed CB & the Problem was to get it. best poss'l. Gen that CB. — However, we ended up w. each OT having a "CB or" to the pc that it would get the best soln in that CB (or "proportional to CB"). So [illegible] There is some Q as to who Pat is.

Soln. I obtained was correct! Even if it were, the practicality of its use in "Anytime probs", (in which the "CB" is quite unclear) is very weak (→ non-existent).

So, there is some Q about 'just how GOT_1 should be derived', & in what forms of GOT_2 we should [illegible] calculate & just what is the (good/best) way to update the P.D. used to guide OZ search?

This "GOT" update problem (& the MXT corpus Tom which is involved) seems very important. Can we tolerate the present "soln"? — Its applicability to "Anytime probs" seems very poor!

Perhaps we could use an "expected value" for the "Time" of "Anytime"? Even so, .13–.16 [illegible] look [illegible] "regular" OZ probs, & i.e. it's not clear that it's bad for OZ probs in which the CB is given. A is it still is with "a constant factor" of "optimum". Say we want to minimize the "Constant factor". I guess the "constant factor" is a function of the CB.

A (sort of) way to do OZ search = "Anytime version": Use $T \leftarrow 2T$ method; Alternate each "T" update w. update of assignment of pc's to OT_2's wrt the new CB, T.

[In the Continuous Timeshared, (, version of OZ Search; We use ½ of time shares to update the pc's of .29.

Re: .28–.31. I still don't see why this method is ½ "Optimum" → tho, at any time of "soln" (when the CB is given/posted/decided) it's ½ the fixed CB soln.

So how good is the fixed CB "soln."? One avg. way just for a measure TM the pc's of the "Best" OT was not far from 1. — so far ½ all problems, the "O.T." was solving them about the best poss'l. Even fairly far from Nature, being within a factor of say, 1000 of optimum, may not be bad if one has a very fast CPU.

537.00

3/11/02

ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 ! (536.40 spec) : A trouble w. 536.34 ff : T. no. of OT's that need be considered can perhaps be very large: Particularly if they have continuous params.
Perhaps OT's don't need continuous params! (even tho t. PEms being optimized could have continuous params.) → [Hvr. it still may be a very Serious Bug] ←

[redacted] This seems to be an old problem I worked on — maybe "solved": Thusly:
In INV Lsrch, t. trials can be very close — very many w. ~ pc.
However, If one learns during Lsrch. After t. first trial is unsuccessful (ECB = T / time T.) this fact & pc of "nearby" OT's, so OT's tend to be spaced further apart.
. Hvr, I don't know if it works w. // Lsch, r/o "random" Lsrch.

.10 — Also — how does it work for OZ Lsrch?

.11 Try this: for Random // OZ Lsrch: We start by choosing a OT; (ITS ∝ — randomly = w. pc = pc)
We work on that OT for a fixed, fairly long time; Then we work on S.I. to modify t. [C_j] for time to automatically.
.13 Partly in view of what happened after our [redacted] last trial. w. t. new set of Σ[C_j] we loop back (∝ .11)

.14 W. (.11—.13) say our first OT trial ran into the wall. Then we might have a lot of OT's very close to that first one. Maybe not so bad! If t. OT's actually do trials in attempts for optima Then those trials are known by t. next OT to be tried & it will be no great loss. It would continue those trials w. t. new "close by" OT.

.20 Some OT's do not involve trials, much. They involve mathematical Analysis of t. problem — so t. remarks of .14 ff are not relevant. "Not involve trials" — since an OT can use any reasonable method — put often. We expect a "trial" every once in a while, or else it would have no output!

I may want to Gloss over t. 6i PMs w. GOT (last manual) & see if there are any other Serious problems in t. present Model. It appears, hvr, that GOT is t. main "Engine" in t. system.

.26 So draw up early forms of GOT for INV, IND problems. → (538.04)

SN It may be possible to draw up a General TM in which Induction is t. primary Engine, with OT's & INV probs being (somehow) done by induction.

So, t. present main problems (1) TSQ design. (2) How to improve GOT:

.30 — For GOT! Any heuristic can be regarded as a "way to improve GOT" & should be "/ usable".

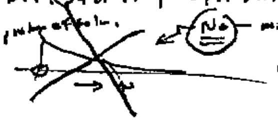
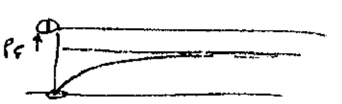
I will start TM out, with a certain GOT, that is "not too bad", Then t's good enough, so that it could learn to improve [redacted] it's GOT, w. a suitable TSQ. We can list t. (heurs / improvements) we want, & then devise a TSQ that will enable these (heurs / improvements) to be learned.

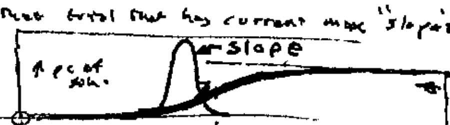
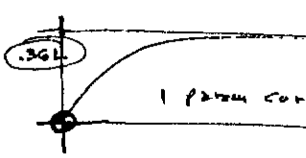
To start off, we have 3 types of probs that GOT can recognize.
1). OZ probs w. a) fixed CB b) Anytime probs.
2) IND, 3) INV.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 537.40 : : INV probs will initially be solved by direct Lsrch. As noted earlier ( ) we will be looking for a fcn. that maps the problem into the soln. Functions of these kinds tend to be similar to one another the use common concs, so a TSQ is appropriate, & (except for the scaling problem) adequate.
INV problems are initially solved by direct L srch — to try to find a fcn of max pc.

.04 ! 537.26 : Re: the usual Lsrch "soln." of the OZ problem:
.05 If we assign pc's to OT_i, $i < B = T$, Then using pc's T as a factor OT_i,
we get a certain D.f. of G_i (G for OT_i).
If the OT_i's are indep, the df. of the Max of all the OTs is = the simple SOY soln problem.
.08 " " " " notindp we get a non-simple SOY soln. (but still expressible w.o. too much diffy) W.o.
.10 [If we want the pc's to actually represent "probys" Then I imagine there are many poss. solns! (as there are ...)
.11 [I.e. spend T on OT_j & zero on all other OT_j. ($j = 1/00$).
.12 In .05-.08, say the pc's are simply wts, & we want to assign them in the best way poss!
Say we distribute T evenly among the n "best" OT_i's. The SOY soln. will tend to give a diff. MAX
That looks like  for large n G_0 will prob be $\propto \frac{1}{n}$. (plus something ... slowly w. n.) The width of the peak → ? I don't know.
If we take n = 1, hvr. maybe  A higher mean G, but v.g. proby of very low G.
.18 It would seem that small n would be better than large n. The n = 2 or 3 may be o.k.

.19 . The justfn for n >> 1 or using a pc d.f. on OT_i's, is perhaps mainly that I want
.20 .
.21 to get empirical info on the utilities of many of the OT's. [This involves L.A.! (Look Ahead)]
[On the other hand, I had that argt. for ENV probs, (That seemed to generalize to OZ problems)
.23 [but it all flows into the P.d., that Lsrch (in both cases) was close to optimum (within a factor of 2)
.24 The counter argts (against OZ Lsrch) of (.10-.11), & of (.12-.18) would seem to apply v.s.
INV Lsrch as well! — Not exactly, hvr. The D.F.s for INV are diff, The "G" is —(time to solve the problem) —
For each [redacted] trial soln. function, we have a P.d. over time it will take to solve the problem domain X.
This looks like an exact WON problem! (what to work on NeXT)
I believe that for certain assumptions about the D.f.'s for the proby of soln in time T, Lsrch is optimum.
Perhaps the curves we all may exponentials,  (No — maybe the curve of proby density of soln.) w. only 1 param. [redacted]
.30 or maybe pc  } a 2 param curve.

But it would have to be a 1 param. curve since we assign a single pc to each trial fnctn.
Was the Soy soln. in this case, "work on that trial that has current max "slope" i.e. most pc of soln per unit time.
.35 A more likely curve for real problems:  3 param curve: 1) asymptote. 2) tc of max slope 3) max slope. } 3 params. we could use a Gaussian curve. or Normal cost Squashing fnction.
.36 
Interpretation: 1) Asymptote is Proby of soln. of all time
2) Expected time of soln (If there is a soln)
3) σ^2 of (2). (or) σ^2

→ 539.00

3/12/02 LP

WON

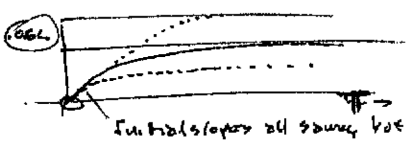
JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 : 538.40 : Even w. th 1 param curve of (538.06L), Lsrch wouldn't be right soln!

The correct soln. for corresp. sort, is to pick curve of max initial slope. Work on it solely until it has initial slope of next best curve; then do both curves simultly, spending time so as to keep slopes the same,

— then when slope is that of third best curve, work on all 3, etc.

.05 To justify normal ENV Lsrch, initial slopes of all curves are the same, but second derivatives differ;

.06 So (.06L) 

Initial slopes all same, but asymptotes differ.

$A(1-e^{-\frac{T}{\tau}})$ Initial slope is $\frac{A}{\tau}$ = constant = c

$\tau = \frac{A}{c}$ so $A(1-e^{-\frac{cT}{A}})$

(c=1) so $\boxed{A(1-e^{-\frac{T}{A}})}$ = pc of soln.

slope = $e^{-\frac{T}{A}}$. Say T_i is spent on OT_i

.10 Then $e^{-\frac{T_i}{A_i}}$ is same for all i so $T_i \propto A_i$. so $\boxed{A_i = \text{or} \propto pc_i}$

and Lsrch so Lsrch works in this case.

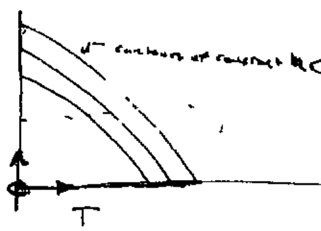
Can I show that this is the only curve type in which Lsrch is optimum?

Say curves are $f(A,T)$. $\frac{df}{dT}\Big|_{T=0}$ is same for all A. (= indep of A)

$\dot{F}(A,T) = \frac{\partial f}{\partial T}(A,T)$

$\dot{F}(A_i,T_i)$ ~~[illegible]~~ $= \dot{F}(A_j,T_j)$ say $\dot{F} \equiv h$

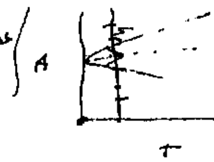
$h(A_i,T_i) = h(A_j,T_j)$



Perhaps any curve like (.06L), that has a positive initial slope & smoothly ↓ in slope, would be ok.

normal ~~[illegible]~~ h is same for all A

.20 slope. Take any curve (like that shown in (.06L)); dilate it in T and p directions by constant so as to keep initial slopes the same. (This constant can be the asymptotic value.) The slope of ~~[illegible]~~ original curve must be monotonic ↓. The curve can't ever be > 1. W. these constraints, the curve will



.24 most likely look much like $A(1-e^{-\frac{T}{A}})$: here we ~~[illegible]~~ took the curve $(1-e^{-T})$

& dilated it by A in both directions → $A(1-e^{-\frac{T}{A}})$

Instead of $1-e^{-T}$ we could use $\int_0^T e^{-x^2}dx$ (~"erf").

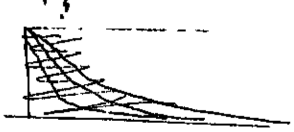
Any integral of a positive, funct. that ↓ ~~[illegible]~~ T>0 will do.

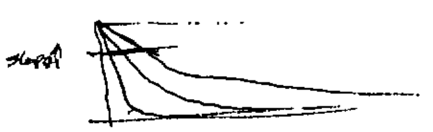
I'm still not sure that .20 – .24 is the only ~~[illegible]~~ curve that gives Lsrch's soln.

.30 [Hvr, in fact, functs like (.06L) are not normal found as characteristics of problem solvers.
(538.35R) & (a 3 param curve) is much more likely/reasonable. So Lsrch is almost always not best way!

→ BUT SEE 590.01 Re: GAMBLING HOUSE THM: $\boxed{\frac{p_c}{c}\text{ order is Best.}}$

.32 $F(AT) \to A\,F(\frac{T}{A})$ $\frac{d}{dT}(A\,F(\frac{T}{A})) = F'(\frac{T}{A})$





I think $F'(T) \to F'(\frac{T}{A})$ is prob'ly right — see 638.23

I think it's the only case where Lsrch is Correct.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 539.40: An alternative to 539.30 (that Gerch is not best), is that one's info about T. different Conds is not in the form of curves of pc v.s. T.

Re: Solns of WON problem: I had the idea that if A, B, C ... were P,T curves for tasks that A+B (meaning new curve corresponding to optimum jumping strategy) was always solvable for any A, B pair. So the Q is is (A+B)+C = A+(B+C) = $\overline{A+B+C}$ "optimum comb. of A, B & C."

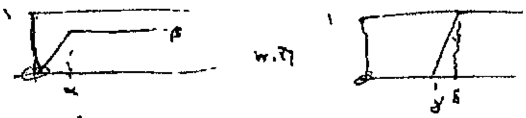
.05 Just what was the criterion for soln? That expected time for soln was min? — The expected time might be ∞ — I Gick that all methods of combining A & B give same pc for T → ∞.
So: when is one curve better than another (if both have same asymptote)?

.09 Maybe take difference betw. 2 curves (this dif = 0 at T=0 & T=∞) — & maybe first Moment.

.10 Any strategy for combining A & B yields a C; so we want a strategy so that C

.11 has min first moment. (? ← very probly true)

An interesting problem: combine

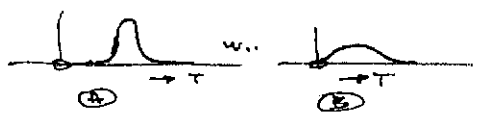


depends on α, β, γ, δ

Certain value sets give simple solns.

In my earliest attack on this WON problem, I took each function (P vs. T) (or some aspect of it); then I divided it into vertical strips of narrow width. Then I would execute the best of each "top strip" of the set of functs. I may have been using Gore (.05 – .09), perhaps p and the derivative (= pc per unit time) of the functions. If so, remarkable.

.20 $\dot{p}$ ↑



given a problem. We would finish B first — and then work on (A) (but never work on it if B → 0 asymptotically as T → ∞ — since A(0) = 0)

.26 $\int(\,)\,T\,dT < \infty$ $\int(\,)\,T\,dt = \infty$

The integrand(s) in .26 is **not F'.**

.29 The analysis of 539.00 → .40, (which put down L's ren) may be "correct" but the curve satisfying the desired functional eqns.

.30 was not F'. — so the commonest curve form may yet be (538.35 R).

.31 Say $f\,dt$ is the probly of success during $T, T+dT$ if we have not succeeded up to T.

The probly of no success to T is $\prod_{i=1}^{T/\Delta}(1-\Delta f(i\Delta)) = \prod e^{-\Delta f(i\Delta)} = e^{-\sum \Delta\cdot f(i\Delta)} = e^{-\int f(t)dt}$.

Probly density of success at time T is $f(T)\cdot e^{-\int f(t)dt} = -\frac{d}{dT}\left(e^{-\int f(t)dT}\right)$

$e^{-\int f(t)dt}$ is still the pc of failure to time T. **Is $\int_0^\infty T\,e^{-\int f\,dt}$ what I want to minimize?**

Need not → 0 as T → ∞. Note f(t) can be any ≥0 function for t ≥ 0

3/13/92 WON 541

ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 540.40 : Say we have 2 f functions, $f_1(T)$ $F_2(T)$ & we want to mix them, ~~say~~ using $m(T)$:

$0 \le m(T) \le 1$, $T \in [0, \infty)$

We want to minz $\int_0^\infty T \, e^{-\int_0^T F_3 dt} dt$. $F_3 = m(T) F_1(T) + (1-m(T)) F_2(T)$

How to get best Mixing function $m(T)$?

Wrong! "Mixing function doesn't work that way! This is wrong.

.06 $F(t) = F_1(\int_0^T m(T) dT) + F_2(\int_0^T (1-m(T)) dT$ ← ?

Prob is not good to spend much time on WON just now; Just assume that a soln. exists — i.e. that we can combine 2 f. functs (via, perhaps .06) to obtain a single equiv. f. funct.

eq. .00 can be genzd ~~from~~ from 2 to an arby no. of f. functs. T. Soln need not be unique.

A reason ~~~~ WON is impt. is that "Modific. of f. Guiding PD" has info in a form that WON can use — but I'm not sure just how Lsrch can use this info! (Also WON seems much better than Lsrch).

I'm not sure (in view of 540.29-30; .31 ff) that the impt. of 539.00-.40 v.s. Lsrch is o.k. or that 538.35R curve is reasonable.

I'm hoping the fuzzy WON approach will somehow be useful for general OZ probs (since OZ probs are ≈ to INV probs)

It does seem that WON methods are directly applicable to pyton problems.

With OTI's one has more info: for each problem, X, OTi pair, we have a for each T, a d.f. or obtainable G. Hyp. w. OTS one has modific. of f. info during search. — (one also has this w. WON)

T. idea is that if, after time T is spent by OTi, it obtains G_1, then this fact usually modifies our expectations of OTi's future performance on X.

EN I think in dealing w. problems like 540.20, I may have approximated them by functions in which f increases goes to a peak then stays there or ↓.

Well!: I want to drop WON for ~~~~ t. time being: I suspect that it is not critical to this part; that I can prob'ly use Lsrch for ~~~~ all 3 kinds of problems — tho Lsrch may not be t. BEST way.

Re: WON: For 2 d functs f_1, f_2, there exists at least one soln to t. OR problem — using Gorc 540.05-.11 This yields an equivt ~~~~ f_1 "OR" f_2 → f_3. So any no. of functs can be combined this way, like resistors in ‖ — This is true when I know how to do f_1 "OR" f_2 or not.

It means that to do n f in ‖ requires ∝ ∼ n

for "Serial" "AND" tasks: there is no way to avoid doing all of them, but it may be, that by working on them partially in various orders, we can get a f. function that starts out as big as possl.

Another trick is proving that any of t. AND tasks is impossl. — this gets rid of t. entire AND set.

In [redacted] a set of "OR" tasks, proving one or more are impossl. also saves much time.

.36 → My impressn is that if we have n tasks in a AND, OR network (like a maze of resistors), the total time spent on deciding which should be worked on "next" is ∝ n.

Th. previous work on WON needs to be reviewed! I think much of it is wrong, but much may be quite useful. A MAIN (likely) result is .36 The details can be worked out by m. means working on a "well defined problem".

3/13/02 E.D. ~~Everything Necessary Master Reader~~ WON (cont)

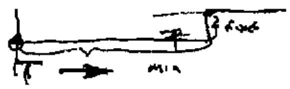
JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

541.40 : The WON is prob'ly true for INV problems,

We can xform OZ probs of fixed CB, into INV probs; we take G (index) to correspond to Time (Tmr.) the INV.

In INV, we want min Time for a given fixed G. In OZ we want max (Min → G) G for a given fixed T — so they sound very similar!

Min T ⇒ Max G. In INV, T is monotonic. In OZ, − G is monotonic (G ≥ 1 but softer — so it can't k).

Inv.    OZ    So, turn OZ problems to converse of Inv

analog of INV; G become "cc", we want to get max G when T end occurs.

Troubles w/ [redacted] OT's ① usually they interact much. E.g. for OT's that do "trials"

② If we have > OT's & we want to know what fraction of time to spend on each;

.10 Best expected G (or whatever) is obtained usually by putting all cc on one or the other.

.11 [redacted] (usually true if the OT's are "indep")

.12 If they are not indep (e.g. if they use "trials") then perhaps one could switch from one to the other ~~fitfully~~ → 543.00

.13 SN AH! "Soln of AND nets! Say I have 2 tasks f_1, f_2

.15 I want prob'ly that both will be solved by time T. To find this I do $\int dt\, f_1(T-t)\, f_2(t)$ ≠ "S_n"

So its a convolution of (f_1 & f_2)! No! $S_1 * S_2$ ($s = f e^{-S f}$) Very simple! Hvr, the Prizing being convoluted may not be f_1, f_2 but $f e^{-S f}$ or whatever (← Most Likely!!! / Most Likely ~~Normally~~)

This does not include any one's trying to prove f_1 or f_2 is impossible, hvr.

→ NB There is also ~~a question about~~ time sharing S_1 & S_2 so as to Minz expected time of Soln. ~~Is this wrong?~~

Given $f e^{-Sf dt}$ & find f; integrate → $-e^{-Sf dt}$; ~~a~~ ln e^{-Sf} → Sf, differentiate to get f.

.20 So $-S$, $\ln(\ )$, $\frac{d}{dt}$ | inversely $f \to -\int f$, e^{-Sf}; $\frac{d}{dt}(e^{-Sf})$

$-S + \ln + (\frac{d}{dt})$ v.s. $-S + \exp + \frac{d}{dt}$

$s \equiv f e^{-Sf}$

$f = \frac{d}{dt}(-\ln(\int s\,dt))$

$= \frac{d}{dt}(-\ln(S\,s\,dt))$

$f = -\frac{d}{dt}\ln(-S)$

S x^2 $\frac{d}{dt}$ v.s. S $\sqrt{x}$ $\frac{d}{dt}$ is the inverse

$S\ H(\cdot)\frac{d}{dt}$; $S\ H^{-1}(\cdot)\frac{d}{dt}$ inverses!

$(\cdot Sf)^2 \to (Sf)^2 \to 2fSf$ |

Def s, f — So $f(t) \to s(t)$; $s(t) = -\frac{d}{dt}(e^{-Sf})$; $f = \frac{d}{dt}(\ln(-Ss))$; somewhere f & s formed after a ball tuck.

.30. $f = -\frac{d}{dt}\ln(-\int s) = \frac{s}{\int s\,dt}$ $s = f e^{-Sf}$

$f = \frac{s}{\int s\,dt}$ / $s = \frac{f}{e^{Sf}}$; $\frac{f}{s} = \frac{1}{\int s\,dt}$ $\frac{f}{s} = e^{Sf}$

$e^{-Sf dt} = \int s\,dt$

$F \equiv \int f dt$ $S \equiv \int s\,dt$. $f = -\frac{s}{S} = \frac{S'}{S} = \frac{d}{dt}\ln S$

if $\mathcal{F} = -Sf$ & $\mathcal{S} = -Ss$ then $\mathcal{S} = \ln \mathcal{F}$

see 572.23–.24 for derivs.

So use $\mathcal{S}$ and $\mathcal{F}$ to desrb. distribution

also $-\dot{\mathcal{S}}$ & $-\dot{\mathcal{F}}$

$s = \frac{d}{dt}\ln \mathcal{F} = \frac{f}{\mathcal{F}}$

on form of $F(cc)$ of $f(t)$ → 623.36 ff in particular → 623.16

3/15/02
ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: (spec 542.12) : Re 542.10–.11 : Often, after one has worked on x w. OT_i, it will become clear that OT_i is not v.g. for x, & that one should switch OT's.

Perhaps we can have assoc. w. each OT_i, a f function (like for INV probs) that tells How Discouraged we get when we haven't found acceptable soln. after cc_T.

.03

[redacted] Retracing a bit: Some confusion in my Mind about f v.s. S (542.15–.40).
How can F be the probty density of soln at time T w.o. knowledge that the problem has not been solved up to T?
Well, O no way: S is defined as probty of no soln up to t & soln. at T, f is then defined in terms of S + (See 542.15–.40 for xfmas bet S→f & f→S)

10 So far the $S_{12} = S_1 * S_2$ result of 542.13 } Are of much interest
stuff in Lsrch of 539.05–.90
Re F⟺S equvs of 542.20–.40 .36.

However, to understand & solve WON now would require a lot of time to "get into" what it's really about.
Also, I may have some usable solns in older writings.

.16 For now, I want to know if I have some reasonable approaches to the 3 probs!
① INV: Use ordinary Lsrch in [redacted] TSQ. We are looking for a single soln. for all the probs → (OM ←)
② IND " " " " " " single soln all IND prob
This is the standard QA problem.

20 ② has scaling problems of the usual types & see 535.20–.29 for some new kinds of scaling.
It's likely that those same scaling types occur in ① as well.
Also Breaking problem set into subsets of problems of different kinds (using a 'Recog function')
can certainly be used in both ① & ②.

③ OZ probs: ([redacted] Usually fixed CB but sometimes "Anytime" probs.) : So I start w. This set of OT's, & a "frontend" that assigns OT's to problems. At first, The OT assigned to IND problems is Lsrch. Similarly for INV probs.
The next stages of improvement make context dependence of coeffs, & "Recognition" of types of problems.
Since these are Heurs, they can be regarded as improvements of the OT's ← I'll have somewhat "do detail" of the general idea that "All heurs are improvements/modifics in the associated OT's.)

30 I want to integrate this idea w. the (weaker) idea that almost all search heurs can be expressed as modifics of the "Guiding Pd" used in Lsrch.
Another kind of improvement in the set of OT's is obtained by expressing e.g. set of OT's as a lang. w. a grammar to extrapolate the set.

I want to design a Minimal TM, in which almost all development (insertion of Heurs) is implemented via TSQ. (or occasionally "equivalent" i.e. we may write in a heur but only if we can derive a TSQ that would give the same results. ← This may be usefully regarded as a normal "necessary" "USR input")
Possibly I will do some external Advice by trainer of how much CC to spend on SD & where to do the SD.
To start, I need not treat all 3 kinds of problems. I may start w. Induction probs! first deterministic induction

3/17/02 ID [redacted] 5~~44~~ 544

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 543.40 ; **Expo**. A possl. way to present t. ideas; Use Scaling problems to introduce rudiments.
Just start w. Lsrch for induction (or INV): → Give actual example of prob. too large to Lsrch. Give actual numbers.

1) To start, we can only work very small probs w. direct ~~or~~ uncondl. Lsrch. So we introduce TSQ's. → Time must be 2^{..} here using

.03 2) Next; scaling due to ↑ in no. of cones so $pc \to \frac{1}{k}$ (k = no. of cues) → $\frac{1}{n^d}$ for a d cone soln.
. Use context dependence to fix.

.05 3) Testing [redacted] time ∝ n (n = no. of QA's): Teach TM to recognize difrnt problem types.
[redacted] ; when we unify various types of probs, we may be able to do some "statistic testing" of corpus, — which is another way to deal w. (3).05)

.07 4) See **535.17–29** for a few new types of **scalings**
a) very large data sets → Much more hard sensitive S.I. at point to problem soln.
b) very long soln. times (Big problems) : tech. requires for very large probs can be significantly difrnt than those for small probs.
c) 535.26

In general "Very large problem" means (usually) that

.10 TM will be aware "self aware" (i.e. "know about outside world & how to take advantage of it to solve its problem.")

So (4) is probably t. ~~least~~ ~~most~~ difclt problem; i.e. t. problem of **RTM**. — That solving t. problem may involve undesirable "sideeffects" or that TM may find a way to satisfy the constraints w.o. solving t. problem (e.g. self-stimulation, Junky).

.12 → 545.00

SN The "Problem Pool" form of TSQ did seem to solve many TM problems in a natural way.
Hvr, it seemed like I had much less control over TM's education — so good possy that I couldn't get it to lrn certain things as "I think I do them". My impression is that I'd like to start TM w. t. usual type of TSQ, then possibly switch him to a "Problem Pool" when he is smart enuf to benefit from such training of that sort.

An [redacted] apparent diffy w. "problem pool": It is too close to RW — So the TM would be more likely to discover relationship of RW to itself. — Very "Uncontrollable"

.20

SN How to "Pay" "patrons" that let us use their computers while they are not using it.

1.) Give them some parts as ~~users~~ free users of TM!
Trouble is users might be Military!

2) Give users large monetary prizes when Lsrch finds soln. on their machine. (It may be difclt to assign credit for many solns).

3) Have lottery! Probty of win ∝ no. of hrs. spent on usr's machine.

→ 4) Somehow have payment that gives usr sense of participation! — Perhaps send items of stuff problems solved. Perhaps leave out enuf so usr doesn't have an adequate TSQ.
.30 Explain why ~~put~~ these problems were chosen — why impt.

5) Perhaps try to get F.B. from users — suggestions on how to improve performance, How to ~~to~~ deal w. various diffys. There may be too much to be able to read / understand.

→ 6) Send users free SW. that TM designs for various problem sorts, e.g. a symbolic integration pgm. Voice recogn. Voice generation, Some text understanding, etc.
Debug Linux a/o any other [redacted] "Free" SW. ; Better solns t. Trav. Sls. Prob. — In terms of what it is practically useful.

3/18/02 ID AI? Nyquist 545

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: (544.12): Well, say I had a reasonable TSQ for QA. How far am I in solving 544.03 (context dependence of pc's of concs), 544.05 Recogn. of diff. problem types a "sampling" of corpus for testing. ?

.02 In context dependence I may have very large data set for induction → Scaling problem 544.02a

.03 → If I solved these 2 probs: How far away would I be? → 546.01

.05 Would TM be able to (in hours) of r. form that modify PC Guidy Search? note → 03

.06 " " " " " " " " " regard IND/INU probs as OZ problems in

[redacted] = express hours as OT improvements in (PC problem) → pc's of OT ? (improvements in)

Just how does TM implement these kinds of Hours?

NB In .05 I had mainly (maybe exclusively) considered hours found by "empirical induction"

.10 That a certain search tech would (or would have) work in several past cases. ABCDEF

On the other hand, most hours are obtained (by humans) by (Logical/probabilistic) analysis.

However

On the other hand, discovering good context, may not need "Mathematical reasoning". At first level, a complicated context is a purely (empirical/statistical) concept. What kinds of context seem to be assoc. w. given conc? — The context can "be found" to conc, or be "related" to it in any derivable way. Only "simple" relationships will be observable. If we want a particular kind of relation in context to be used, we may have to teach TM that it is a useful kind of context.

At first adjacent symbols will [be] (constitute) context", Also "classes of problems" will be early, by pc. "contexts". In general a "context" of a (conc/symbol) will be a situation in which it occurs.

To make sure TM can find needed contexts: Look at list of various context types that I find important. Say that TM has vocabulary of concs to express these contexts. —

If a context is diff. (for TM) to express (slow pc) devise TSQ to ↑ its pc.

Q: can context ↑ pc's (FAST) enuf to offset ↓ of PC as corpus Grows? Recency

.24 On "RECENCY" "Context": W. a suitable Recency context, TM gets equiv. of Finite Memory (it can still be quite large). This may not be a bad idea! We could simply have exponential decay of pc's of concs if they are not used often (or at all). If TM has Finite Memory, it should periodically (or continuously) discard/review Memory contents (or give low pc to) items that it thinks are less likely to be useful ("window" of concs")

.30 In General Recy "windowing of concs" is a way to deal w. scaling problem as see w. a growing corpus, but can become "ENORMOUS! I.E. a TM dealing w. RW Data can be rapidly overwhelmed by "Data Mass", i.e. no. of new concs. E.g. on The Internet.

However, even if we use "recency" to help w. $pc_{main} \propto \frac{1}{k^2}$ we end up w. a v. large PC, so it's still a serious problem.

The ANL problem was "solved" in SAARB was an extreme example of the k^{-2} effect in size of pc of a/n. I put sons of all probs into Memory w. equal pc. So for 3 way accessing, by r. 10^{th} problem Ris was 10^{-3}

.38 If we make pc of (1/distance in past of conc. reference), We get a normalization of Ink. 3 accesses is $\left(\frac{1}{rank}\right)^3$

Why Ris is quite tolerable — Ro it still gets intolerable as $k \to \infty$.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 545.40: We ~~mite~~ use 545.38 → 39 until TM has good enuf context concepts to deal w. this scaling problem. (1) Context dependence of PC's of concs. (2) Recognition of dif'nt "Types" of problems by TM.

.01: 545.03: —— It seems clear that humans always use these 2 techniques in problem solving. → (.11)

.02 [SN] Cost of Computation: Computing takes → 0 energy as one does it very slowly (reversibly). One can get ~~an~~ arbly low energy & still do comp'n. arbly fast; if the comp'n is paralizable. So one exchanges Energy for "Realestate" (= spaceownership) = MATTER. Is $E=MC^2$ relevant? It relates Energy to Matter. I also had that conjecture about ~~an~~ pure serial computation, that Energy for one bit comp'n, Times time to do the comp'n. was Action & ≥ h or ħ. (I think ħ is more likely).

.06 A practical way to look at this: Say one "rents" the computer. Time and Realestate are interchangeable (if the comp'n. is paralizable). In this case total cost to do a comp'n. in 1 second, using 1 cpu at 1 Hz, v.s. 2 cpu's in ‖ at ½ Hz each. So the ½ Hz cpu cost ½ as much as other cpu. ~~Energy~~ ~~overhead~~

Cost = Rent × Time for Computation × (total cost of cpu's).

Hvr. .02 & .06 seem to be Quite difernt ways to look at the problem: ~~...~~ .06 neglects Energy cost.

.10 Thus we have adequate ways to deal w. these 2 scaling soln's.

.11: (.01) ↗ Well, let us drop .01 for a while, & assume that we have adequate ways to deal w. these 2 scaling solns.

.01 is for IND & INV problems. How do we solve OZ probs? We use the GOT (Grand OT)

TM looks at problem x: F(x) gives pc's w.r.t. [OTi] cat. We'd like the no. of OT's w. much wt. to be small. —— but I don't see why this would occur. i.e. we could have a large bunch of OT's ~~...~~ & they are very close to one another. ~~together~~ This is a familiar problem in INV & IND probs. We simply ~~...~~ modify ~~...~~ the pci's before switching OT's (Learning during Lsrch).

.16 So F(x) just chooses the first OT. How long ~~...~~ the trial lasts, depends on how well it does.

During the trial, we are also thinking about other OT's & whether they mite be better.

.18 ~~*~~ 2 things in srch: (1) switching betw OT's (2) Adding/modifying/improving OT's.

.20 →[A very impt property of the (GOT; ≡ OZ solver) is that a new OT that it creates can be arbly good — it can be a new, better way of organizing OTs; so it can be better than original GOT. So, eventually, there will be one main OT, & it will almost always be chosen by F(·). The v.g. OT need not be Lsrch or ≈ Lsrch. (Actually "GOT" as is described in .16 ff is not a real Lsrch — but a gross modification of it!]

So .18 would seem to be the remaining ~~a~~ problems/diffys!

—— In a sense, that would solve most of the "Prelims TM" problems because the INV & IND problems can be solved best by ~~...~~ regarding them as OZ problems. If we had a way to improve OZ prob. solns, We would "Have it Made".

→ [Switching] I (vagely) feel that in (.18) I know how to do the [switching] betw OT's: (conditional)

Hvr, improving OT's, can be very diff: depending on the OT & its "target" (i.e. what it is meant to be used for).

.30 In "Improving OT's", there will be "General Principals" that apply to most/all OT's & also methods that apply to only certain kinds of OT's.

Some impt categories of OT's (1) those used to Solve INV Probs; (2) those for IND probs. (3) Those used to improve OT's (≡ "SI") (4) Those for General OZ probs. Newsylent

.35 Main TM problems (1) TSQ Design for IND, INV & eventually OZ probs.

.36 (2) PC assignment of conc via Context (extended contexts). (Recentcy as an impt kind of context for young TM's.: (546.24))

.37 (3) Breaking up of corpus of problems into Recognizable Problem Types & Dynamic Changes of Categorization. Also Sampling of Corpus for testing ~~...~~ trials (conds).

.38 (4) Initial set of OT's, & initial F(x) to assign OT's to problems & Preliminary "Indexing" of probs by Trainer

.39 (5) Method of switching OTs during soln. of OZ prob.

.40 (6) Methods to improve old OT's & devise new ones & (probabilisticly) reassign them to problem types. (updating F(·)) .38

ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

546.40: (Except for #1 (546.35) (TSQ Design) which is special.) (?)
I have partial solns to each of these 6 problems. How far can I go w. these solns? How far along a TSQ?
The situation may be not so Bad!
Write out each soln., as far as I've Gone: (perhaps w. Bibl. refs.).
I think that this outline should work not too Badly. It seems close to Human Prob. Solving Methods.
(Well, the context dependent pc's of Concs, & t. "Recognition of problem type" & t. "Choice of OT by F(x)" are very "human-like": Lsrch itself is not much "Human-like", but TM can eventually effectively replace Lsrch by any other OT — that it finds "suitable".
Well, Lsrch is a Bit "Human"! We do normally search over t. cands of highest Apriori FIRST.
We also do much "Logical Reasoning" — {but until TM is suitably trained, it can't do that!}

Perhaps a (another) main argt. for the workability of t. Progs. Model: That any modes of Solving Problems (Heur or Whatever) can be lrnd by t. model — It may be nasty for much Skinnerish TSQ or Broad "Hints" — or even "wiring in", but it can be lrnd" (acquired). After an adequate set of "tools" have been acquired, I think that any additional "Tools", heurs or whatever, would be acquirable by t. system in an "Acceptable" Manner.
The other Big workability Arguments: ① "If all info is in Guiding Pd, Lsrch is close to best."
② For OZ problems, t. system can acquire any conceivable OT — which could then effectively replace the current GOT.
③ All Heurs are expressable as OT's. (This is a stronger statement than ①: which assumes all heurs are expressable as "Modifns of t. guiding Pd".
How do the (x20) ideas of 512, 515 fit into t. above? They are usually trims / improvements.
the ⑱ on "Indexing" is probly quite impt. | Impt. w. ter ABCDE abcdefgh ABCDE abcdefg |
546.35–.40 is a Good Summary of MAIN PROBLEMS.

So: Th. Report:

3 kinds of probs: INV, IND, OZ:

N.B. we may leave out INV probs, since They can be expressed as IND probs & I don't know of any advantages to solving them directly — It may be simpler; Looking at a INV prob as an INV problem brings in Similar problems! Considering INVs as OZ probs is, hvr, very helpful.

OZ is master problem type: IND is next in heirarchy, INV is lowest.
→ Explain all 3 types, Give examples.
all INV probs can be expressd as IND probs! All IND probs can be expressd as OZ probs.
Why not just solve only all OZ probs? Eventually we will do just that, but for early training of system, it seems easier to have it lrn how to solve these problems in difrnt ways, Then, later when t. machine has had enuf tng. to do so, it will be able to integrate these 3 originally disparate problem types.
At t. beginning TM solves all problems by simple Lsrch.

[There are 3 common Ways to implement Lsrch: T ← ($2^{?}$) T; (1 timeshared or Multiprocessor; Random.) (See Appendix !!)
Each has its own advantages, disadvantages. We will try to use appropriate form for each problem.]

Simple, unconditional Lsrch is fine for ~~very simple problems, but t~~ problems with simple solutions, but as the problem gets ↑ in complexity, the cc of solving them quickly becomes impractical.
Σ Give example: calculate no. of yrs. befor 1000 computers can solve it in 100 secs.
If we try 1000 secs, it would take many yrs if we use 100 computers 10 yrs less.
Give eq. relating # to computing soln time.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

:00: 547.40 In general, as we increase the [redacted] number of problems as well as the difficulty of Prob problems, Lsrch becomes more and more impractical. The inappropriateness of solutions to small problems in new more complex environments is called the Scaling problem. [redacted] Scaling problems of different kinds are introduced when we increase the amount of data used in a problem.

The first scaling problem in Lsrch will be solved by a slight modification of Lsrch:

.06 — If we use a certain technique to solve a small problem, then find the technique inapplicable to larger problems, [redacted] the difficulty is termed "a scaling problem".

Normally in Lsrch Lcost= Time for presentation to M: 2^k : k = length of description of soln $\frac{T}{p}$

.10 we let the search be very guided by a probability distribution. Least = $\frac{1}{p}$ p is probability assigned by Pd. to the soln. As we have noted, $\frac{1}{p}$ can become very large for moderately complex problems. This Pd is an unconditional P.D. that is independent of the problem being solved.

We can reduce that Lcost considerably by using an unconditional a conditional probability distribution to guide the search. Suppose x is a description of a problem. Then using an unconditional probability to guide the search, T/p(x) is the Least of soln.

$p_0(x)$ is a distribution that

N.B. 549.06-.09 is complete solution of INV problems of INP type

Use $P(s|x)$ as the cond'l P.D. Similar to 414.18!

-.18 The desired soln is $\arg\max_s P(\cdot|\cdot) \cdot \prod_i P(s_i|x_i)$ should be Max: But we have additional

.20 condition: that we spend not too much time to find $P(\cdot|\cdot)$: otherwise there is a well-defined formal soln. (Must take a lot of time): i.e. we can code the s_i's as functions of x_i by expressions s_i as an integer: s_i is the n_i'th soln of problem x_i using a certain Reference UM for Lsrch.

→ This is different from 414.18, because 414.18 does obtain an optimum soln in finite time. Yet, .18 & 414.18 seem very similar.

In both cases, TSQ's are very impt. In both cases we can do the whole search by using Modifications of the pc's of concepts used in earlier problem solns.

Try to devise a TSQ for INV probs — Perhaps Polland's [illegible] used for IND probs!

Perhaps try to solve equns! We could do this using either IND or INV Lsrch!

.30 Perhaps IND would get $P(\cdot|\cdot)$ & INV would continue to search to get the actual soln. — It looks like IND Lsrch finds $P(\cdot|\cdot)$. This $P(\cdot|\cdot)$ is then subsequently used for the updates for INV's next problem.

A.I. as the Holy Grail. Find out just what it is H.G. was & what city connotations it has

Note that here $P(\cdot|\cdot)$ is in terms of the Pd. induced by the IND $P(\cdot|\cdot)$ assignment!

After an INV has found the soln, we get new $P(\cdot|\cdot)$, but usually we can then compute the soln, by making modifications — which gives the P.D. for the next INV Lsrch.

Q: Would I be able to teach TM to solve equs rapidly, using a TSQ of this kind? How can I teach it the [redacted] concepts (needed/adequate) for solving equs rapidly? —

It seems that I can do it using a TSQ of IND problems: Why are INV problems different (If indeed, they are!)?

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 548.401 INV Lsrch uses AZ141 in same way as IND Does. It looks for functions that have been useful in t. past & defines them, so they are more likely to be tried in t. future.

[SN] How does TM use AZ141 to take advantage of common sub-nets in t. durns of functions? Well, any sub-net is a function, & we can re-code t. "Grand" function by dividing it up into sub-functions in many different ways.

.06 The INV problem becomes a trivial applicn. of IND! We have this set of [S_i [X_i] pairs]. X_i is a problem & p_i is t. pgm that was found to solve it. We simply apply 414.18 ≡ 548.18 to it to get t. best possl. pc for that (.06R) corpus. Given a new problem X_{n+1}, we get a good dfn. for S; and a large $P(\cdot|\cdot)$ for the correct soln. means that Lsrch will be fast! — PROBLY WRONG! see 568.22-31 also 572.03-28

.09

.10 More on ["scaling":] Another type of scaling occurs in IND probs. w. very large Corpus. We usually don't want to code t. entire Corpus (we don't have cc) so we try to select t. part of t. corpus that is most likely to be relevant.

Another factor affects this like "scaling" but not exactly t. same: As TM learns new things, various new techniques become possl. e.g. logical reasoning, Ability to conjecture useful thms & prove them. This last is a very skill for Hsrch to become useful. Also learning to read English!

.18 In t. discussn. of INV Lsrch; It seems clear how t. problem X_i is made after Chyet pc of solving by expressing t. Guiding Pd as a conditional pd. —

.20 T. "conditions" being [S_i | X_i] ? i.e.

In t. IND Lsrch, (perhaps) we do this by making t. Guiding P.D. th. coded "content". This may be a better way to clarify (in my own mind) just what t. "content" P.D. is. I had been considering it to be a P.d. for individual codes in t. soln, but this is just an elem. It's really a pd for t. $O^n(\cdot|\cdot)$. I think I did think of it that way in some of my earlier work.

Ideally, I wanted a pd so that O^n would look at t. entire past history ("trace") & get a v.g. P.d. for O^{n+1} — which is used for Lsrch.

This last seems to make it clear that we want to do "long" during Lsrch, so t. "Guiding Pd" is updated as frequently as possl.

.30

Go over .18 → .30 very carefully. It seems very impt. in clarifying just what's going on!

In .18 → .30, i.e. going from [redacted] INV to IND, we improve t. srch. for t. INV problems at 2 levels. Is there another level? — or an ∞ of levels — so one could do a recursive improvement?

.36 In fact, I did do that by (eventually) having TM use t. GDT to work on improving GDT.

There is a lot of "Handwaving" in .18 → .36! At present, my mind is not very clear on this!

3/22/02
ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.001 549:40 ! So first order Lsrch for ENV: using fixed Apriori Pd. $P_0(\$)$; we do Lsrch Guided by $P_0(\$)$.

Second order Lsrch: To solve ENV prob. desc'd by X, we use Pd. $P_0(\$|X)$ a cond'l. pd.

Initially $P_0(\$(X)) \equiv P_0(\$)$. Hvr. as the corpus of prob, soln pairs $\{X_i, S_i\}$ comes in, we modify P_0

.03 So that ~~[redacted]~~ apriori $P_0(\cdot|\cdot)\cdot\prod_{i=1}^{n} P_0(S_i|X_i)$ is maximal. = M

I had a way of showing that .03 was an approx to the general universal conditional PC..

The universal cond'l. Pd is: $\sum(\alpha_j P_j(\cdot|\cdot))\cdot\prod_{i=1}^{n} P_j(S_i|X_i))$ If we include $P_j(\$|X_{n+1})$ in the product, we get a pd on [redacted] $X_{n+1} \to S$.

If we have 2 functions in .03; P_0^1, P_0^2: The one having larger M has a better fit ~~to~~

.10 (in MS error sense & in Max likelihood sense) & is "∴" more likely to have "good fit" (or same entropy per problem" in the future.

In the present context, .03 is a strategy to get a large $P_0(S_{n+1}|X_{n+1})$

So Now, how does "Extended Context" fit in? We have this New problem: the END problem.

.03 is that problem. The first order soln. to .03 is simple Lsrch — using AZ(?) language & PC assignment.

This "simple Lsrch" means Unconditional Lsrch. We can improve on that if we use Cond'l

.18 Lsrch. Say [redacted], $P_1, P_2, P_3 \cdots P_n$ are successive P's that we've found for [my serial] 1, 2, 3 — n problems.

.19 So, we set of "pairs" for induction is $P_1, [(S_1, X_1)]$; $P_2[(S_1, X_1), (S_2, X_2)]$; $P_3, [(S_1, X_1), (S_2, X_2), (S_3, X_3)] \cdots P_n, [(S_1 \cdots X_n)]$

.20 We then use "simple" Lsrch to solve this ~~[redacted]~~ which gives us [redacted] trials for $P_{n+1}(\cdot|\cdot)$, that tend to be [v.g.] for .03.

[.18 – .20] is an ~~[illegible]~~, induction problem, so we can use [the methodology of] [.18 – .20] on it!

.22 → Hvr, if we want a "Grand" Cond'l. P.D. The data for P_n .19 should also include the set of pairs, $\{S_i|X_i\}$ $i=1/n$.

.225 [I'm not 100% sure of .22]. → The that's what Act Corpus Thm is about! MGT → $51.00

Anyway, we ~~[illegible]~~ do have a [not infinite] but "regress of depth n".

So we prob'ly want to [do the] ~~[illegible]~~ it recursively (since n will [quickly] get large enough to justify this).

But: How is [just] even (.18–.19) Related to my idea of "Extended" Context" ?

.29 Also, would not the language stuff (& recursive translation, say) include "Breaking up of corpus into Recognizable problem types" in itself? [→ Partitioning Corpus] Unclear, since this Heur is ~~[illegible]~~ (~~[illegible]~~ (apparently) mainly)

.30 ~~[illegible]~~ for ↓ cc of testing Conds, (like [Quick Abort] & may not be discoverable by techniques

Such as these. [Improving the all over GOT can, hvr, deal w. "Q.Abort, & perhaps to ["Corpus Partitioning"].]

Whether the recursive method can ~~[illegible]~~ eventually get [Corpus] Partitioning or not — we want it fairly

.34 Early, since it seems like a v.g. heuristic in several [redacted] ways.

Another Heur not discoverable by finding "better PC distribution for soln. is [Hutter's method] of

↓ testing time by looking for [Theorems] [elems] & proofs; [Heurs of this sort are findable by improved

OT's (≡ GOT)]

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 (spec 550.275): Q: What about "Multiple Double Counting" in Art cognitive MCT? [May be very IMP Q] — In 550.18–.22 Reparticular

.01 [SN] In t. reduction of CC of Verifn: One approach is to use random statistical testing in attempts to quickly find probs that reject t cand. ~~~~ probs can be ordered by simple statistical selection of t. probs marked by past freq. of cand. rejection. Another (more complex) approach would be to have a separate TM (maybe a GA — or maybe a TM operating concurrently w. same as t. Main TM — perhaps via a "Call" to t. main TM) try to find probs that t. cand cant work. It should do this by having a P.D. on past problems that is a function of t. cand to be tested. T. Algm looks at t. cand & try t. guess at a proper set of probs that is likely to reject it, rapidly ← So we

.10 [redacted] also weight probs on basis of how long we expect Verifn. to take. This (.01 ff) could essentially reduce t. verifn. time to a small amount: Hm since CC of a problem is ∝ verifn time for soln. (well maybe ∝ that using .01 ff) — anyway, we can t. verifn time further by a large factor by Partitioning Corpus.

.15 Note: In t. "Second order Lsrch" of 550.18–.20: It's not really "complex"/similar to Lsrch for INV, because in INV Lsrch there is only one soln. — So t. ideally, t. Guiding P.D. first should give t. pc of a could being "Best". — So it becomes a standard OZ Lsrch, w. fixed CB. (Since I have no good rigorous defn. for a "Anytime" problem.) We could just pass a CB — run it for that time, then do T ← 2T, repeatedly. As T changes, t. Guiding P.D. should also change. Even tho we are looking for a diff kind of obj fn, the "Grand Cond'l PD" can be t. same for INV problems & all Levels of [redacted] IND Lsrch (in MCT).

.20 In .15 t. 2 srchs are closer than it seems, because t. higher order/srch is done (initially) by simple Lsrch — just like t. INV problem.

Since it is an Lsrch guided by a P.d., we can always (recursively) ask for an improvement of that Guiding P.D. — which leads to a yet higher order Lsrch — so we can do this ad. OO (and.). & it looks like some kind of single recursive search would be appropriate & correct.

.28 To start off, we have 3 "guiding PD's" one each for INV, IND, OZ. — They are hierarchical. The IND P.D. is to improve t. INV P.D.; the OZ P.D. is to improve t. IND P.D.

.30 Would have P.D. to improve t. OZ P.D. (by Lsrch etc).

But actually, there is only one "Grand P.D." (viz MCT). This P.D. uses certain "parts" of itself to improve other "parts" (t. "parts" are for different Universes of "Condition" values.)

.33 For OZ probs, The "condition" argt always includes [redacted] t. problem & t. CB. → (Note .39–.40)

At t. present Time: Unclear: ① Just how we go to recursion & how use of a common "Grand Cond'l. pd." May do recursion. ② Even if we used 3 pd's for t. 3 types of problems: How does th. main "Partitioning" heur. fit in? — How is it introduced into t. System? → [is on this Q: 550.29–.34]

How are other Heurs introduced into t. system as Modules (or New) ops?

.39 Perhaps t. "recursion" is already in .28–.33 in that t. P.D. that guides t. improvement of the

.40 P.D. is that same P.D.

3/24/02 ID

UNCONDITIONAL LSRCH is TOO SLOW .11

552

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 : 551.40 ; Looking at 530.18–20, 29–34 it would seem that Quick sorts, partitioning, Corps, Hutter's Lsrch method, statistical theory of Cards would not be discovered by 530.18–.20. They are improvements of a OT, but a not a

.02 different kind — they do not operate by improving P.D. Rational to Guide Lsrch

As for Hutter's method, one could use it instead of Lsrch for all INV problems. At first no proofs would be found, but later, when TM had learned how to do proofs, it should begin to try them seriously.

I Guess the way this works: TM acquires a good pd that's able to look at a INV problem & "suggest" (= give hypothesis) a method to solve it of V&b Sound PDs, & prove its pd (w.o. directly using the Vanilla method that defines the INV problem). → 555.00

.10

.11 SN Onshowing Ucond Lsrch is usually impractical: Min soln time for Quad. Eqn ?

1) use double precision to prevent direct trials: 32 bit integers: [$10^9 \approx 2^{30}$ ~ 45 yrs.

So $2^{32} = 4\times10^9$ so 32 ints would be directly solvable.

64 bit would be $2^{30} \cdot 2^{34}$ = 34+17 = 51 yrs. : Using AZ141 (for $F2 = X_1(X_2^2+X_3^2)$) I got $pc \approx 1.47\times10^{10}$

Compute [illegible] pc of quadratic formula in AZ141: Assume sqrt takes 1 time unit (!):

~~$\sqrt{B^2-4AC}$~~ $\frac{\sqrt{B^2-4AC}+b}{2A}$ — B^2 [illegible], $-4AC$ [illegible], sqrt = 2, $-b$ = 1, $\div 2$ = 1, $\div A$ = 1; 8 operations = 2^3 = 4.5 yrs

For 1 expression. $X_1(X_2^2+X_3^2)$ $pc^{-1} \approx 1.47\times10^{10}$: w. cc = 8 → 1.17×10^{11}

.20 This used only 6 primitive symbols: 0, 1, X mul sum, F_1. If I used 0, 1, X sqrt, mul, div, sum, sub, F_1

the pc^{-1} would be lot. For $(\sqrt{B^2-4AC}+B)/2A$, the pc would be much larger. Looks at least unclear case Complex — 2×10^{20}

[illegible] 2B & could be Sum 1, Sum 1; Sum 1 Sum 1 Sum 1 Sum 1, Post; Or, I could define a function E

F1 = Sum 1 Sum 1 [illegible]; then 4 = Sum F1 F1. I want to define F1 just beforehand, so I don't waste [illegible] ~~or~~ = mul F1 F1.

$\sqrt{}$ cost by 4 × cost of other symbols. Anyway this may not be so. 2 & 4 are used only once each, so it wouldn't change result much.

For $(\sqrt{B\cdot B-4\cdot A\cdot C}-B)/2\cdot A$ $= 2^{74.375}$ → $x^2-2x+1=0$, $x = 1$

I got $pc\,(pc^{-1}) = 2.45\,E22$: Divide by 2×10^9 Hz CPU ($\approx 2^{31}$) $= 2^{74.37-31} = 2^{43.37}$ → 65 yrs. not 71.6 !

& take 1.5 $\ln_2(\;)$ → 71.6 yrs. If count takes 2 clocks,

in 71.6 yrs Lsrch will take seconds to find this soln.

20×10^{18}, $2^8\times2^{60}$, 68×1.5 = 102 yrs. 34

.30

See 553.00–553.40 for some "Analysis".

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

-00: 552.40! PC of $\sqrt{B^2 - 4AC}/2A$: (B) (6) $X_1 \equiv A$, $X_2 \equiv B$, $X_3 \equiv C$

symbol	Replaceable symbols	num	Damen	Σ no. of ... symbols
X				
sum				
sub				
mul				
div				
√				
0				8
1				9
√	x, sum, sub, mul, div, √, 0, 1	1	8	10
sum		1	9	
mul		1	10	11
B (X		1	11	12
2		1	3	
B (X		2	12	13
2		2	3	
mul		2	13	14
mul		3	14	15
A (X		3	15	16
1		1	3	
C (X		4	16	17
3		1	3	
sum		2	17	18
1		1	18	19
sum		3	19	20
		2	20	21
sum		4	21	22
		3	22	23
1		4	23	24

107 x 10^4

$10.7 \times 10^{9} \approx 10.7 \times 2^{30}$

$\times 1. \times 2^{40} =$

5.4×2^{50}

(x08) ≈ 26 so ≈ 2^{54} → 81 yrs → 537.00

54
27
81 yrs

woops! Div X1 Div2
or Div Mul X1 2

first Div Div 24 25
A(Div X 25
X (B)
4 Div 26
sum sum 27
28
1 29 30

This is very (identical) close to Z141! except for subscripts of X.

so 29! damen probs × 8! num. $PC = \frac{8!\ 2!\,3!\,5!\,5!\,6!}{29!}$

no. of occurrences:

	no.
X	5
sum	5
mul	3
div	2
√	1
1	6

× 1.64 E 3.

so rc = 1.76 × 10^18 (w.o. rcost of indexes of x)

$5!\cdot5!\cdot5!\cdot(6\cdot6\cdot2)$ 72

· 8! = 5.1016 × 10^12

÷ 2×10^9 = .8 × 10^9 = .8 × 2^30 ≈ 45 yrs.

Σ = 22 + 8 = 30; 30 − 1 = 29 so 29! for damen.

(numbered.)

$\frac{1}{3}\frac{2}{2}$ $\frac{1}{3}\frac{2}{3}$ $\frac{1}{3}$ = $\frac{4}{9^2}$
6 A C
$= \left(\frac{2}{81}\right)^2$

= ≈ 1.31 E3 × 2^30 = 1.31 × 2^40 ≈ 60 yrs.

rc including index choices: 1.76 E18 × 1.64 E3 = 2.88 E21 = 2.88 × 2^70

2 × 10^9 ops/sec = 2 × 2^30

so $\frac{2.88}{2}\cdot 2^{40} = 1.44 \times 2^{40}$ or ≈ 60 yrs.

rcost of 4 indexes:
$= \frac{9^2}{2^2} = \left(\frac{81}{2}\right)^2 = 1.64E3$

woops! I forgot to add in B^2.

.90 sum mul x x 2

4 more symbols. damen 30×31×32×33 / X 11 × 12 | num sum 6 mul 4 X 6·7 2·2 → × 9.33 × 10^5

rc = 107 ...

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: (553.10Ri) Conclusion: To Express $\frac{B \pm \sqrt{B^2 - 4AC}}{2A}$

requires: [redacted] 72 yrs: Assuming a 2×10^3 MHz computer today — cost ~ $k, some 71 yrs. At computer will ... such things in ~ 100 sec. (if it takes 100 clocks to test the formula.)

29! → 31! denom. num $\frac{6}{31}$ × $\frac{4}{10}$ = 6.88 E-3 : $(\ )^{-1}$ = 145.3 = $2^{7.18}$

7.18 × 1.5 = 10.77 yrs. So 60 + 10.77 = 70.77 yrs.

T. accounting. x 5+1=6) so [redacted] .87 yrs indices A B C $\frac{2!}{3^2}$ $\frac{3!}{3^3}$

sum 5 + (1 = 6)

mod 3

bin 2
sub 1

! 6

Σ = 24 !

.60 So $\frac{8! \quad 6!\,6!\,6!\,(3!\,2!)}{32!} \cdot \left(\frac{2!}{3^2}\frac{3!}{3^3}\right)^{12}$ [redacted]

num: 2.167 E15) num. 2.95 E22 operations
denom 6.39 E37)

÷ 2×E9 = 1.47 E13 = 14.7 E12

14.7 2^{40} & 2^{48} ; 48 × 1.5 = 72 yrs.
= $2^{3.9}$

one fewer sum
bin over sub.

So [redacted] num 6 → 5 sub → 15

So $\frac{5}{6}$ → .4 yrs less so [redacted]

72 − .4 = **71.6 yrs.** If it takes k clocks to do each, it will take k seconds to

do each [redacted] 71.6 yrs in future.

.20 Make it clear what the primitives were.

A simpler derivation would use A, B, C for primitives, but not x. — So 10 primitives.

24 symbols → 34! ; Instead of x = 6, we have A 2 B 3 C 1

2!·3!·1 = 12 ... 720 in numerator for factor of 6.

factor of ... $35\frac{5}{9}$ = 7.72 yrs. So it seems that using A B C rather

Actually 720 × $\frac{12}{3^5 \times 2 \times 3}$ = 35.5 only

than x1, x2, x3 is slower by a factor of 35.5 (or 7.72 yrs).

Actually is worse: 35.5 × $\frac{33 \times 34}{9 \times 10}$ = 443.35 → 13.18 yrs. 84.88 yrs

Anyway, using A, B, C, the results are exactly the Z:4:1 formula.

" $X_{1,2,3}$ " " Z:4:1 formula but w. special simple rule for the 3 indices.

.30 Z:4:1 formula = (Σ base) $\frac{k!\prod_{i=1}^{k}(n_i!)}{\left(k + \sum_{i=1}^{k} n_i\right)!}$ ← k is no. of distinct symbols. n_i is the no. of times the i-th symbol type appears in the true output.

i.e. for k = 8, the pc of the dash symbols $\frac{1}{8}$ not $\frac{1}{9}$

.33 More likely: $\frac{(k-1)!\prod_{i=1}^{k}(n_i)!}{\left(k-1+\sum_{i=1}^{k} n_i\right)!}$

[redacted] I should modify above calcs (perhaps) in view of .33

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

:00: (552.00 spec) : A (very possl.) Objection to 551.28-.33; .39-.40 is 552.00-.00 That is methods used only improve to Guiding P.D. & they don't help e.g. to discover "Quick aborts." Or, more Generally, they do not Use the most General methods of improving Srch.

.03 On Second that, they write! Consider OT's: we want to improve them. We do this by looking at the Empirical G's obtained by them for various CB's. These Empirical G's include any "Speedup" tricks like Q.A. or statistical testing." Also, we use a Universal Lang, so we are sure that we are considering all possl. OT's.

Implied Problem of "IMPROVING OT's" is the "Master problem", because any heur can be expressed as a OT — or Improvement on an OT.

SN Time Series Predn. of MCT!
A specific time series is treated very much like an element in finite BAG induction
Difference is: In Bag induction we only consider codes that eventual stop. In TS Predn, we consider all codes, whether they stop or not — whether they have finite or infinite output.
In TS predn we often have only one element. The may be we often have > 1 element, since less have experience in other D.S's — But can be quite useful. (Viz STEIN — But also other common rep variations.)
All T.S's & Bag induction problems use the same conditional P.D. for "Guidance" of Search; (implemented)

A Possly Trouble w. MCT (Σ not at all sure This is a Legit Complaint): That is, method it uses to solve OE problems does not include Things like Quick Aborts. [It may be that .03-.10 applies to the General MCT (& we'll) — or MCT or that MCT can be modified so .03-.10 applies.

.19 .20 If .03-.10 & .19-.20 are (true/"under control"), then the TSM project is in V.G. Shape!

If they are true, then "Corpus Partitioning" & other Heuristics could be (or Index) by the System.

The having the recursions as 551.28-.33; .39-.40 does not specifically include all types of Heurs, it may not need to: the "Quick Abort" types of heurs will be discovered — tho possibly at not optimum speed — But I have to Think about this More!

.30 So the Most Critical Q now is: is .03-.10, .19-.20 "True"?

After showing it is true, we need to find ways to propose conds of hy expected yield.....
This is a "conditional proof".

It certainly seems that .03-.10 is true! • We take any OT. The OT can include Q Abort, or any Other such Heur. We evaluate it Empirically (usually? — but may be other ways are possl....
Consider Hsrch — Say we can Prove that OT_1 is uniformly Better than OT_2 for all problems!)
In general "Reasoning" can shorten cc considerably: In evaln of OT_1, say, Partitioning of Corpus makes it possl to be sure that certain parts of the Corpus will not have their G modified by a certain "key part"/often algm.

556.00

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

:00 : 555.401 "Empirical Evaln." means that we have tried this OT on various problems & we have a set of [problem$_i$, CB$_i$, G$_i$'] triples assoc. w.t. OT. From this, we obtain a general P_1 distribution that gives, for this OT, and any problem, P_j = p.d. fn. each CB$_j$ on Fi/reasonably good G$_i$'s. from all the OT's. Given a new problem, we can then compute t. probty that each of the OT's will give best G w. this CB. After we have worked on t. problem a while, our current of expected G vs. CB changes & we may switch to a new OT.

.04

.09 This seems to look like a WON problem, but is certainly a bit different from Normal WON!

(WoN is for INV probs so far) (on 542.00) I considered whether a way to xfer OZ probs into Inv problems, but I haven't been able to get it to work. (542.11-.12 is Disturbing!)

.10 But Note 543.00 -.05, anyway.

Remember: That Lsrch wasn't meant to be a v.g. OZ prob. solver, but that we expected that it was capable of leading (w. suitable TSQ) to a set of V.G. O.T's w. rather large wts, for almost all problems. Its virtues for INV probs (& perhaps others) is CJS is available — .15 Mat facilitates t. writing of TSQ's. It is "≅" "optimum" if all ins is in P.D. : .16 [This is wrt. "Most" users, & is true if we learn betw (or during) trials — which makes it no longer Lsrch!] It is relatively simple to understand & implement. (4) Σ n'impt vertex .15 - .16

.20 So, Improving t. GOT: part consists of Generating new OT's: Reuse "evaluated" as .00 - .04 — So we know how & When to use them on new problems.

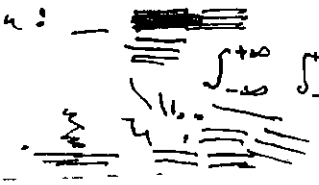
We Generate new OTs by observing t. affinities of existing OT's & extrapolating. We can also Generate OT's via a Universal Alem. We want to do this efficiently, hwr. A "No Brains" listing of Alems, will contain few OT's & be among many, few "good" OT's.

SN Any Type of operation in t. system: We can improve it by "Inductive Analysis" — This will give us a probabilistically ordered list of ends to try for "Improvements"] An Impt. BASIC IDEA w.t. SYSTEM

Anyway in 555.03-.10: We use "not-such-good" regularities to find "Arbly Good" Regularities.

.30 Perhaps call such Induction using only P.D. modifn. for Lsrch improvements "Classical" Lsrch: — Any (Modifn augmentation) of it (e.g. "Free Form Srch".

I might include "Long dorm Lsrch" as either Classical or Freeform & Unclear)



Draw up flow diagram of TM, to see what parts I (don't understand / can't make) Tell what happens for INV, IND, OZ probs at various stages of Maturity of TM.

3/27/02 FD 557

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 556.40: 1) INV problem: Given M(·) & c, to find y ∋ M(y)=c: We start w. initial apm given by AZ141: we do Lsrch for p(·) ∋ M(p(y)) = c : Apriori cands, P(·) given by Z141 — W. more Mature TM " " same Lsrch but t. cands " " " given by H(M(·); c) H(·) is a grand/P.D. cands. It looks at t. "object" M(·), c & generates a P.D. on possl. soln. cands, P(·).

Perhaps Desc. "steady state" operation of TM: Then "initialization" & early (pre steady state) operation.

.08 Also updating of various kinds: Updating of P(·). → (.27)

.10: 524.40 — [SN] on Hsrch: Usually for a problem, one can't prove that the sought-for soln. is a certain thing w. certain properties; so Lsrch would find such objects w.o. proving any thing. Hvr., usually in M2A, one wants to understand t. soln. more than one wants to acquire it! — So even if Hsrch usually doesn't work — when it does work, it's much more useful than Lsrch! In fact, if we obtain a Lsrch soln. but no Hsrch soln., we would often feel we found very little! In practical Energy problems, e.g. TSP Lsrch solns. will, hvr, be accepted.

I think my idea of what Hsrch is (which may differ much from Hutter's!) can be generalized much to include all results obtainable by logical reasoning — i.e. t. kind humans usually get.

Hvr, I should read Hutter's paper, he may have already gotten this result adequately | (C)

.20 On t. other hand! Lsrch solns obtained by a conditional Guiding PD can be rather short & possibly understandable to a Human!

(Try this Note to other discns. of Hsrch.) (524.40)

On "Recognition functions" & "Partitioning of Corpus": There are some natural partitions of t. corpus

.24 for Grand PD: e.g. INV probs, Various kinds of IND (Language, Bag. induction, grammar), Time series, OT induction; QA induction.....

.25 and subdivisions of these categories. Compare this kind of partition & its treatment, w. t.

.26 the Recent stuff on Recognition operators for QA problems. ← Various kinds of OZ probs: Noisy, Non noisy, open, closed, partly open; Usable for "Mass" Gore: Also Various Costs of getting for M(X) TM can try to find things for this cost; say approx. (No) time varying M( ) fncts.

$\int_0^\infty e^{-x^2} dx = \sqrt{\pi}$ → 576.16, 594.21 on Hsrch

.27 (.08) [Steady State:] First list t. kinds of probs it can solve (or try to solve) } see (.24–.25) ↑ earlier.

Explain them (to some Extent)

.30 Next explain how each prob. type is solved in terms of t. condl. GPD: Each problem (along w. its "Type" (which is a kind of "index") is t. "conditional" input to GPD. T. output is a PD on "solns". A soln. is a pgm that solves t. problem by mapping prob. domn. into soln. Its Domain & Range can differ for t. various problem types.

[Give D.R. for several prob. types.]

.36 Inv: D: M(·), c ; R: X_i which is its a soln., $M(X_i)=c$.

.37 IND: D: $[Q_i A_i]^n$: R: $P_i(A|Q)$: GPD outputs Pd on $[P_i(A|Q)]$

.38 OZ: D: M(·), T : R: OT_i : GPD outputs is Pd on $[OT_i]$

3/29/02 558
ED 558

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 : 557.40 : 557.37 & 38 (IND OZ) seem diff't / they should be about the same!!

Well, for a mature TM, 557.38 (OZ soln) is used for all problems (including INV).

So 557.36 — .38 is a "steady state" but not for a completely mature TM: I will

have to give detailed development of each problem type & how they end up at OZ.

.04 So INV problems:

(a) Start w. Lsrch using uncond'l universal pd.

(b) Then " " conditional universal pd. (GPD).

(c) Express as OZ problem w. "Meta" "MESA" Goro. Solve as general OZ problem — anything but "Anytime".

No! A OZ problem which we want to get a certain G level in min time. GPD can be arranged so that it gives a P.D. over "OT's" that have No Goal. This needs a bit of work, but I think it shouldn't be hard to do.

.10 IND problems: we are looking for $P_i(t)$ s.t. $P(\ldots)\cdot\prod P(A_j|q_j)$ is max.

(a) Lsrch using uncond univ'l d.f. of P_i's.

(b) " " conditional " " " "Conditions" is d'cn of problem. USE GPD.

.13 (c) express as regular OZ problem using Goro of (10) "as judged by Grammar"

— OZ probs

.15 (a) Start w. set of OT's selected by me. Perhaps use Uncondl. pd. & Lsrch. By "General/apparent Goodness of each OT"

(b) Use conditional pd to guide OZ Lsrch. Conditional TM's partial GPD is obtained Empirically (as in all other cases).

(c) Use IND techniques to Expand set of OT's. General method to Extrapolate: Devise Grammar for a larger bunch of Good OT's (this could be partly done by me) — extrapol. includes empirical utility — "Conditional" ability for solving various prob types (prob. descr is the "condition").

.21 Use IND techniques to assign Cond'l pd's to the fuzz. OT's. → (.31)

N.B. Lsrch is initially "non-learning" during problem but soon evolves dynamic learning during problem.

Maybe unnecessary! See (.31-.36) .34-.36

.20 AN Alternative way to look at (.04 — .21) (More "unified"): That All problems IND, INV, are always solved as OZ problems: But initially the OT's used, decide that the more elementary methods of .04 — .13 are best — then they automatically move on to better methods when these methods become available — i.e. the better methods get more wt. as more evidence is Accumulated for their efficacy.

At first, the "Advanced" Methods OT's get Zero wt., because they haven't yet been Created! The mechanics of shifting from uncond to cond'l Lsrch is simple; but going from cond'l to OZ is not so obvious!

Nevertheless! the listing of .04 — .20 is maybe a v.g. picture of what's going on. — So retain that format for perhaps Exposition.

.31 (.21) Is there a way to do [redacted] express OZ srch as an OZ problem? Go thru .15 — .21 again —

GOOD! (a) Use Lsrch w/ uncond'l pd (Standard universal pd.) — to search & test OT's.

(b) " Lsrch w. conditional pd on OT's.

.34 (c) If we allow generation of promising new OT's (This "Generation" problem may be expressed as a special kind of OZ problem, by the General System), then I think step (b) is as good as one can get! It assigns an OT (of arbitrary kind (No restrictions on what it may be)) to a problem on the basis of "past experience".

.36

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 558.40 | 558.04 – .40 looks V.G.: INV & IND problems go from uncond to Condl Lsrch to OZ probs. OZ probs go from uncond to Condl. Lsrch. — & that's it!

There are various (probably essential) improvements I'd like in Condl Lsrch, updating PD bounds, trials ("long" while srching).

A Big Q is to how to insert the Various improvements into the System: e.g. ("Augmented Content" / "Extended Content") for assignment of pc's of Concs.; Partitioning of Corpus.

["Partitioning of Corpus" can be a "regular" hour (as well as a ~ Quasi-short type of hour) — which could justify using it at an early stage of TM development.]

I think "Extended Content" may be part of "Condl P.D. for Lsrch". AZ141 alone gives Uncondl. pc.

But AZ141 does include Ability to define Sub-functions. ◇ ..... Let's Analyze just What TM does in Simple INDuction: (say) It starts out by constructing [redacted] const. Functions of in ↑ pc order, using AZ141. — These functions can use defns. — even Recursive Defns. At first, the pc of a conc will not depend on its "context" (other than perhaps the "rules of AZ141" that make certain things (legal/illegal) in certain places.) Modification of the pc of a conc at a particular pt. could be based on "previous experience in ~ situations". This seems like "conditional pc" — but it's not yet obvious! Consider it Think about it "another way". —

.19 Say we had a Condl. pc. The pc of a conc would depend on ⓐ T. problem dem.

.20 — ⓑ any "Context" formalism: For each "cond" argument, T. & P.D. can be anything, & this "anything" could be achievable by arbitrary constructs on positions of Various Concs: ≡ Context dependent pc

.22 ... ~3N On Partitioning of Corpus. I think this is a "Normal" way of solving problem by "Conditional pc": Say a problem dem is X_0: we want to put X_0 in one or more "Categories", so it can be solved using methods identical to or similar to, those used for other "members" of its "category" (category ≡ "partition" of corpus).

.25 It may be best (Even in "Baby TM" "AB INITO") To have all problems solved as Optzn problems — This makes all heurs fitable into the formalism. Uncondl, Lsrch & condl. Lsrch are regarded as 2 "early" (elementary) methods of working problems, that may sometimes be usable.

.30 Actually TSQ's need condl pc Lsrch (to be able to do); yet they still have their apparent "Scaling" problem"! So how does the condl. pc mechanism automatically introduce "context dependent pc's"? (see .19 – .21) According to .19–.21 & .22–.25 both context dependent pc's of Concs & Partitioning of corpus should be automatically found by condl pc for Lsrch (probably needing a suitable TSQ)

I want to see just how many "Discoveries" occur. First show that there are regularly actually ↑ pc of a reasonable corpus: then demonstrate the actual Concs (as coded), then try to find TSQ or equiv. or set of "Hints" that could lead to these discoveries.

3/30/02
FD

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

559.40: So: T. general operation of TM: A problem comes in: Is it INV, IND or OZ? — IF

(1) INV: α) If TM is very young, use uncond. Lsrch.
β) if TM has more data on INV probs, use condl. Lsrch.
γ) " " " even more " " " " " " " w. Lrng. during or betw. trials.
δ) " " " " " " It do it as an Optzn. problem, w. suitable OT: It normally converts the Yes/No Score to a continuous (or at least multivalued) Score then uses another OT.

(2) IND (similar to INV) α) Very Young TM: use Lsrch
β) older TM: use condl. Lsrch.
γ) older yet: " " " w. Lrng before or during trials.
δ) Do it as a special kind of optzn problem.

(3) Optzn (or OZ problems, generally) ≡ (Not a INV or IND problem).
α) For young TM Use uncondl Lsrch over set of OT's.
β) " older " " condl " " " " "

That's it!

$\int_{-\infty}^{+\infty}$ $\int_{-\infty}^{+\infty}$

T. next thing deribd is t. update Algm for t. condl. pc (= G PD)

List variations of OPT probs.

Any time, OZ, INV (Get fixed G in min time), Variations of OZ, Time varying M(·) (forbidding), Noisy M(·), Time varying M, Varying cc of delay M(ω). We may know M(·) a little, Lpd on Paysys Personal proofs.

Doc SN INV problems are less like other optimization OPT problem: They differ from OZ probs, in that INV probs try to get a certain Score in min time, (rather than Max Score in Given CB). Also INV probs often use GPS: a vector Score, which is signiftly diffrnt from other OT's.

So t. Q is: why categorize INV probs as OPT probs at all? One answer is that INV probs are ≈ to IND probs (for Younger TM), & TM spends much time as "Youth".

Ultimately, I'm using OPT category as a "Kitchen Sink" free for all — Any technique of prob. solvng or personal proofs.

— Tho I do have various specific categories, any not in the categories are also given t. least included as OT's. $\int_{-\infty}^{+\infty} e^{-\frac{x^2}{2\sigma^2}}dx$

So, When a problem comes in, it is initially Categorized as INV, IND, OPT

It is then further subcategorized (t. results of this depend much on "Maturity of TM")

— To: uncond. Lsrch; Cond Lsrch, opt, other || The for IND, INV, OZ, Cond Lsrch maybe other opt, other —

because it includes any ypm that looks at t. prob. dscrn & then tries to find a soln!

T. Main Connection betw. all problem types is Their Common Condl. pd (GPD).
The GPD can (assign actors) recommend any dscrbable method of prob solving.
(INV IND OZ, OPT ...)
So all types of probs are solved in ~ same way: (early) uncond Lsrch, (later) condl. Lsrch or at set of poss. ways to solve problems.

So, say I started out using unc Lsrch for all problems. For Lsrch, I feed in finite strings in Lexical order. Somehow, for each problem type, t. strings are interpreted as a mapping from problem dscrn. to soln. I guess one could represent probilistic functs (for IND probs) as strings representing Lisp functions — which could represent P.D's using 2 inputs, Spec 561.17

3/30/02
ED

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 560.40 : SN In Dyn. Pgm Problems like "Cure Cancer" What is Criterion for (Merit / Goodness) of a cond. Soln?
<Say t. situations in which choices were made, cannot be repeated.> One knows how certain decisions "came out"; so one knows what pc was assigned to them. (This seems a bit strange! Usually if t. result of a Sci Expt. was given very low pc (much surprise) t. expt. is regarded as 'much more impt.' — Much more "info yield"

This last seems to point to a Big Difrnce betw. Normal "Goals of Science" & What ALP does!
This idea of "Going where t. theory hasn't Gone before" seems beyond what ALP normally does —
tho probly on could adapt ALP to this task .... If one could define t. task!

.10 Note that .00ff discusses 2 problems: ① "Cure Cancer" Genre ② Planning Sci experiments.
They are related but not at all identical! There's a big difnce betw. rsch directed toward a specific Goal ("cure cancer"), & "openended" Sci'c rsch, in which t. goal is "New ideas".

On t. other hand, in t. "Cure Cancer" program, if one got a very unexpected result, this could be very useful is correcting a faulty theory.

.17 (560.40 spec): This would work (in theory), but would be very slow for all but t. smallest problems (see 552.11 ff)
Doing a little Induction on past successes/failures, using (cond) Lsrch would (perhaps) recognize 3 problem types & modify t. function construction algms accordingly — e.g. in uncond Lsrch,
.20 OPT problems would need functions that try to solv OPT problems (i.e. OT's) — almost all funcs are not of this kind .... so usable OT's got very small pc.

Anyway, using cond Lsrch would make things much better.

Hor. usually we would start out (even in "uncond" Lsrch) w. search pgms. that would be oriented toward t. problem type being solvd. — so for OPT probs, we would try only OT's! Later we mite do t. "OT Grammar" method of extrapolating OT's.
We should start out w. as good a cond pd as we can, w.o. unnecessarily "biasing" t. System.
Choosing a Lisp-like lang, & starting w. a good set of OT's is in this direction.

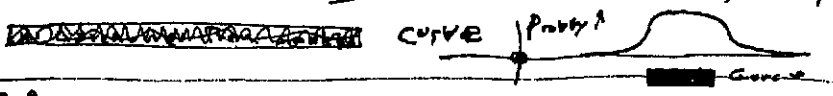
30 Again "in theory" we could start out like 560.37, but with cond Lsrch. At first, tho, there would be little or no inductive info, so t. system would effectively be doing uncond Lsrch.
As time went on, it would do more & more induction & have better & better cond pd's for Lsrch.
Assuming a reasonable TSQ, This TM would eventually become arbly intelligent.
Note that t. pgm of .30 seems to be very simple, & probly easy to write.
.35 We would want "Lrng during Lsrch" (so alternate lrng w. (complete) Lsrch, or do them in ll w. "time sharing") = "UPDATING DURING Lsrch."
It mite be of interest to try to write t. pgm for .30 most simply as poss., then add refinements that exploit my skills. Write original pgm so that it's easy to insert these "Refinements/improvements/changes in the heuristics" 425P
I want to be sure I know how to do .30ff. The not-certain

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

:00 : 561.40 parts are : How search works for IND probs, & how it improves its own GPD (i.e. GPD).

In 91418 : the problem of IND (re QA) is to find $O^n(\cdot|\cdot) \rightarrow$ a certain Gore is Max (CO_2 problem).
It uses GPD to Guide the search : GPD's argument is the $[Q_i A_i]_{i=1}^n$ & the algorithm assigns pc's to O^n conds.
To do induction, TM looks at previous (& present) $[Q_i, A_i]_{i=1}^j$ $(j = 1|n)$ & the O^j's they have been mapped to & their associated Gores. (Data need not be only cands of hypoth — but the cands of hypoth are of much interest).

.06 From that info, GPD gives a PD that gives Probty of a certain Gore as a function of the O^n cond. (somehow cc available is also an input — say for a given cc it gives this Gore probability



.10 Def .11 T. Pgy. problem does have the practical difficulty of being a very large problem dom (i.e. $[Q_i A_i]_1^n$ only if it exists) → .29
I will have to find good ways for TM to deal w. this →.29

Re: T. conclusion about ENV using the cond ENV d.R. to improve (& provide) GPD : This should cause no problem. TM uses the current GPD to control search for the next (updated) GPD.
This is a normal $f_{n+1} = \phi(f_n)$ type of recursion. The "boundary cond" for GPD is the unconditional GPD. The recursion has GPD_0 = uncondl. PD. : $GPD_{n+1} = \phi(GPD_n, \text{soln of Prob}_n)$

On the Q of "Convergence" of the recursion: Say we gave TM no new problems (& therefore became fairly smart), but we continued to update GPD using $GPD_{n+1} = f(GPD_n)$: (This update function would have somewhat fixed CB.) Would it converge to any pgy?
Note that we do not normally do .17 : we do $GPD_{n+1} = f(GPD_n, (\text{Problem})_{n+1}, \text{Soln})$ — We might do .17 for a while, if we had many more than problems.

.20 Hvr., an alternative to .17 would be to do .17 once but w. $CB \rightarrow \infty$.

So we have 2 ways to use time T. ① do $GPD_{n+1} = f(GPD_n)$ once, w. $CB = T$
② do this k times, each w. $CB = \frac{T}{k}$. It would seem that ② would be better for large T & some k value, because each recursion improves the search algorithm somewhat. — But I'm not sure. In general, the $CB = \frac{T}{k}$ would be too small for certain pgys to be observed (Dependency on T, k & the pgy!).

.29 : (.11) Normally we refer to large objects by giving their addresses (or addresses of addresses .....).....
.30 — In present case, the addresses of successive Q_i, A_i will be simply numbered.
.31 On "inserting information" Normally, TM updates GPD using TM's soln to problems. Hvr., sometimes the soln to a problem may take TM too much CC, & we would prefer to insert the soln into GPD. So we might insert these soln. into GPD by having a GPD updated on that soln.
We would want to "factor" soln. as much as possible, so that it is maximally useful for TM — In factoring down the pc for TM is much ↑, but not enough to give a usable CB for current TM.

[redacted]
4/1/02 ID

.08 : 562.40 : ~~[illegible]~~ EXPO: In addition to the Extreme Simplicity of the System; It has the nice feature of Combining Imp. w Meta Imp. : i.e. the cond P.D. guiding improvement of the GPD is the same cond P.D. used to solve "ordinary" problems & is = the P.D. being improved.

I'd Like the System to look like ≈ 558.04 – .26 (in particular 561.20 – .36) solves OPT problems only — thus giving no limits on the functional form of solns to all kinds of problems.

But then I Realized that the "simple" method of 561.30 – .33 did this also: That any function mapping prob descrs into solns, was legalized by the conditional GPD. It (recognizes the problem) (looks at the problem) & it can do anything to solve it!

I think the Critical Q of self improvement limits is whether TM can discover arby OT's & assign them to appropriate OPT probs. [The in the Srch for solns to INV or IND problems, any algm relating problem descr to soln is legal, so OT's of arby form could be devised & used. — so classifying them as OPT probs would seem to add no new features. A possy hvr. that by classifying INV & IND probs as OPT probs, they are pooled w. OPT problems that have OT's that they'd find useful — so they mite "Get better faster".

On the other hand, by using a Common GPD, — This mite give adequate "coupling" betw INV & IND probs on one hand & OPT probs on the other.

It would SEEM that one could Construct a V.G. TM that solvd only INV & IND probs — No OPT probs at all! — then it could find arby search methods as "conds" assigned by the GPD looking at the "problem descr".

→ Is it True that : IND problem solving using GPD must proceed in 2 steps: GPD first gives the pr that w a given CB, a particular cond will give a certain "G" (or pc) for the corpus (562.06). Then we do this integration, that gives the PC that various conds will be "Best of all conds". Is it nec'y to compute the needed Pd in those 2 phases? Is there not a more "direct" way??

ON second thot: Don't we have to do the same thing w INV probs!? — we want to know the pc that a given cond is "best" (see shortest soln cc) (Hvr., see .36 ?!!)

.31 So for all 3 prob types INV, IND, OPT: we seem to have to do the same "2 step" production of the GPD; $GPD_1 \xrightarrow{\text{Integration}} GPD_2$. The only practical way I know to do this integration is by M. Carlo, Lsrch — so it looks like that's how I'd do it! I really don't know how practical it is — the RAM is very inexpensive.

.36 On the other hand, for INV probs, GPD of a cond, could be the proby that it was a soln. — for any cc! Since these GPD's will be w.r.t. data problems solved within the normal/compn bounds of T's work, this may be a not bad scheme. PD to use: It certainly seems easier to do than the more accurate 2 step soln of .20 – .31! — (564.23) (see 554.23 for discussion why this may be not so good!)

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 563.401 Is there a "sloppy" way (fast, cheap) way to do INV problems, corresponding to 8.
"Shortcut" of 563.36 - 40 for INV probs

.02 — I think so: we have a set of pairs $([Q_i, A_i]_i^{\beta}, O^{i})$ $i = 1..n$ as data.

.03 Given a new $[Q_k, A_k]_{\alpha}^{\beta}$ to get a direct P.D. on "O" functions.

Could we do .00 - .03 for orby OZ problems? — (To obtain a PD on OT's).

$\int_0^\infty x^n e^{-x} dx = x!$ works for complex x if $re(x) > -1$. It can be extended to rest of complex plane by $(x-1)! = x!/x$.

T. non-sloppy way to do INV$^\beta$ OPT problems is still in that in theory, we have to consider all possible OT's (say): But since we generate them (t. compact OT's) M. Carlo, we do do that!

I seem to be getting things badly confused/mixed up!

.10 .02 - .03 is wrong!? What we want for a cand, is a function that maps from $[Q_k, A_k]_{\alpha}^{\beta}$ to possible "O" functions (?).

Certainly (?) in t. case of IND, we want an OT — usually something that gets better as cc→∞.
For INV probs, we may also want a function that makes successive approxns to t. soln, or "Hill climbs" using a Gorc that it assigns to t. INV problem.

So: I want to write out various methods of solving t. 3 prob types.

.17 For IND probs: ① uncond Lsrch ② cond Lsrch w. ②a P.D. on cand w. last expected soln time. ②b P.D. on cand viz. 563.36. ③ Use at OT for soln: It requires that this is a INV problem. It may use a Lsrch based on ②a or ②b or assign a Gorc & use a conventional

.20 OT or use GPS or ....

Now could 2a or 2b result in an algm = ③? superficially, it Seems So!
(2a,b) can assign an orby Algm. to a problem!

.23 563.40 → A trouble w. ②b (563.36) when GPD assigns a function algm (cand) to a problem & it eventually solves t. problem, this doesn't mean that that algm is particularly good for that problem, but if many other algms were searched, we know that it's $\frac{cc}{pc}$ for that problem was min among all those searched. However this "min $\frac{cc}{pc}$" depends on t. pc's assigned to other algms.
That this algm "won" is a property not only of t. problem domain, but of t. pc dist of t. competing (cands, algms). ②a (.17) doesn't have this dafiry.

.30

I need to investigate t. diffys assoc. w. various "Sloppy" ways to do Cond. Lsrch.

Re: IND: In uncond Lsrch, one search directly for a function O^n — This rather poor ②. We look for a function that can create O^n from the corpus (in many... this is like an OZ problem).

SN in OZ problem defn, CB is fixed: Could I have a CB that is a mix of Gorc & cc? (like Lcost) well, a mix of pc, Gorc & cc!

In t. Sbary TSQ I "learned" directly for a function that would do Q → A (Good).
I ran into scaling problem. Would this scaling problem go away if I did hunt for an OT instead?

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 564.40: 2 Things that say that "T. System" has to work: (1) If all info is in PD, then system is within factor of 2 of optimum. (Cf. "Improvement of PD" type heurs only — updating during Lsrch is many... but almost all heurs are of the type [redacted] simulated) (2) All heurs can be expressed as OT's or Modifns of OT's. — Very Impt.!

Variations of Lsrch (other than T ← kT; || Lsrch & Random Lsrch): these are all places for T ← 2T

Say X is problem, y is a proposed soln.

.06 Uncond'l Lsrch (1) For INV: (a) look for y in Lex order in Least Limits

(b) Look for pgms ∋ that map [redacted] X to y. Stop when soln is found, discard those outside Least limits: examine cand pgms in Lex order.

.10 (2) For IND: Corpus = $[Q_i : A_i]_1^n$ to find $O^n \ni P(O^n) \cdot \prod O^n(A_i|Q_i)$ is max.

(a) look for O^n in Lex order. Stop trial if $\frac{cc}{P_0} > T$. ($cc > P_0 \cdot T$) ← I think I did this in the SAAR reqs

(b) " " functions (OT's) that map from Corpus to O^n trials, that get resultant Gores of cands in Lex order. Quit trial when cc > p of T (input to OT; output of OT).

(3) For OPT probs. M(x) = is goal fnct; to find x in cc < CB_0 ∋ M(x) is Max.

(a) Try x in Lex order.

.18 (b) Try OT's in Lex order. } ≈ IND..... An OT is a function that has (M(·) & CB_0) as args.

SN: The idea of OPT probs always being OZ probs (w. known CB) does not often match what we want.

.20 — For IND probs, we may be satisfied w. a certain Gore level — in which case the "end criterion" is more like in a INV problem. We may decide: "Gore ≥ G_0 or T ≥ CB_0 — whichever comes first." It may be that after some Experience in a problem area, one will — 567.30

.23 acquire an idea, a measure, an expectation (e.g. in Chess) of how much G we expect to get w. a given cc. If we get much more G for small cc, this is a "Gift from Heaven" which we will tend to accept ☺. Events of that class will lead us to modify our expected G from a given cc. — (See 567.30 on "CHESS")

The OPT srch problem is complicated by the fact that using various (non-optimal) OT's on problems is useful, in that it gives us idea as to how good these OT's are for various other OPT problems. My Guess is that usually it is of more long-run value to know that

.30 a particular OT is Good for a particular OPT problem — Info on badness of a OT for a " " is of occasional value in telling which OPT's it will be Good, but... "BADness"... We may use this "prejudicial info" a certain OT to ... ... low pc ... which is directly useful. Hvr, what this means, is that if we consider the OPT problem as a new problem ...: "Which OT to work on, when & how long", the Gore may not be simply the "max expected G" (or whatever) for a given cc ... i.e. we have to consider the benefit obtained by "lrng" during the trial run.

I could devise srch strategies for either fixed G acquisition or fixed cc.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 565.40 : In line in. 565.06 ff : Condl. Lsrch. simply replace "Lexical order" by pc order (pc from GPD).

.05 Hvr, in each case, we have to define t. GPD, so we know how to Construct it. (.05)

[SN] T. reason [redacted] IND probs are "different" from other OPT probs, is that t. Gen is (partly) that t. pc of t. Cond. be hy; Hvr, this is always a fixed uncondl pc. It is not necly t. same as (& usually diffrnt from) t. Condl. pc = GPD. — LL Cases Somehow

.05: (.01) Going back to 565.06 ff: If we use Condl pc, (1a) [redacted] becomes much better. It is easy to define the GPD: Its data are {prob; soln;}

(1b) may be v.g. It is a dif. over functs that look at t. problem & produce an output that [redacted] is a func of that problem dscrn. If there is a "Best" way to solve all INV problems, 1b should eventually find it. Note that this include "Quick ebort" type heurs (soln improvements).

.10 [SN]: T. present state of TM: I will use t. Condl pd (GPD)", but I'll have to decide between 2 & b for 1, 2, & 3. (I think mostly as) See 22-30 (29-30 in particular!) Then decide on TSQ; but first see how t. system implements various

.12 heurs... how to get them via TSQ's, hints or whatever.

[NB] Uncondl Lsrch in all 6 cases is probly too slow to be used, except for very primitive problems. So we will be mainly using condl Lsrch.

For 1b (condl): How do we update? What is t. corpus?

.18 Woops! 1a is not so bad! Given very many (problem, soln) pairs it should be able to induce soln. from problem rather directly: Hvr. cc is not taken into account in this model! — So it

.20 could be very slow. Well; cc is considered via Lsrch. So t. Q is: Is it particularly good (fast)?

.22 Also, under some (Maybe all!) (maybe many) circumstances, 1a = 1b ; i.e. Int. Updating phase (in update of GPD) {For [redacted] Problem $M(X) = C$ [redacted] to find X :} TM notes that for a class of [redacted] functions (F(·)) of which $M(\cdot)$ is a member, t. problem $\{F(x) = c\}$ has soln. $x = H(M(\cdot), c)$; (This is a compact way of coding many $\{M(\cdot), c; x\}$ triplets.) so (1a condl) uses this soln. — which is t. same sort of thing (1b cond) would lrn.

.28 So for condl pc. 1a & 1b seem to be t. same. [redacted] (Hvr. Note 558.27-31 (?) Correct soln. to INV update) → So proly 1a ≠ 1b

.29 For condl 2 & 3, a & b may also be t. same for similar reasons! If so, then this solves t. decision problem in [.10-.12]

.30 So check [redacted] to see if 3a = 3b. It is concievably diffrnt because OPT is Updated Signiftly difrntly from INV. (3) (1)

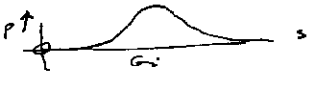
Lets Look at t. updating scheme for OPT: First (3a):

.34 T. problem is : Given $M(\cdot)$, CB to find in cc = CB, $X \ni M(X)$ is max.

.35 For updating we have many 4tuples: $\{ M_i(\cdot), CB_i, X_i, M_i(X_i) \}$ — problem, result ($G_i = G_i$)

Here we are assuming Lsrch (so $M_i(X_i)$ is actually computed for X_i — which is "Cond".) Using Hsrch we might not compute $M_i(X_i)$ e.g. in linear regression, we may find an equivt. way to obtain it much cheaper

Anyway from t. quads of (.35) we are able to induce a df. on all X_i, G_i pairs for t. problem $M(\cdot)$, CB

.40 So for a X_i,  say.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 566.40 | From t curves of 566.40, we can obtain t probly that each X_i would have max G_i (of all X_i). GPD Gives t curves 566.40 and (w. an index on t argt) also the pc that X_i has max G_i.

.02 This latter is used to control Lsrch

In 3b, instead of searching for t "soln", X_i, we look for OT's, t functions that Map t prob. dcrn, M(.), CB, into X_i, G_i. As on 566.34-.40; we do same thing, but Write "OT_i" in stead of X_i : There is a (No!) corresp. betw. OT_i & X_i.

.06 Actually, a given OT_i will for a given problem, often give Many CB^j, X^j, G^j triplets. This gives a much better SSZ for Induction! — But final output, will be a curve like 566.40 ↑p | ⌒ | for each OT_i for t CB specified in problem dcrn.
G_i —

.10 N.B., hvr, in the full testing of X_i in 3a, we could find various CB^j, X^j, C^j triplets

.11 Just as in .06; since 1a will have some (f(.) generator) generate X_i w. problem dcrn M(.) ∈ CB

.12 (It will automatically generate some (perhaps) X_j's for M(.) ∈ CB^j ($CB^j \subset CB$).

In 3a: whatever function, f(.) (.11, .12) generated X_i, could be regarded as OT_i.

So it looks like 3a & 3b are about equiv. (But my mind is still a bit foggy about t Q!)

Well, say t. a & b are all equiv for condl. Lsrch. They are not equiv. for uncondl. Lsrch.

I think I want to use 1b: I'm not sure how impt 2b is over 2a: I did use 2a at SAAB, w. some success; I'm uncertain whether 2b would have been Much Better!

.20 Anyway, for IND & OPT, I will want to have a set of OT's, that I carefully Select, w. initial PD. This to start t System off rapidly!

I have been Talking about PD's (& GPD in particular). It is not clear as to what form of "Data Structures" this will involve, & how practical it all is! What form (in SW, RAM, HDD) does GPD take? alternatives (.23)

.27

.28 Is there some "Ruff and Dirty" way I can simplify a/o Eliminate t.

.29 need of 2 levels of PD in updating IND & OPT? (i.e. from 566.40 (GPD_1) to 557.00-.03 (GPD_2)) → (.33)

.30 566.B SN CHESS: The Program TM can learn from its own past history, make a DF of EC needed to solve a given problem. (This is (exactly) what it obtains/when it updates GPD or any OPT problem.) I think this is a strong step toward solving t CHESS problem.

.33: (.29) Actually only t first level of GPD (or t. OPT/IND update PD) need be stored. T. second level (values are) is created M. Carlo-wise "on the fly" when needed.

.35 SN VERY IMPT. Q: What CB to use in Lsrch for OPT, IND.? → 569.00

Also, t. update for INV Lsrch needs clarity/clearer definition. For INV Lsrch "a" it is clear: t. corresp for update is simply t. {problem dcrn; problem soln} pairs. For INV Lsrch "b" (it seems) that we may need t 2 level Lsrch of OPT probs! If so, this would be VERY

WRITE SLOW and Large!

4/2/02 568

(Maybe not! — mostly) .27-.30 is the TRUE soln. but the INV updating problem

00:567.40 : INTERESTING, ~~Because Conditional INV ... are the "Same"~~ (This will help w. 567.28-.29) For INV Lsrch "b" the corpus for update is (probdem$_i$, H_i([redacted]), soln$_i$, cc$_i$)

H_i(·) is the ~~...~~ func ~~...~~ that ~~...~~ xforms prob dem$_i$ into soln$_i$. (partly unnecy)

if we did induction w.o. the third argt, cc$_i$, it would "sort of work." Updating in 1b(cond) is discussed on 566.18-.28

I need to get 566.18-.28 very clear in my mind, before I proceed. → i.e. (why 1a = 1b.)

Def) .08 In ~~...~~ 1ac (cond) (1aU (uncond)) or 1aU, we ~~...~~ finally got a ~~...~~ set of (probdem$_i$, soln$_i$ = X_i) pairs. (INV)

.09 NO! (To update, we want functions, f_k ∋ P(f_k). [redacted] $\prod_i F_k$(~~...~~ X_i | probdem$_i$ = Q_i) is large!
Fk$ is a problistic function..

10 No! We can ask for functions that map Q_i into X_i exactly [redacted]: there are many such functions — They can be search functions, since the function knows when a X_i it finds is "correct".

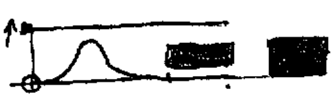
.13 We want f_k ∋ P(f_k) a max and $F_k(Q_i) = X_i$] i=1/n. .... TRUE ~~...~~ but (what about cc?) — .09-.13 seems wrong: See (.27-.31) for what looks much better

We start w. 1aU (no updating): we do Lsrch using this Apriori for Guidance:
We want a "better" apriori, so that Lsrch terminates faster! (we also have to include [redacted] — cc to compute the pc) ← IMPT → [I've been using Elaborate Methods for pc compn — perhaps >> cc than the "cands" being chosen! (i.e. GPD)

or we may start w. 1bU (conventional Lsrch). We find ∃H_i ∋ $H_i(Q_i) = A_i$ | Q_i is same problem, A_i soln.

20 We hunt for ~~...~~ D-algos, like H_i that relate Q_i to A_i so $A_i = H_i(Q_i)$:
In 1bU, we ~~...~~ do Lsrch for H_i's using the univ.d.f. for apriori. Least of soln ~~...~~ is cc to generate & test H_i (pc$_i$) ← this includes cc of compn. of GPD. We find H_i's in ~~...~~ Least order. We'd like to assign hy GPD value to H_i's with low cc to generate & test.
(— & also have ~~...~~ low cc to compute pc$_i$!).

:28 ~~As a first approxn.~~ Our ~~...~~ update data is [∃Q_i, H_i, cc$_i$]^n_i (cc$_i$ = cc to Generate & (Successfully) test H_i) → An Algm that successfully solved Q_i using cc = cc$_i$.
← This (.27-.31) looks like the right way! (new)
Our induction gives for arby Q_j & arby H_j a pd over possl. cc$_i$. ~~...~~

30 Pc  $\int_0^{\infty}$ [redacted] pc d cc [redacted] = α ; if α < 1 the ~~...~~ 1-α is proby of no soln. — So usually α << 1.
(cc →)

.31 Useing M. Carlo method, I then get D.F. [redacted] that an arby H_i will have the shortest cc of soln. → 572.03

My Original idea of GPD was that it would be a "quick and ruff" approx soln. to all problems. That as it matured, it would gete better and better: But usually we would have to "refine" its "soln" to a problem by doing (L) srch. The cc needed to get the GPD for a particular "cand" (= argument) should be small .... but if the distribn is very narrow, we can afford more cc!

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: ~~[illegible]~~ For OPT probs, "Anytime" is the usual (situation/requirement). We can use || Lsrch — which means [567.35 spec] we don't have to commit ourselves to any particular CB. This is o.k. for uncondl. Lsrch — but condl GPD has CB as one of its arguments — so if one did || Lsrch, (or any other) the pc's would change as the total cc on each cand changed.

IN General, the TM I'm devising is probably too "complex" (i.e. too much cc) to actually do. — but I want to get a good detailed dscn of its operations, so I can intelligently begin to approximate.

.07 I think that that "2 layer" updating scheme (of 568.30–.31) may not be the best — & it is a bit expensive (I'm not so sure that Monte Carlo 568.31 is so expensive!) — but any way, given the pc v.s. cc curves of 568.30, it looks much like the WON problem. — It probably

.10 is a WON problem for INV problems; & I may be able to devise a fast way of doing them. I did devise ~~some~~ a fast way, yrs ago, but ~~[illegible]~~ I think it was wrong. — Tho it may be good enough for ~~[illegible]~~ use in the present system! (i.e. Lsrch not the best: see 538.24 ff) ~~[illegible]~~ for ~~[illegible]~~ recent WON stuff: see 537.00–.40 for why Lsrch isn't so hot! (Note 340.24 ... ?)

.14 It might be well soon to look at ≈ WON approach for OPT problems

We don't want to be "too clever" in our heuristic srch. By narrowing the search (Even tho we do this "intelligently"), we are re ducing "Creativity" — the consideration of "far out" ideas. This is a diff't problem: As is, we are doing a very "Elite srch" — which, in GA, is very bad. The reason it is bad in GA, is that the criterion for "Best

20 cands", is very poor. When TM is young, it may have the same problem. Later, Maybe not so serious! ☺ ~~[illegible]~~ ~~[illegible]~~ Re: Elite Srch: The model we are using for the Guiding P.D. will always be imperfect, so by trying "Most likely" (divided by cc) cands, we will lose certain impt [redacted], v.g. cands. This ~~may~~ be an Unavoidable diffy. One way to "smooth out" the specificity of a Stoch Grammar ~~—~~ to substitute each "choice vector" of the Grammar, by a "Whitened Vector" — i.e. one in

30 which all vector ~~[illegible]~~ components move toward the vector having all components the same. We still have the specific "Grammar/structure itself (indep of the vector values) that can limit the possl. class of Things Generated.

We may want to deal (w. this) experimentally!

.34 So I'm returning to an earlier TM model (in a sense) INV & OPT problems. "OPT includes OZ & Anytime". INV are regarded as a ~~particular~~ very impt Class of OPT problems. Hvr, more ~~[illegible]~~ generally (& better): All probs can usefully be regarded as OPT probs.

Hvr, the main idea now, is that using condl. pc for Lsrch & updating that pc often, we can get a v.g. TM (as if we have a suitable TSQ).

4/4/02
ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 569.40 : To do updating of GPD (t condl.pd) shared by all problem types) :

.01 We has to identify t corpus, t sprtpd of these define t. induction problems in OPT problem) and find a suitable OT to (.01).

.03 Selection of a set of OT's for OZ Lsrch is via / t.same GPD shared by all problems

— The (.03) is a Metaproblem.

We will initially include some OT's that we've found to be v.g. for induction.

We may also include some OT's that are v.G. for INV probs: They either do (1) INV Lsrch (t. OT's) (2) (not's) They realize that we have a (discrete) INV problem & find a good fitness function of set of fitness functs (GPS), or any other heuristic tricks for solving INV probs. That t TM has lrnd.

One OT for IND could be simple Lsrch for t proper S-function. (TM could lrn that for certain compara conf, certain named concs (subfunctions) have hy pc. — Also, once usd in a cond, these functs have much less pc of being usd again.

So far, t. system is more or less fixd, except that t. OT's have to be decided on/designed. I have (probly) some default OT's, & I have various ideas on how to speed up discovery of Good induction functions. I want to find a nice way to put them into t system so that it doesn't degrade t. [redacted] evolvable capability of t. system.

So Rite now, t. details of how OPT Lsrch & updating work are not entirely certain. I could just settle on regular Lsrch, but it could well be that something closer to Won (see 569.07-.14) would be much better. For my own mind: keep clear just what options are open in TM's operation, & give some alternative choices.

Note that in normal Condl Lsrch, arguably good OT is always findable: I want to construct TM so that this is most easily facilitated. Also, I want t. other v.g. [illegible] property: that all cond-ordering heurs can be achieved (or emulated) by modifying GPD.

Note t. 2 ways of doing INV. (1) look for a good F(·|·) (of F(A/Q) form) directly (2) try various OT's that try to find a good F(·|·). (t. indirect Method" t. — in which (1) is one possi-"OT").

So write up a dscn of TM as of Now: Dscb all parts that I understand, & indicate what parts are undecided, unknown, or unclear.
If enuf is clear: Draw up a prelimy TM. → 572.00

572.08

ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 : 570.40 : Random Remarks:

1) T. use of "T. Report" as a proposal: It can be quite short yet get impt. pts. across. Emphasis on (a) Deficiencies of other Approachs (including Cyc), & how we overcome them; (b) T. "Super wrapper" approach: (c) Compare w/ Koza's work —

2) Try to get Swiss support / Open Publishing rather than restrictive US Govt. Military involvement.

On Hardware: Organization of RAM, HDD, using Many Machines in ||, is unclear. Originally, I had idea that each machine would be rather indep searcher — but ideally,
.09 GPD maybe updated after every cand trial. Look at details of pgmg. See how
.10 much (OSL) can be realized! We could have 2 sets of CPUs: one set for direct problems, One set for updating. The direct set gives the update set any interesting developements. The update set uses this info to modify t. GPD & possibly sends modifns to t. direct set.
Hvr, this may not be so good, because t. update set is working on update as a direct problem, so it would need a "higher order" update set, etc....

A way to integrate update & direct problems: [cc is assigned randomly (wrt GPD) to cands: These cands include both direct & update trials. Any new update results are fed to t. "assigner".

There is some Q about Lsch in which Guiding pd varies during srch. Ideally, t. update occurs w. current GPD as Goal. If a trial notices that its cand hasn't been worked on enuf (wrt t. current GPD) we may want to spend more than "standard" cc on it. Whether we should

.20

(.21) Spend enuf to bring it to "current expected $\frac{cc}{pc}$" is unclear

(.21) is essentially a new (M. Carlo) update strategy, & should be looked into carefully!

.23 T. way it works: Say T is total time spent on prob solving for this problem. GPD then generates a cand address, along with pc of that address. If the amount of cc spent on that cand is $< kTpc$ (pc is pc assigned by GPD.) then we work on it until $cc \geq kT$. If not, we don't work on that cand. ← (k is ≅ 1)

As GPD Generates M Carlo trials, it also generates t. assoc pc values.

.28 Re: Value of "k": we may slowly "servo" k so that selected cands are worked on, say 50% of times. ← I don't know t. value of this "50%"!

.29 → I don't know just how well this would work: it needs much more thot!

.30 — (.23 – .29) may Not Be Bad! Re: (.28) : "T 50% Q : If no updating is occuring,

---

What does GPD look like? How do we change it?! I imagine that it is mainly a tree of probilistic choices. At each node is a list of choices & a pc for each: t. pc's may be in t. form of Case Counts. To change t. GPD; we can add (or possibly subtract?) to case counts, a/o add new choices to one or more lists of choices.

Since t. lists for choices have elements in common, t. values (case counts) for those elements are stored elsewhere: The lists of choices contain names (& addresses) of case counts.

If Multiplicn. takes much longer than Addition (As I think it should in an economical computer) we may want to use logs, to some extent. Hvr., I shouldn't be worrying about details like that at this pt. Yet.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

(570.40): 1) 3 kinds of problems: 2) To start INV are mainly solved by Lagrching a cpd (≡ GPD). cc

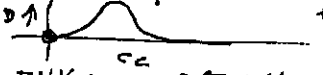
b) The update of t. GPD for INV probs is a INV Problem. See (568.27-.31) for how this is done: but that page could be a lot clearer — Actually 568.27-.31 is probly correct, t. earlier stuff is not correct. .10

.03 568.31 SN Re: 568.29-.31: T. way TM Does INV probs: Look at ibid(.09 -.13). I bet that F_k would always be a d-function, (not an S-funct). Actually, I want to max ln ibid .09; a' F_k could be either D or S. I think ln2(.09-.13) may be essentially wrong But t. discrn. that follows! ibid .27-.31 seems much better.

.07 Thus, t. cases for INV are regarded as OT's: They try to find exact soln to prob. w. min cc. Drop to .03-.07; The my 568.27-.31 does N looks like t. Rite Way!

· T. exposition of 568.27-.31 is quite clear & is correct.

.10 · The output of GPD (It's in t. form): GPD(cc | H, Q): "T. probty that given INV problem Q, H will find soln. w. comp. cost ≤ cc."

.11 

.12 So t. 568.27-.31 expresses t. soln to INV as a OPT problem; The GPD (of .10) is xformed to a PD on OT's (it is essentially an OT).

.14 NB I haven't really Solved any INV problems using t. Target Technique (568.27-.31) It involves a pd on OT's for solving t. INV problem. I feel very uncertain about this soln — The Formally, it looks like t. "Best possl. way to solve t. problem" could be put into this formalism — i.e. as an "OT". But I don't have any "feeling" for this OT motivation INV probs!

.18 SN Also note: t. curve in (.11) can be of >1 in meaning: It can be probty density or integrated probability: See t. "WON" 592.20-.40: $S = r(t)\, e^{-\int_0^t r(s)\,ds}$; $f = \frac{d}{dt}\left(1 - \int s(t)\right) = \frac{s(t)}{\int s(t)}$.

~~$\frac{s(t)}{e^{\int s(t)}}$~~ $f(t) = \frac{dS}{S\,dt}$ say $S = S(t)$ $s = \frac{f'}{f}$ ~~$s = \frac{d}{dt}(\ln f)$~~

$S = Ss$

.23 $Ss = S' = \ln f$ $S = f e^{-f}$ $= \frac{d}{dt} - e^{-f}$ $\Rightarrow \int s = S = -e^{-f}$: $-S = e^{-f}$

.24 So $\ln -S = -f$

Let $f = -\int f$; $S = -\int s$ Then $f = \ln S$

This S & f are nice ways to describe t. OFs of interest.

Dropping INV for t. present (But note .14-.18!): Re: IND probs:

We want a good Soln to 414.18: 2 things to optimize: ① Gore (value of 414.18) ② cc needed.

.30 In t. case of INV probs, t. Gore is always monotonic (non ↓) in time — The ↑ curve has steps: ↑Gore →T

About t. best I've been able to do in defining INP as an OZ problem is to fix t. "T" (≡ cc) at some level, & work toward that CB; until t. CB is reached: If there is cc left, Double (or ↑ in some way) t. previous CB & continue. THE GPD need not be modified, but then optimum pick of OT w. expected Best Gore. Then we do (12-13) or 568.30-31 to get a new PD for Lsmn.

I Guess that one thing that disturbed me about t. OZ problem soln, is that If we assign a pc of pc_0 to OT_0, & out comes CB_0; Then we endup spending only $CB_0 \cdot pc_0$ on it: yet then OT_0 is t. "best" OT for $CB = CB_0$, not $pc_0 \cdot CB_0$, which could be much smaller. A couple of "Hows" to get around this diddy: ① for mature Th, pc_0 would be not far from 1 ② As we change CB_0, the decision variable pc of course

4/5/02
ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 571.40 OT's being "Best" does not change much. → see 574.21–.30

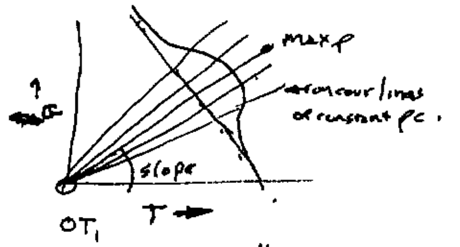
.01 Ideally, what I'd want, is a pd on OT's ∋ for ccc pd; CB_0, T. pc of OT; being best was just pd; CB_0. Unfortly this has many simple solns! Pick any OT_j: let $pc_j = 1$ all other $pc_i = \phi$ if $i \neq j$. This proby has to do w. t. convexity or concavity of t. GORC.

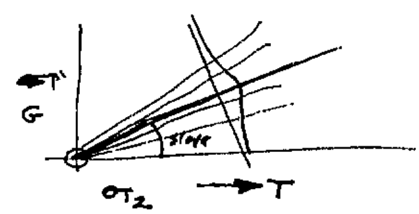
Note 549.08 for Suppose "Simple Soln" to INV. problem see .26

.04 I.e. for certain kinds of Gorc, this kind of soln — (usually unsatisfactory) always exists.

.05 I have worked on this problem within t. last month or 2 (~572.20). Given pc — That only uses a 1 dim. distribn. — Given pc pc as a funct of cc & G (2 dim function) for each OT, to decide on a budget of cc allocation as a funct of time. Maybe instead of "WON" call it t. **Budget Problem**

.10





~~Set of surfaces,~~ for all crosses.

$$\int_0^\infty P(G,T)\,dG = 1 \quad \text{for all } T$$

So for each OT_i curve, we have a slope S_i instead of t. peak and we have $S_i + \sigma_i$ as t slopes of t. standard deviation pts.

Because $\int_0^\infty P(G,T)dG = 1$ for all T

.17 SO: Given 2 or more of such curves each w. its own S_i & σ_i, how to best Budget cc? Time

The $P(G,T)$ surface has a certain monotone quality; increasing T will not ↓ G.

.20 → Perhaps t. orig. diffy is assoc w. t. fact that Lsch is not ideal if one has info on t. form of s or f curves. (572.20 for stated meaning of s & f & WON analysis)

Note: After one works on one of t. curves (maybe f) f  for time T_0 it → f  i.e., it is displaced by T — thus giving a new comparison to other "f" curves.

.26: on 549.06 I got a simple soln. to INV problem! In view of 568.27–.31, it's clear that the → method is not optimum. Hvr. it does take cc into account to some extent, since Lsch (which is biased toward smaller) is used to get corpus for updating.

.29 So this 549 method may be a not-bad Quick & dirty soln. to t. INV problem. → ~~573.08~~ 574.08

.30 {Can I devise a v simple soln. to t. OT problem?}

~~549~~ 572.20 was on WON problems: It may well be that that is t. critical way to do INV & OPT probs. T. Basic Theorem is that 2 "F" curves can be combined (in an optimum way) to make a new f curve. This (theorem) is also (probably) true w. 2 dim. f curves of .04–.17, which means is that t. time needed to combine N-"f" curves is usually ∝ N — (very desirable!).

What I may do is assume a soln. exists for combining f functs, (both 1 & 2 dims) & see if that is adequate for solving all updating problems (INV & OPT (& END)).

I did have that nice "slicing" method of combining f. functions: I think it may have been

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

00: 572.40 wrong, but maybe it was a usable Approxn.

NB In Lsrch, (in OPT probs in Particular) one of the reasons for using an OT on a problem was (in addition to the pussy of my Gobstand) that we would "Exercise" that OT & get Empirical info on its Efficacy, so we could update more Intelligently.

SN In 1998 I discovered MCT (Mpt corpus Thm): which got me about where I am now in realization of a final TM! What was the difficulty at that time? (I was in a rather Manic Mood!)

08: 572.29 SN Another Q: T. "Soln" to the ENU prob. of 54500: If I had solved a given problem before, this "soln" would be able to reconstruct it & go very quickly if the problem was repeated. This quality was not present in the "better" soln. of 568.27-31 (also 572.03 ff).

10 Well: A possible way to fix the "better" soln.: We do have this G-PD (conditional): By OSL, we might regard the (problem→soln) pair solved, as a kind of OT. It's a way of solving $x^2=16$ problem very rapidly (Thru memorization of previous Result).

T. main present problem is updating IND & OPT: INV updating via 568.27-31 seems OK., But it is contingent on IND & its updating.

Some rough ways to update IND:

18 1) in 4.4.18: O^n has only uncondl. opnds Just do Lsrch use for O^n using Past as prior for Guidance

19 2) in 4.4.18: use a "condl pd" for simple O^n Lsrch: T. corpus for this PD.

20 is the set of ($[R_{T_i}, A_i]_1^j$ O^j) $j=1|n-1$ W. best O^j's.

21 SN In hunting for a

OT w. highest G for a given $CB_0 = cc$, we only expend $cc = pc(OT_i) \cdot CB_0$ on that OT. (241) Perhaps look for a OT_i of max expected $\frac{G}{cc}$. (.30) Also may look for a OT_i w. max expected G, w. Given $CB = CB_0$; but we only attempt to use $cc = CB_0 \cdot pc(OT_i)$ on it. This is not a (top caliber) instance. T. OT_i w. hi G for $CB = CB_0$ will hunt to have "Best" G for smaller CB's as well. The CB_0 selected usually is rather arby, anyway.

30: (241) Re: (.241) If we select a CB_0 as optimum for max expected G, we get

31 Max expected $\frac{G}{cc}$ for $cc = CB_0$ trivially. ☺.

Idea [6] may well be that usually, selection as OT, is not critical. We can have a large bevy of them, & usually one or more will be fairly good for any particular problem. e.g. (18) & (19) are OT's that are to be used for IND probs w.m. very young TM.

O.K., so I start w. a good set of OT's, I use the usual MCT methods of updating w.r.t. G-PD w.r.t. OT choice. W.O. introducing any new OT's, TM should be able to go rather far. If new OT's are needed for certain problems, I can insert them "by hand".

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

:00: 574.40! Later, I o/o TM can "factor" t. set of OT's & devise new OT's via a "OT Grammar."

This seems to be an adequate system.

A problem I'd like to work on, is replacement of L-search by Word-typo techniques.

A more immediate problem is analysis of how much time is spent in deciding how much cc to spend on various trials / OTs. (=Time spent on "Administration")

It is not immediately clear as to just how my ideas about cons in induction, & how they are combined, & how subtree discovery is impt, etc) are related to t. formal Model!

.09 .10 Superficially, it's not clear how t. IND learner is going to bootstrap itself!

The GPD improves, but only in selection of a good OT (from t. "fixed Bunch").

Suppose that for IND, we use a special, exchangeable, set of OT's. These are functions that map from t. problem descr. to t. soln — in this case, from t. corpus & prompt to the S-functions, O".

Only if t. set of OT's is "exchangeable" can we have much Bootstrapping.

Perhaps this was what I was thinking of when I said that "T. Grammar" would learn along w. TM." That I'd match how I wanted TM to Lrn, & then incorporate those ideas into t. S system as generally as possible.

.20 The GPD is capable of containing all info in TM. To do so, it/he has to contain t. list of OT's as well as t. pc's of them. — So we would have to have means to Generate new OT's.

.23 Say TM is constructing/causing stochastic functions for IND during trials: How would we Make t. pc of a conc (in such constructions) be "context" dependent? Well, t. pc of each conc. can be as context dependent as we like & this pc can be stored in t. GPD. That t. pc of each conc should be context dependent in any particular way is a way of deriving a stochastic Grammar — (t. rules being t. target S-functions to be used for induction).

.30 Any way I can Gen. .23-.30? If "context" is as General as poss., this may be adequately General.

→ Is .23-.30 t. most General poss. S. Grammar of S. Functs? ←

.36 Essentially what I want to do is adjust t. "Generality" of TM's soln. to t. IND problem. Using various "Narrow" Heurs, means less "generality" (more "strength") & quicker, but I risk "intelligent" solns. Much generality ≡ "weaker" methods — slower, broader search; more "creative".

4/6/02 Hutter.16

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 575.40: 575.23-.40 (& .36-.40 in particular) may be t. soln.

So I (Want to Study / am concerned) w. th. Variety of OPT srches. I can write down Various kinds - Various Classes of them. From this, get a clearer idea of "What's going on" - & what I need.

.03 Some OZ prob. Solns:

(NSN) On expanding t. class of INV, OPT solns: In t. past t. Operator OT would always test (Y/N for INV, G.(.) for OPT) & t. proposed cand. Actually, this need not be so. An op/OT could present cands that it feels are very likely to "pass" or have hy G. If a op/OT proposes cands that, it turns out, are not rigs, that op/OT will get low pc in GPD. For INV probs this is O.K. — but for some

.10 OT's where we want to do "Anytime", it's not so clearly usable. [The FM could still times "MARK" t. Cand that it had tested & thus far — This would solve t. "Anytime" problem]

Examples of ≈ this sort of thing: In analog Curve fitting (real params), we can solve a bunch of linear equs. for best fit (≈ max whatever), & not test for how good it is.

In Hutter's srch, he gets proof that Cond is adequate, but does not directly test t. cond. (.28)

.15 / 16: 522.00ff (597.26) [SN] on H srch: A criticism of "What I Understand to be" Hsrch: Say we do 50% time share between ⓐ trying to Mult specific hdsts (w testing) & ⓑ trying to find a faster general method to do t. Multiplica-, & then applying it to a particular problem.

(↑ = Hutter.)

If a soln. to ⓑ is found first, it will have cc [redacted] < 2 × cc of generating & testing proof / pc of 3

.20 (factor of 2 due to 50% time sharing). This [redacted] will be < icc of gen & test of Mult via ⓐ

— but it can be about as large as ⓐ: say it is done by Lsrch.

Essentially, we replace this particular Lsrch by an "OP" problem: We hope that t. other "OP" will take less time, but usually, t. savings will not be large. The idea of "additive" v.s. "Multiplicative" large Constant is (compatible) (extreme). → 594.21

.27 illusory.

.28: (.15) .16 ff "Hsrch" is an example of one kind of OT/OP. [Note 566.28 only aly to INV probs. 578 (top of pg?) & 578.27-.30 suggest not!]

[SN] on t. "2a v.s. 2b" dicotomy (for IND probs.)

.30 Why 2a^c ≈ 2b^c seems true: 2a (cond. GPD) involves getting a good GPD to pick a good O''(.1.). So 2a,c does in GPD, what 2b,c does in its OT: 2b,c involves finding a good OT that picks a good O''(.1.). Is there any advantage to t. 2 step process in 2b,c. ?

This arg. would seem to work for INV probs. 578.27-.30 is a possl. counterarg. t. That applies (if it is true) to IND & OPT probs as well.

One possl. advantage of (½)bc over ½ac: ½ac [redacted] is designed to give a good problistic choice for O''(.1.). ½bc chooses a OT that can use problistic choice for O''(.1.), but need not be limited in this way to choose O''( ) — Superficially, this/seems silly: Prob choice "covers" Deterministic choice, so its always at least as good. (≥).

4/7/02 ID

577

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: : furthermore, 2 ac can do "Quick abort", if t. Gorc includes t. entire decision process. (? CALC
That would mean that t. GPD computation process could include search techniques that
Used QuickAbort.

(Right margin: Faster Package, CYBERCALC, Spreadsheet, Symbolic Math)

On t. other hand, if GPD uses an elaborate srch pgm to get PC's — would it be possl. for it to give t. conds in ≈ pc order? Or generate them M. Carlo w. pc probys?

.05 Going back to INV probs: There's an INV problem, Xi. If I have a pd. on (data problem X_i, solution fcn S_j, cc) i.e. McElroy

Given an arby X_i, S_j, it gives me a pd on cc needed for S_j to solve X_i. (C_{ij})

It means that for t. current set [X_i, S_j] $_{i=1, j=1}^{n_1\ n_2}$ that I've found a stochastic algm. that

take all of t. X_i, S_j data pairs & assigns hy pc for t. corresponding C_{ij}'s.

.10 i.e. $\prod_{i,j\ \text{in dataset}} P^f(c_{ij} \mid X_i, S_j)$ · aprp P^f = "reasonably good".

This could be by P^f being just part of a

.14 [o: Actually, P^f is linked to other data sets as well.

.15 large Grand PD function that assigns pd's [redacted] to t. entire corpus.

o... Or, aprp P^f can be calculated in a way that gives it linkages to t. corresponding [redacted] functions

[redacted] that were used to solve other problems (≡ difrnt parts of t. corpus). — This

may be a kind of approx. to t. more exact formulation of .14 – .15.

Anyway from t. PD in .05 I use a M. Carlo method to get a d.f. on S_j's — as to t. pd likelihood

.19 that they will give quickest soln to X_i. See 568.30–.31 for method to do this. And from that, LevinSearch X.

.20 — [A difty in this last, is that there are an enormous no. of S_j functions, most of them not much relevant, otherwise .19 would take too long.]

w.r.t. present problem: we have to find way to deselect them;

.22 [redacted] are looking for In 414.18, we want a good O^n(.1.).

.23 A 2ac soln Try to apply Torgg. to IND problem:
Say we have a CBo cc available to find it:
we want a pd on α (≡ value of 414.18) as estimator of

.25 We go to GPD: it gives us a pd. on α (≡ value of 414.18) as estimator of ... It is based on data sets such as this, in t. past (t. corpus of QA's, O^n, cc expended). Using M. Carlo, → (568.30–31) we get a P.D. on which cond will

(Actually, it's ideally based on all of JM's past corpus. This pd is part of GPD.)

Given CBo & t. pd's for various relevant "O^n" conds,
have best α, & we do a Lsrch based on that P.D.
I think .23 ff applys as well to any (OPT OZ) problem (since we're given a CBo).

.30 → Note 578.28

.31 → fivr., .23 ff (= 2ac) is not OZ-Long?! — (which is a 3 bc (≡ 2bc) soln.
.23 is not much like .05 ff

Actually, for IND probs, .05 ff looks like 1 bc not 2ac!

Try 1 bc for t. [redacted] INV problem of .05: $X_i \in M(y_i, k)$

.34 GPD looks at X_i: It couldn't get a pd on t. soln., y_i (i.e. $M(y_i) = k_i$) — well, it might,

.35 — But any proposed y_i can usually be checked more quickly than GPD could generate t. pc of y_i!

I'm not sure .34 – .35 is relevant. T. no. of possl. y_i's is can be enormous: even if "checking" a trial y_i is fast,

it would tend to be slow (?). A possl. way to implement a "Fast" usefully

A possl criticism of .23 ff is that it would tend to be too slow (?). Also, these concs could be cut back

.39 (.23 would be by assigning pc's to t. concs that constitute O^n:

.40 (dependent to an arby degree. (This follows my early thot on IND probs.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 577.40: for OZ probs we could have cores like 577.39-.40 (also depending conditionally) on the nature of the OZ problem.

So its not actually bad.

Woops! I don't see how the pc assignment of 577.39-.40 can implement the pd on α of 577.25!

It doesn't seem to need it!!?.

[SN] 3ac can be regarded as an important "OT" of 3bc!

[SN] Look at sol 86, 89! The system proposed then was very near that I have now! How did that system work? How was it updated? It may be that I felt that when I wrote to TSQ,

.07 that ALP, being a gen. soln. to all induction problems, would always tell me what to do.

While .07 is "true" in the past, it has often taken a lot of work to do particular problems!

.10 Some of which I've not yet solved! E.g. Evaln. of General "Strategies" (for SM or TM).

.11 [SN] on strat evaln. of .10! In SM if strat yield is Y & pc of strat is P, then long term yield is at least P·Y. No! say Y' is the largest yield obtainable from the corpus using long term. Y ≤ Y'. So P & Y alone are not enough to tell whether the yield is A.H. correct.

E.g. in linear regression! (or non-linear regression) One can pick a set of coeffs using standard "best fit" criterion & know how good to expect its future (ms error) to be. While I can do this for "proby" & its "ms error", I don't know how to do it for an arby pay-off function. I had an idea of converting SM payoff functs to probys — but I don't think I figured out how to do it yet!

.17 Going back to 577.23-.31 (= 2ac) for (OZ/OPT/END): 577.39-.40 is a major step toward its being of value.

.20 I guess a Main Idea of the GPD model of TM, is that whatever problem or sub-problem comes up, TM can use GPD to decide what to do — often using Lsrch (if such is needed). That the GPD

.21 Contains all of TM's Accumulated "knowledge", makes this not only "possible" but it makes

.22 it the Best way to operate! — But note (.23)!

.23 [SN] Remark on (.21-.22) This statement is somewhat ("Questionable") "Deficient" in that it contains no reference to cc!

Well GPD Does contain cc info, about how long it will take algms to solve probs, etc. — But (does) can it contain (useful — utilizable) self-knowledge w.r.t. cc? Well, since all of its pc's were obtained by Lsrch, they cannot take too long to implement!

.28: 577.31 · Comment on 577.23-.31: It is very much like the INV soln (using Lsrch 1bc) of 577.05 577 ibid .05; (Soln Method S_j) maps to $O^n(\cdot|\cdot)$ of ibid .23

.30 Just as S_j converts problem defn into soln (exactly) | $O^n(\ )$ converts Q (problem defn) into A: Probabilistically but possibly exactly

Here the Gcrc is cc of soln. | Here, the cc of soln. is constant, but Gcrc is α (=value) of 410.18

If the proby is exact the problem ≡ the INV soln of ibid 577.05

Perhaps I can show 577.23 has the usual (desirable) Lsrch soln properties, w.r.t. optimum soln.

It certainly looks like a legit INV Lsrch! Hvr, it's not clear how 577.39-.40 apply to 577.23!

577.39-.40 don't seem to consider cc & 577.23 does! 577.39-.40 seems to eliminate the finding of O^n's of hy expected α for the given CB_0. In fact, [illegible] a CB_0 is not mentioned here!

So its not clear how 577.(39-40) relates to 577 (23-31)

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00:578.40: **One Way**: Perhaps indep of cc (i.e. indep of CBo) T. methods of 577.39-.40 give good cands. That cc, CBo optimizn. can be regarded as a "refinement", of considering not considering cc — considering k & α only.

.03) → Rammandary: Any hour is a [OT] or improvement of an [OT]! —
Anyway: still t. Q of How to fit 577.39 into an all-over system!
Well, say in trying to mize α for 4A.18; we use general OT's.
One O.T. for that kind of problem is 577.39! Perhaps that would be t. best way to look at various heuristic schemes for INV probs!

.09) — [So any thing that I can think of: I list as an "O.T."! Then later (or sooner!) TM can try to
.10 — extrapolate my initial set of OT's (& associated problem types that they work well with).
[So 577.23 is another "O.T.".
(.09) MAY fix up t. whole system!! It really gets a lot of milage out of (.03)!

→ **So**: Go thru t. whole system: see if this is True: Every problem starts out as an "OT" ("OPT") problem.
Here, OPT problems can vary over what is to be optimized! for Inv. & IND problems, I've found only a few forms to be optzd; but t. general OPT problem is much more general.

.17 ● For [ENV probs] 1) I can start w. direct "Lsrch" for uncond. pd for solns. in some cases.
2) for other ENV, classic Lsrch w. Uncond Guidance PC.
merge — other env: 3) Cndl. pc for Lsrch, say 577.05-.23 — Go over this again: There may also be some easier ways! } 1,2,3 are "OTs".
.20 4) Soln. as General OT problem (Use of old A.I. search heurs: devising a Gorc for "Hill climbing" devising "Differences" (or a Vector Gore) for GPS-like soln
.24 ● For [IND probs]
merge (1) Direct Lsrch for Oⁿ, using uncond pd.
2) " " " " using 577.39. (Cndl. pd): A specific kind of cndl pd
3) " " " " 577.23 (Cndl pd) A more General kind of cndl. pd.
.25 4) Oⁿ as a "OPT" problem using other OT's (1, 2, 3 are OT's): This is most General kind of Soln.

● for [OPT probs]:
1) General OPT (O2) probs: using "good" set of OT's furnished by Trainer.
2) Expansion of OT set as TSQ progresses: TM lrng. Trainer lrng.
3) Attempt by Me & eventually by TM to "factor" set of OT's; double stack discover
.30 — to extrapolate them. [NB] In listing kinds of OPT probs: Consider cc of delay for Soln.

For Expo: Start out w. "empty" OT's & a TM that uses them via a GPD. (= uncond. pd)
All probs are OT probs — TM gradually invents O.T.'s & lrns wts for them, etc.
w. suitable TSQ — this would work, but very slowly.
So, to "jump start" TM, I put in various OT's & TM lrns to use them via its developing GPD.
Later I factor t. set of OT's & help TM make a universal mesa to extrapolate them.

Mention in Expo: That there are many methods in induction involving techniques for discovering regularities in corpus. Some will be lrnd by TM, others perhaps inserted by auto.
{ Note that it's much better for TM to lrn them — even if "HM" are quite broad

4/8/02 ID

found it! [struck through] 580

word.ssd.com/vps5

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 : 579.40 ; Also Mention: Many General Methods of OPT : GA, ANN, RANN.

How TM actually works : One way : A problem comes in ; TM looks at GPD — this gives a PD over relevant "OT's" in 579.17 -.30 Early in TM's life, cond pc's of OTS are biased toward "lo" indexes on. These pc move toward hyper index ness as TM matures. T. exact mechanics of this process is unclear.

No! (Choice of (1) ("TM INV/IND" "Unusual pc") is unnary : If young TM chooses (2) instead, it's a punt. — We can elim (1). No! INV ... diverge. INV ... continuous. IND ... (Merged)

The in some Cases, INV probs can be solved by IND probs w. "D" index; In general, Many INV probs are different from IND probs — e.g. in (4) (577.21)

.09 (.17)

10

T. Difference betw. INV & "D-IND problems. If t. INV problem is "OPEN" one knows certain characteristics of t. target. In IND probs, [redacted] (updating) one knows t. target only, has no certain (other) kinds of characteristics of it. In both cases, one can verify that t. target has been reached.

It may be best to have all 3 or 4 types of OT's available at all times, but in the early stages, t. hyper indices are not much good because GPD hasn't much info in it. As GPD gets better t. hyper indexed OT's get better & get more probability wt.

.17 : (39) Another approach would start w. a system that solves IND probs only; It starts w. D-IND (certain work for many (most) INV probs) then later (soon) S-IND probs. I Guess a machine of this type could be fairly powerful.

(x Minimal Systems)

.20 —

Starting w. it & a few Hours for solving "4 little q's a day", we could introduce various concs viz TSQ — these concs being needed to (discover / implement) specific ... e.g. OT's that we want TM to Have.

.23. (.33)

SN **SERIOUS Q** For devising new OT's & Testing them. Testing is usually Quite Expensive! One couldn't afford to test one unless its pc was rather hi!

So, Alternatively, GOOD New OT's are perhaps Mainly found by "Logical Reasoning".

(Tho it is possible to Solve H.C. probs w. having a very Expensive Score — e.g. Koza's designing of Electronic Ckts using SPICE — [[redacted] slow circuit Simulator Prog.]

30

→ Note also, that this can be speeded up by Various Quick short Methods, Limited Evals, & (Simulating / Modeling) — cheap, approximate Score.

.33 : (25) So we could add t. rest of t. tricks of 579.17 -.30 by slowly adding in testing as t. TSQ progressed — Adding in" either A.H. or (better) by Using Via TSQ. The "improvements / added features" in 579.17 -.30 Can be regarded as Parts of t. TSQ.

.37 Q : How did today differ from t. last QATM that I started writing a REPORT about? What Bug(s) killed that approach? (1) Maybe lack of ability to do OZ/opt probs. Was that all? : T. System had (a) Recognition/discovery of Contexts for conc insertion. (b) Partitioning of Corpus.

4/19/02

ID

JSD33-191:13 [illegible] was little of 12 OT's commonly used.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

580.40: T. features (a) & (b) were initially inserted & I felt I could get it to improve upon them.

I may have felt t. "Partitioning" part was not V.G.

512, 515 is a "review" of (most) ideas up to then; perhaps this will suggest why I dropped old "report".

Looking at 512, 515 — I don't see why old approach was Abandoned! May be need to go back to ~ 414.18

→ 461.00 is a review of QATM: looks like it (original) "Report": 449.00 is another Review

486.00 is another (briefer) review of ideas

Perhaps t. best thing would be to go thru a detailed analysis of both systems, looking for bugs!

Also, look at what I wrote in attempt to "continue" t. last report (a bout 4??.?? continuing containing)

I think one of t. ideas that worked against t. poor QATM, was it seemed diffic (if not impossible) to teach TM, what an OT problem was!

> 447.22 on 3 input UMC "Simulator" via Az141.

447.23 (SN) Reviewing: 447.22 — on how to simulate a UIO machine in AZ141 (≈LISP):

It seems very inefficient! You put in a bit: & then has an output that tells whether it wants another bit. Or if not, it gives final output. So there's no real output, until t. machine decides t. input is in t. "prefix set of choice" — then it gives output. It never gives partial output for partial input! — which a UIO machine would (usually) do.

So, this device does realize t. 2 function idea that I had been considering earlier: One function Recognizes t. members of t. prefix set: t. other generates functions of that prefix set. T. System seems very inefficient as compared to a real UIO machine!

> Later I considered that TM might Learn to put out 0's rapidly: Also we Could just do R input trials in Lexical order, Not adding to input that had Φ or didn't stop.

One General diffy w. 447.22 (& w. R input systems in General): To discard a poor trial s. funct. cand takes many R trials: i.e. we have to do many short R trials before we get a long R trial that works. On t. other hand, for a s-function that fits short R's take less time (usually). T. computation for a short R s-funct can take a lot of time, but .... unavoidable.

I could just put in random trials. If t. permanent ... had no correct output, then pc/is usually quite small .... Still it would take much time to test these "short R" trials.

> 448.31 felt it was inadequate as report: Stopped writing report. 449.01 then has a Review

What I'd like is a system that looks at 414.18, gets a bunch of good O"'s, then is able to find new A's of large $\sigma(A|Q)$ rapidly, for large or by Q. 447.22 will do t. last part, but will not be so good for finding Good O"s from 414.18.

Well! There are t. "Reviews" of ... each giving idea of State of progress at that pt: 449.00 Had idea that discovering good S-funcs (for induction) was something that developed much skill at — i.e. techniques for TM to do this.

I will get

4/11/02 ED

REV 00

582

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

00:58:40: My impression of present state of development of TM:

1) [redacted] It will solve INV, IND & OPT probs. These are in order of solvability: IND can solve INV; OPT can solve IND & INV.

2) Prob. Solving is Guided by GPD.

3) When problem is put to system, it goes to GPD: This looks at which OTs have what d.p. for solns — & it gives probby of each OT being "Best" for that problem

4) After OT works on problem for a certain time (can be sooner in concatenation (quick start)) it stops & updates GPD — which evaluates how well the OT that was used was. That also how the expected utilities of (usually) related OTs were affected.

10 If GPD then gives a choice for next OT to work on this problem — This OT may or may not be same as the first/or to work on other problem.

5) The GPD is always updated by trying to make it better. This is the problem that various OT's can work on. I have several methods (OTs) that I will use to initialize TM — that can (presumably) solve simple problems

Updating GPD / automatically increases / possibly of deriving new OT's (automatically because it involves assigning bigger pc's to OT's that better, had pc's too small to be listed — so GPD has to "invent" these new OTs)

One way to devise new OT's is via a [redacted] of successful OT's.

20 6) Any heuristic can be/usually be expressed as a modification of GPD (w.o. introduction of new OTs).

21 If not it can always be expressed as an OT.

I think the prog. does have capability of working very hard probs. after suitable TSR.

The idea of "judicious" int. rsq. to provide "contextual info" about the problem — "where it came from" — "what kind of problem it is".

Search is an important prob. solving technique in early & adolescent TM, but eventually (see p. 29, p. 39-40) it is replaced by better methods. — This is because WON says so!

What needs to be done before I have a workable System:

30 A good set of OT's w. links to various problem types.

So far, the main OT's: — And perhaps Hsrch — Hsrch

1) Lsrch (for INV, IND, OPT probs (using GPD for Guidance))

2) Use of updates of GPD during a/o before (snapshots).

3) Use of context of subpy sort/size, for placement of cuts in Cand construction.

4) Partitioning curves by "Recognition functions": Tho I have done much on this, it still needs a lot of work.

5) I don't have V.G. heurs (a/o OTs) for discovering Induction functions (E.S. functions).

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

"T" may depend on colors occurring during PST execution

00:582.00 : A simple way to explain QT's: Given a problem, PST is sent to GPD, which (if PST's prob. solving tech requires) calls which QT to use for how long T (589.26). When problem solved (or at end of time T), GPD is updated.

If prob has not been solved, we send it to GPD for assignment again

If it has been solved we loop to ⓐ for new problem.

.06 Updating (may can) also occur during prob solving as a II process.

To Start Discuss various problem types to start out: Some emphasis on IND probs.

After .06 discuss updating GPD — Various kinds of updating.

.10 a) Simple induction: Maxon of 4.14.18. for problems Given to TM.

b) updating to pc of any OT getting a certain score for a certain problem.

c) Devising new OT's. (Actually, if invention of an OT may be suggested by a "problem" given to TM!)

The mechanics of this is not clear to me!)

Discuss initialization of GPD w. universal/uncondl. PD.

.17 I think the General plan is to try to find a way a person would learn to work a reasonable

.19 TSQ — Then find ways that could implement that thing, & try to fit them into TM in as General form as possible, & have TM learn heuristics as much as possible.

.20 The ideas of 582.00 ff are meant to be a framework into which the ideas of .17-.19 can be inserted. I think 492.04-16 is in this direction.

.22 DEF Instead of OT's, I want to generalize to any (PST prob solving tech requires) TM methods involving variety of search are OK to start, but the system should be able to solve any problem in any way — no restrictions on how probs can be solved.

This stages of Growth: ① GPD a uncondl. PD.

② GPD a condl. pd.

③ Development of IND facilities (so GPD can be better)

④ More OPT problems using OT's.

.30 ⑤ OT's become (partially learned) prob solving techniques. 5' use of (PST's for Choosing OT's) for Choosing OT's

Discuss what a GPD is: How it is "secondary" PD. for the "secondary" PD (which is an integral of the first order) which is used to choose promising (functions of OT's).

Possil. use of ANN, RANN, GA for various problem types.

.36 Eventually the GPD will be probably be partly replaced by analogs (not necessarily stochastic), that looks at/under problem, assigns a small amount of cc to it on a chosen "OT" — it watches what occurs, then decides "what/how to do next". Actually .36 could itself be a complete OT! ← In General, the whole system could be replaced effectively replaced itself by a big OT, that just about all of the work, all of the time!

.00: [redacted] 583.40 : 583.36–40 is a very impt idea: that Lsrch can automatically replace itself by [redacted] a superior System.

.01 By "Lsrch" I mean conventional Lsrch (using cond(PD)). The "OT" can be any Prob-Solving Method.

The system of 582.00—584.01 Seems very reasonable! I have a way to start it, some early heuristics, & ideas on how to continue: A picture of the "Teenage" steady state, w. ideas on how it should mature & become arbly good, w. no restrictions on how System can evolve — — — limited only by TSQ [redacted]

Talk about "Scaling" in Part

.08 Can I write a title & Abstract of talk?

u Inductive Inference Machines: [redacted] Goals and Methodology

.10 Discuss shortcomings of Present Systems: GA, (& SVM schemes), ANN, RANN, ect.
[Also See 609.19 for poss. title.]

4.13.02 Noise here → 4:30AM!

Re: 3input line to realize [S: functs]: this was built around the idea of a U10 machine — I don't know how much "U10" implies the prob'ly a TRMC. TRMCs are not good computers! As I

bad features of a TRMC. Try to get an Induction model for a Random access Machine.
LISP (if it has sideeffects?) can perhaps [redacted] have random access storage.
Hvr, I should be able to devise arby S-funct. representations — a TM should be able to use them. I have previously discussed representations of S-functs — there are certainly many.

First Ordinary S. Generative Grammars, w. several pc params (like a S. CFG).
.20 for Analog stuff: input vector → p.d. on vector space: Vector maps to function space. $\vec{X} \to f_{\vec{X}}(\vec{Y})$.
.24 → Perhaps try to find examples of problems & solns of this sort, then genz. solns.
→ A single S. Grammar is a response to a single input type: we want the Grammar structure & params to be functions of the input string/vector. But note .21 Also BBN: see 602.00ff

Q: How does my present Model differ from Sol 86, 89?
On 140.00ff I told about new developments since Sol 89.
IN S 86 I had a GPD that was supposed to contain all problem specific heuristic info: This is a very impt. component of the present System.
IN S 86 the method I describd for "Compressing the GPD" (p. 483): I'm not sure how good this was — tho superficially, it now Seems O.K. || In the present QATM, hvr, its seems a lot clearer.

.30
.31 [SSN] Re: Z141 (& also AZ141): On the "ε" factor/barrier for conc. induction.
.32 Could it be assoc. w. our not considering certain 11 codes? → (620.38 22)

.33 LOOKS GOOD! [SN] On "corpus partitioning": w. CB=∞, one always wants to code the entire Corpus for each predn. For finite CB & large corpus, it is best to code only part of corpus to obtain best predns (& pc ratios).
For a given (say fixd) corpus, & a set of finite QA's about the corpus: Consider the Corpus to be "free", but specifying parts of the corpus, has a pc cost. (This could be very useful in SM!)

.37 Say we have a large Encyclopedia, & a long seq. of QA's about info in it. Say the System sort of understands "English" (the lang. of Encyc). We want a pgm P ∋ $P \cdot (P(\cdot|\cdot)) \cdot \prod_i P(A_i|Q_i)$ is max. $P(\cdot|\cdot)$ will include references to Encyc. — — Ideally, the Encyc will not be computed

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 584.40 Consistent so the same A's will have pcbltys > 0 & < 1. The P(·|·) will have to refer to several statements in the encyc. — & perhaps make logical/statistical inferences from other sets of statements.

So 584.37 may be clear; Then we have an Encyc. (which's "Prior"), we have the {Qi} set, which is free: We want to best code the {Ai} set w.r.t. the Encyclo. [Qi] given a certain cc for the whole {Ai} set or perhaps a certain cc for each [Ai]. The pc of the function that accesses "E" & [Qi], has a certain pc & we want to mini that pc times the $\prod_i P(A_i/Q_i)$ → But see (.21) ↙ So I think the problem is clear — well defined.

So: What does P(·|·) look like? It will have a large section that tells how to find in E, Ai's cands, for a given Qi. Usually no extra coding needed — For Ai: Several probl & Ai values for each Qi will (perhaps) be obtained & TM will prob. (with values) to each of them (if they are all different).

.12 If the Ai's are not so simply related to Qi & E, then the P(·|·) will be less "deterministic".

In this (last) mentioned case P(·|·) must, for each Qi have a way of narrowing down, regions of E that are relevant. (This can be done in a problistic way, so for each Qi, several (wtd.) regions of E are selected). Then < Qi, assoc. part(s) of E > is regarded as a "Macro Question".

.19 — So we want P(·|·) to map that "Macro Qi" into a pd. on Ai's.

.20 More generally (while, for actual cases .12 → 19 is the way it looks) the problem is to maxi:

.21 $\alpha = P_c(P(\cdot|\cdot)) \cdot \prod_i$ [redacted] $P(A_i|Q_i, E)$. ← Note Modifn of (.36)

If the answers is rather "D" from E & Qi, then "R" = null — But if there is ambiguity, "R" will have to supply the "decisional info" to get the Ai.

.24

.25 I think the same thing holds for SM: We have in "E" Many Series of time Series (TS's). To predict one of them, we want a function that selects which TS's to use as data & the proper functional forms & params. We want a funct so that its pc mult by the pc of the predicted TS (w.r.t. it) is max. So we do have pc's for cost of selecting which TS's to use for info. We could have some kind a priori pd. on what those auxiliary TS's are.

.30 At any given time, the P(·|·) has access to all previous pts. in time. → see (.36)

Is there any need to "Genz." [redacted] .21 [redacted] — to apply it to the General Partitioning Problem? So first: what was the "Older" Partitioning problem"? It was Something Like This: we have this large, Growing Corpus. We do predictions in different parts of the "Growing Edge" of the corpus. So we have a seq. of Q's & A's: Each Q,A will be assoc. with a different (partly separated) Corpus. I think .21 solves it, but we replace "E" in (that eq) .21 by Ei since E can be different

.36 for each Q. (Actually, this is what's needed in SM (.25))

4/15/02

ED

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 585.40 : Eq. 585.21 could use a lot of study! What happens to pradns when node is repeated ~~?~~ several times in "E"? Does it necessarily get hyer pc? It would, if this trick did, indeed,
.02 give hyer PC's to "correct" answers (cont Alonzo).

[SN] Well, I think I finally understand The calculated FN ~~of~~ of 56189 : ~~[illegible]~~ I think what it means, is that Pc of a cand. is a ~~[illegible]~~ of (1) T. present problem dom (2) T. traces of every thing The has done up to now. Because of (2) t. system is no longer "Blind". Each trial can be arbly much influenced by t. results of previous trials. It remembers results of all previous trials. [Memory means Not Blind]

"It is not diffl to prove" is indeed true/ [redacted] Hvr t. phrase "Any Conceivable Heur srch algm" can be put into form of a "Blind srch" algm of this sort; Is not exactly true. It does not cover Quicksort + Whether it covers "Look ahead" is not clear. — i.e. cany it do "Experiments" — to gain info — & not be direct trials.

A. T. System in 589 RN4 does appear to be [Maximally Greedy] hvr

.14 ABOUT "Blind Srch": T. [redacted] Set of Algms that look at previous trials can be selected by "Blind Srch", in t. sense that these Algms are given a Apriori for Lsrch & we use that aprid "Blindly" — we try them in ~~[illegible]~~ $\frac{pc}{cc}$ order — How good one of these algms was in a previous trial, is not considered. This would be ok in regular Lsrch (?)

—— Whether it is "regular" Lsrch or not, is not t. Q; One big Q is: Does t. CJS bound hold? Well, ~~the search presupposes~~ each particular soln. of t. problem (known by Tosiner) presupposes a certain partial ordering of the trials — i.e. before t. ~~successful trial~~ trainer-known final was made, a certain subset of other trials must have been made, in order for t. ~~successful~~ final trial have available to it, ~~[illegible]~~ T. proper ~~info~~ (needed) info.

T. foregoing may make || Lsrch a/o "random" Lsrch "impossl." to predict (?)

for || Lsrch, The flow of info betw || processors must be "fast" — T. idea of one processor having available to it, all previous traces of all other processors, is unlikely.

Hvr, t. Timer-Share version of || Lsrch seems to suggest a way to deal w. this.

In t. "random" method of Lsrch, since t. criterion for completely executing a [redacted] trial is $\frac{\sum pc}{\sum cc}$, we can get reasonable estimates of execution times; ~~because~~ This is because probly of total ~~an~~ pc applied to a trial being > k, is not hard to compute, ~~[illegible]~~ & it's also "easy" to compute probly that a certain set of trials all had > needed (total) pc/assigned randomly to them.

Well, each [trial] can ask about t. results of certain (hopefully earlier Previous E) trials. If these trials haven't been ~~[illegible]~~ completed yet this trial will abort. This "asking" mite be possl. in t. "parallel processor" Model. It would seem to be easy to do in t. random || ~~[illegible]~~ model, since each trial has its own known address, & this address can be used to make queries about status of completion of trials.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

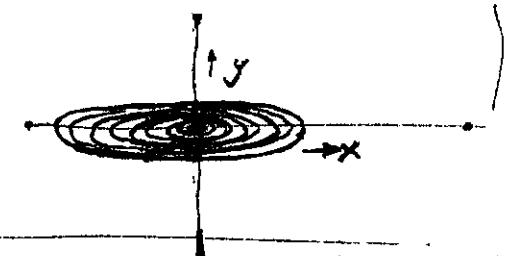
00: 586.40: Just how do we implement the pc of a trial depending on (many) results of previous trials?
Think of how I normally implement this in problem solving.
One simple example: In simple/smooth analog hill climbing: if the trial $\vec{R}$ gets $G = G_0$ and trial $\vec{R}+\vec{\Delta}$ gets $G_3 - 1$ then we want to make trial $(\vec{R}-\vec{\Delta})$. Here we look at 2 previous trials.
This is special case of "look at all previous trial results & trace": from this, induce best next trial!". In this $(\vec{R}, \vec{R}+\vec{\Delta})$ case, the input info and the predictive model is simple. → 588.00

.07
.08 SN In N-Dim analog (smooth) option: I used this local modality of N dim quadratic form. Each step required ~ 2^N evals (expensive!). Consider convergence speed if we randomly select k of the dimensions & make quad form model (only 2^k evals per step).
.10 What is best size for k for fastest expected travel? Consider k = 1 to start.

In the k=1 case, we are moving in the direction of the basis vectors only, so the true shape of the surface can be sharply between them — & very slow convergence.
Selecting a random (still k=1) direction may help a little. To have 3 pts on each direction's vector, so as to know where the peak is.

AH! The random trial vector is usually not best! If I use a Gaussian d.f. in each direction (coord. Basis), I get the same Gaussian d.f. of the component in the right direction! — So the expected Exp. value of ΔG should be
.20 quite reasonable! So N dim. hill climbing should be very fast! Essentially, we want (the first & second partial dirvs) in a random direction: (the second partial is in some direction at first)
.22 Would this work for Protein-Folding? PROTEIN Protein

A simple example is try: $G = -\frac{N}{\sum_{i=1}^{N} a_i^n k_i^n}$. Here one a_i can be very large. Start at random pt. away from $\vec{R}=\vec{0}$. For trial vectors, make $\pm\vec{\Delta}$ [illegible] so that non-linearity is apparent.
So we compute Good value for second partial. In the 2 dim case, We seem to get a constant ↑ in $\ln \frac{G}{G_{peak}}$ per trial. If may have $\vec{R}$ & trials along each trial vector to decide on, & execute the optimum jump.



Perhaps [illegible] general 10 dim. case & try N=2 to start.
look at $\Delta \ln(G-G_{peak})$ for a trial. $\Delta \ln(G-G_{peak})$ per trial or "growth in $\ln G$"
.30 Would it be useful to consider k = 2 (in .10)? Would it converge appreciably faster? For comparative analysis: Compute mean $\Delta \ln(\frac{G}{G_{peak}})$ & compare it w/ value for k=1. Compare $\Delta \ln(G-G_{peak})$ for k=1 vs. k=2 in terms of increment per cc. (Or more usefully, increment per no. of evals of the function (we consider this to be most expensive part: Esp in Protein folding).
It looks "Not Bad", but drop it for the present — other more vital things need be done.
For k=K I think 2^K+1 evals are needed. No! I think it's simpler. Note that the quadratic form matrix is sym. For k=1; 3 evals; for k=2, 5 evals

spec. 588.00 (2)

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: (587.07): I guess a big point here is that while this Ivng during Lsrch is prob'y v.g., it does not do what lots of impl. hours do — It is a pure Gneed pgm. It doesn't do Gyperments: It doesn't do trials to get information.

The other things it doesn't do are "Quick abort" hours.

So while it may be a "Not-Bad" general technique for a Young TM, it certainly can't be the only/final method.

.05

587.40: I've looked at case of $G = \sum a_i^2 x_i^2$ w. all $a_i = 1$ except $a_{20} \gg 1$. Will it work if we have other relations btw. a_i's?

So ① straight Lsrch w. Ivng. btw. problems then
② " " " " btw./during trials (which is what today's discn. is About)
③ Final Hour is solving all problems by OT's & improving OT's. — & noting that all hours are expressable as OT's or improvements of OT's.

.10 ④ A diff't kind of Hour is the "Recognition functions" that put a problem in a class & two specialized prob. solving techniques for problems in that class.

So .00 → .05 says that FN (2/4) of 50189 is not vital; all hours are not expressable in this form.

586.14 → .40 discusses How this "Ivng during Lsrch" might be implemented — & specificity. Whether one could compute a CJS for a known soln. to a problem that was of this sort. Unclear as to whether it can be made to work!

Going Back to 50189 & 8D140.ed #1

A rather simple Integrated System! It solves only OPT problems!

.20 To start, it has a certain (small) set of OT's that I enumerate.

.21 Also, it has ability to recognize INV, IND, OPT & some sub categories of problems (perhaps by my

.22 "indexing" them — So it could "learn" how to do this categorization) & modify the GPD accordingly.

It has an initial GPD (partly Conditional, but not much difference btwn conds — other than the problem category (see .21, .22)).

It uses Lsrch on OT's to start. To start, Most (maybe all) OT's involve Lsrch.

(NB) It may be better to desc. "OT's" as "prob. solving methods." — They are usually (except, perhaps, for INV probs.) Optim. Techniques.

Any (All) of the prob. solving Systems I've desc this far ... all hours, will fit into this formalism. It is a trivial Genz of OZ Lsrch to "Method of Solving a Problem" Lsrch.

30 (SN) In AZ141: One way to store info: Define F_1 to be the thing we want stored. So we could define "4" to be "1+1+1+1". Also define $\sqrt{2}$ (if "$\sqrt{}$" has been defined)

How can we change Memory content? When would we want to?

.33 AH! Re. Inv. probs: The (E.PST's) functions (see 589.26) we use to Map from "Problem desc to Soln." can be any search functions — so they can include Quick Abort! They can be Any things that solve problems.

I think this is also true of OZ Lsrch. But we have to be sure that the set of prob solving algms we are considering does include ability to do things like Quick Abort .... i.e. the prob-solving Methods must include search as a poss'y.

4/16/02 ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 588.40: What 588.33 ~~implies~~ implys is that w/ help of TSQ, t. system will ~~eventually~~ "eventually" get t. best possl. soln. time x "a large constant". T. purpose of t. TSQ is to reduce this "a const".
Since all hours are possl., presumably a TSQ exists that could do as well as an optimally chosen device, to achieve ~~that~~ that cc. Also, for any sub-problem, TM should be able to do ≥ optimum.

Can 588.33 be regarded as part of ~~#~~ 589FN(2/4)? As far as my actual prob in 1989, I doubt it! that?

.06 That FN was, I think ~~to~~ for Lsrch: that a certain broad class of hours could be realized by a ~~#~~ reordering of trials based on ~~#~~ problems been + all ~~#~~ ~~[illegible]~~ previous traces: (Ideally, this includes all of TM's "memory" (what it remembers) — about traces in [redacted] all problems since Birth. This "reordering" could in principal, be obtained in TM by a suitable GPD guiding Lsrch.

What about those Pereg Thms that show Lsrch is sometimes ~~not~~ optimum? (T. GH Thms)

.10 ~~It~~ While it ~~[illegible]~~ seems clear that Quicksort & Lookahead, Experimentation, can't be implemented by that ~~[illegible]~~ "FN" method, it's still pretty V.G. ~~V.G.~~ — I don't see how 588.33 could be ~~made~~ added to it.

By t. way, T. ~~[illegible]~~ Condns of 588.33 also have to be able to ~~Remember~~ all traces since Birth.

As a practical "Asside": This remembering Every ~~trace~~ may not be ~~[illegible]~~ practical — unless t. traces are "cleverly coded", since TM is fast & generates an enormous no. of "trace bits" per second. ~~Man~~ Man, on t. other hand, is very selective — perhaps only remembers << k bits/sec. — ~~for~~ presumably, very carefully selected bits! This limitation was observed for ~~[illegible]~~ storage of external data. Maybe internally generated data can be stored at a higher rate? Also, in humans, is there a difference betw. Short & longer term storage input rates.

.20

.26 (DEF): Problem Solving ~~Technique~~ Candidate ~~=~~ [PST]: When we input a problem, ~~we~~ part of GPD is ~~#~~ devoted to a/distribn. of PST's: For each type of problem, the ~~#~~ type of PST distribn will vary. But in all cases, the ~~[illegible]~~ GPD will be t. pc. that a particular PST will be "Best" for a given problem. Perhaps for all kinds of problems, this Best d.f. will be obtained by integrating an assoc. part of GPD — Like for OZ probs on for INV problems.

.30

So: we give TM problem; it's search GPD to assign ~~#~~ a PST ~~[illegible]~~ and CB. The PST works on it for that CB. Then t. GPD (including t. PST's) is updated. → see [583.00+.06],
→ but instead of "OT", write "PST".

Other than OT → PST, we keep in mind that a PST can involve > 1 "trial" — ~~e.g.~~ e.g. it ~~can~~ can be Lsrch or Hsrch or Srch using a "Quick Abort".

→ An advanced improvement: Instead of Lsrch use WON. The this last still needs work."

(SN) Quick BUG: I have several PST's that are correlated. After working ~~[illegible]~~ on z^{th} for awhile, the $z+1$ th looks better (tho much correlated w. z^{th}). Switching would be ~~[illegible]~~ wasteful, since I have to starting costs of ~~[illegible]~~ PST $z+1$.

4/17/02 ID

590

590

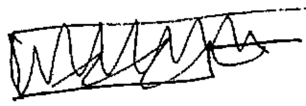
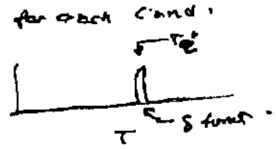
JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

00:589.40 : This problem definitely needs "WON" analysis.

.01 Another Q (589.06R) One of the Gambling House Terms of Sol 86 says that $\frac{pc}{cc}$ order is actually optimum (if you know cc & pc of best or their ratio). How does this confront Lsrch not usually being Optimum? (i.e. in WON; 539.00–.40) — 539.30R in particular!

Well GHT assumes $\frac{pc}{cc}$ is known for each cond.

$\therefore \int_0^\infty f(t)\,dt = pc_i'$

.06  : F.t  $\frac{pc_i}{T_i}$ order.

For other tasks/conds w. f functions of this form, the WON soln is to do trials in

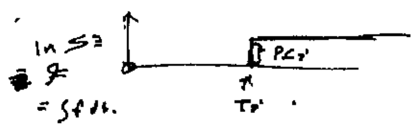
$f(t)$

I guess f is the prob'y per unit time of success at time T.

$S = f e^{-\int f dt}$ → $S_s = e^{-\int f(t) dt}$ → $-S_s = e^{-\int f}$

10 Procedure

$S_2 = -S\,dt$; $f = -\int f dt$ ∴ $f = \ln S$. $S = e^f$



The f curve of .06 may be one of the curves of 539.30 that does work. — Maybe not: Pre (?) GHT is for sequencial search.

539.00–.40 discussion is for ‖ Lsrch.

In General WON analysis is for ‖ or sequencial Lsrch or Mixed.

In GHT, each F(t) has 2 params: T_i & pc_i. The optimum soln is to sequentially do them in $\frac{pc_i}{T_i}$ order.

In 539.30, the F curves are ↗ & optimum soln is to work them in ‖ at 2 (perhaps) different rates.

for each cond — those relative "rates" are constant until the problem is solved.

20 (5?) Another Q in This Area: I've been assuming that one knows the pc's of the conds in advance, but not the cc. In a common form of Lsrch (used in my ~1985 "Opt. Seq. Srch"), One doesn't know the pc of the conds until it has been created! At that pt, one doesn't know cc (yet).

Anyway the (strict?) (true) optimality of $\frac{pc_i}{cc_i}$ ordering Suggests that Lsrch may not be Bad! — Possibly rather Good!

Hvr, WON analysis makes it clear that simply putting trials in order of the pc of each cond being "Best", is prob'y not optimum. How far from optimum is Unknown!

30 In "updating" the "PST" part of GPD, I thought of it as having 2 aspects:

(1) Better PD on existing PST's (with problem demos)! (2) Expansion of set of PST's.

Actually, these are both aspects of the General induction process — which operates so as to find good S-Grammars that are able to Generate (problem$_i$ → PST$_j$) rules/functions.

I had listed S-Grammars as a special trick to Expand the set of PST's; but if we include, in the grammar the problems that go to the problems that they are good at solving, we get the more general update Algm. Very Early in this Analysis, I wanted to put known PST's in "factored form" so TM could most easily devise a grammar.

— But that was ONLY for expansion of the set of PST's. I did view the prob → PST

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 :590.40 probabilistic assignment, as a separate problem. While this elementalization of the updating is a good idea, the true non-del. way Mixes the assignment problem & the PST creation problem into one grand Grammar (as part of Update of GPD)

.03 SO! This idea of putting all info into the GPD seems close to "the Human way": That we should have a large / good set of PST's that we expand & learn to use on new kinds of problems, also seems "like Human".

The use of ALP & Lsrch for updating the GPD may not necessarily be "Human-like", but it's the best I know!

.03 It seems like a "simple System" is probly adequate: The initialization of the system
.10 involves putting in a good set of PST's & associated mappings from problems.
—— AND, of course T S Q design.

.12 Some remarks on fuzzy "All info is in P.D." Not if it uses Encyc or Internet or Library or other External Source of info. → But NOTE (.19)!
(2) The GPD can have its PD's in different forms for different problems: Eg it can be / M.Carlo; f(A|Q), i.e. can be from 3 inputs, of any of various kinds of pd's that are used in statistical analysis. One common form is (corpus, Umc) → P.D. —
(3) GPD is a "Resource Bounded P.D.": for each CB, it can produce a different P.D.
(.19) If we include the "Encyc + Umc" in the GPD, in "literal forms", All info
.20 will be in GPD, but (usually) Most of it in "Undigested form" ☺.
It may be diff't to draw a line betw. "internal" & "external" info. (Like betw. "Self" & "Non-self"). Its a matter of speed of access; Facility to answer Q's using the info.
.23 W.r.T an Encyc: TM will have internally — an index, Maybe key word index,
.24 Subject index, contents, summaries of Chunks of Encyc. — all things to facilitate access to Encyc.

TM would at least do what I do w.r.a. new Info Source that I want to use Much! In addition to .23–.24, I would list areas that I [illegible] input with "problems I am interested in". I try to describe the categories for which the Encyc is (bad / good). I want to do as much as possible to be able to estimate well, whether this Encyc would
.30 be worth looking at in search for soln of Probs I am likely to have (or parts I've already solved — & could solve better.)

.33 When TM is Given a problem, it goes to GPD for soln, or for Guidance in srch — which may or may not be Lsrch). If it looks like the GPD is poor for that problem, it may be because TM hasn't had enough trng. in that area. Another Possy: TM may need more cc for Update in that area. [In general, All updates are cc limited — & they can always be improved by more cc]

4/17/02 ED

REV .17

592

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 591.40 : 591.33 Addresses the fact that when I am given a problem, I often (probably) need to do "Creative work" to solve it.

.01 On the other hand when TM responds to a Q by giving the A or that $O^n(A|Q)$, it's not doing any creative work if the O^n is a simple "Cond" function. When the A obtained here is not of much use, etc., TM then does some creative (Creative) stuff — i.e. it updates GPD (or, more specifically, the O^n "part" of GPD relevant to Q). It may find it best to update more of GPD

Bribery
Burglary
Blackmail
Extortion

than O^n — i.e. suspecting that O^n was too narrow in its focus, that other [redacted] areas of TM's Experience may be relevant. — It may try the Encyc, or Internet ... collateral info. "E" is part of "Q" in .01 So we want Max $O^n(A|Q,E)$. [See discn around 585.21]

.10 [To start, it may be best to have E = null for a long time. TM can learn a lot this mode — Using "E" does require many special techniques that TM will have to lrn. Understanding English could be one of them (tho certain kinds of External info, like (weather, Comp. quotes, Yellow pages, etc.) SM Data, would not require Nat. Lang. NL understanding.)]

Schema ~~Scheme~~ — General PST can include tricks, Q.Abort.

Going back to the simple scheme of 583.00–.06: (except update) OT → PST)

.17 What needs to be explained:

(I) The various types of problems, & ~~what~~ for each kind, what constitutes a soln:

3 main types: INV, IND, OZ (2OPT): ~~Subcategories~~

Subclasses of INV: (a) solved by simple Lsrch on Cond PD. (or long dury trials) — Between trials.

.20 (b) Solved as ≈ OZ problem with Meta Goal : one kind of soln. is to create new non-Meta Goal or GPS (vector Goal). Hmm in Early AI, there were many Methods of solving INV probs;

.23 I'm not sure they were all scalar or Vector Goal.

"usual prediction problems" · Subclasses of IND : (a) Corpus = $[Q_i, A_i]_1^n$ given Q_{n+1} Get pd on A_{n+1}.
(a & b are nearly the same)
(b) Corpus = $[Q_i, E_i, A_i]_1^n$ Given Q_{n+1}, E_{n+1} to predict A_{n+1}. E_i is Encyc, Library, Internet; large data sources.

.26 OZ update (c) Corpus = $[\text{Problem}_i, PST_j, CB_k, Goal_l]_{i,j,k,l}^{a,b,c,d}$ Given Problem_i, $PST_{j \le b}$, CB_k, to get PD on G.

.27 (c_1) · Same corpus but i > b : i.e. PST_b has not been tried empirically.

INV update (d) INV update! Corpus $[\text{prob}_i, PST_j, CB_k]_{i,j,k=1}$ the prob has been solved by PST_j with $CB = CB_k$.

.30 given prob_j, [redacted] a new problem, PST_j (k) and b < b (i.e. a new PST),
to get PD on CB for soln, → see 610.03–.07 for various diffys w. "PST Grammars"

(e) S.Language S. language induction: Given corpus BAG $[X_i]_{i=1}^n$ to get pd. on all poss. X_j : Can be used to extrapolate

.34 → ... Set of PST's. But usually there is More info in corpus (see .20, .27)

Sub classes of OZ problems: (a) Simplest. ~~find~~ Given CB = CB_0, find X: M(X) is as large as poss.

(b) M(X) has random noise in it (c) M is Open or closed to vary-ing degrees. (M explores of b & c).

(d) M(x) time varying (I'm not trying to work this problem: Lsrch seems not to work.

(e) Variety of (d) Maximized M(x) · R(T) T is time to get trial
R(·) is the fcn of T:

593.00 SRec

4/18/02 REV.10 593

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 592.40: BN Looking at Sol 86 § 3.3 (Particularly p483 on details of how TM works)

It looks pretty much like what I'm doing now! I suspect that I didn't understand the details, hmm.

"Compression" consists of finding a code for (D, PS) that is of (approx.) length < Lp + Lps."

Not quite! It is not necy to code the problem descr. They are "free". "We want to ... write a code due to mappings from Probs to Solns" is closer to truth.

In general, I was unaware of many dittys: Various types of scaling dittys, I didn't know soln of MCT or proof of Bay induction method min less QA corpus solution

(margin: It was noted PST's that most likely to find may be is stated CB.)

.06

(margin: .06R: Some time back I was concerned that something Sol 78T3 corollary would give negative values for certain cases: Giving M of infinity in for ε*: Hmm "Gibbs Paradox" may assure it that this cannot occur! The problem also arose in application of that corollary to the proof of the Convergence of ALP for BAG induction.)

.09

Note: the (Cond) argt. must tell just what we want the P.D. of: eg. problem QZ.

.10: 592.40: Kinds of PD's: There are 2 aspects of this: First, Re/arguments for condl. Pd's. Second The practical forms (operational) the PD's take!

For Kinds of args: The IND classes of 592.24–.34 give some of these.

From 592.26 & .27 we can integrate subtly, for a given CB₀, the pc that any particular PST will have to be part of all PST's considered. This dist is used for Lsrch: It is derived from 592.26–.27.

There is a (slightly) different kind arg. mentioned in 592.24–.27 — i.e. ≈ (.31–.34). A p.d. on F()'s giving pc that F() would give quickest soln to problem (M()); s).

.17 On practical (operational) forms: ① M.Carlo: For uncondl. dist. we get outputs w. proby as the pc of that output. Similarly for condl. PD's.

② The P.D. given by ALP using a universal or Non Universal Reference Machine.

.20 This P.D. can be approximated in various ways: One is M.Carlo; Another is to try to find short codes: in either case the result depends much on CB used, & heuristics used to find short codes.

③ A(z) for each z value a proby density of f(z) is given (see 601.16 for useful forms)

F(aᵢ) " " string aᵢ a pc is given.

A sequence of (aᵢ, pᵢ) pairs is given in rough pc order (largest first).

" " " pairs is given in M.Carlo (randomish fashion) the pᵢ's may or may not be given (as in .17).

④ The 3 input unit format condl. P.D. s, Q & R are inputs:

⑤ BBN's: Bayesian Belief Net; 622.13ff.

⑥ ANN's (RANN's) can also be used to approx. the forms. ANN's can have Real a/o Boolean or Integer Inputs. They output ... the outputs are not betw 0 & 1 that represent probabilities! See 604.02.

⑦ [redacted] (622.00) Dot products of factors during Test.

.29

.30 Kinds of PST's Solving INV probs was discussed on 592.20–.23

.31 ③ Soln of INV probs by Lsrch: Given M() & s; to find x ∋ M(x) = s in min cc.

Try functions (in ≈ pc order) F(·) (pc = GPD) that map M(), s into trial x's. General x's & test them using $\frac{cc}{pc} < T$ criterion; then T→2T (or T←2T). The Guiding GPD is the pc that F(·) will do best w. target s (i.e. pc that F(·) is most likely to solve problem earlier than any other F(·) considered.)

.34 ⑥ Simplest soln. to IND probs: ① ... Given new Qᵢ, use GPD to get PDon Ansᵢ update of GPD with IND probs:

See 592.24–.34 for big list of IND problem types.

601.20
607.10

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

-00:593.90: |SN/ on K costs: There are 2 kinds of K cost: (1) for self-del./[limiting] decoders. (2) $K(x, n_x)$: K cost of K given the length of x [n_x]. Q: Is this cost the same as the prefix K cost of x? ($\equiv K^p(x)$)

i.e. length of shortest code for a x' that has x as prefix?

Given length of x, the pc of any code can't be $> 2^{-n_x}$, since there are only 2^{n_x} possible codes.

Li-Vit book may have something on this.

.06 On my method of coding seq's in which the pc of each symbol is calculable: (Arith. coding is one way, but it has some disadvantages (occasionally Bad).)

.08 I use 2^k fixed code words of length k. $k \gg 1$. So this is a self-del. code. I think Arith coding is also a self-del. code.

.10 There is also Huffman Coding using large [fixed size] blocks of original text. Is this "about the same" as my fixed block size for the code? i.e. does it have same diff'ys as text length $\to \infty$?

In the ~~[illegible]~~ Code type of .08, we have to write a short self-del. code for the length of the segments to follow. Since we want to use large segments (for precision), yet we will want the segment size to be arb'ly large; use the code $P_0^{(n)}$ to denote a segment of length $A_k(100+n)$ [Ackermann func.]. For $n=0$ (the usual case) we only need 1 extra bit (0) —.

A segment length $\to \infty$, $n \to \infty$, but very slowly.

.20 I have forgotten just why I needed arb'ly large segments in .08 ← [Well, smaller errors due to finite segment sizes, because larger errors for large and corpus]

.21: 576.27 on Hsrch: One way I think of Hsrch for Matrix Mult: We do this "Look for a proof of optimality" in || w/ [regular] Lsrch. Since verifn. of Mult. normally takes N^3 time, we couldn't ever do it faster than N^3. On the other hand, if Hsrch finds an optimum method in $< N^3$, then it will "win" and mult the matrices in a "constant" plus (optimum time). Normally, hvr, one will not be able to prove

.28 true optimality — only that the time is, say $N^{2.7}$. If criterion for acceptance of the method is "exponent is < 3" then it could be $3 - .001$.!

.30 – (not very useful).

But contrasting Lsrch and Hsrch for Matrix inversion, it's not useful, since Hsrch is not meaningful when verification time ≈ execution time.

Hvr, Matrix Mult is a kind of useful/impt optzn. problem for which Normal Lsrch cannot help.

Lsrch could do this: for CB = CB_0, find (& prove) a method for finding mult. matrices w. minimal α (Time to mult = N^α)

Hvr, Hutter says that is not the problem; the problem is: Given 2 [specific] large matrices to mult them in minimum time. So (I think) he spends a certain

4/18/02

ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 fraction of time on ~~[illegible]~~ theoretically investigating various new methods (& searching for proofs), & a certain fraction of time trying these methods on the actual problems. 10^{50} → 25yrs

task① One way it might work: Start inverting matrices conventional way, using ½ of time share.

Task② for ½ of time share, look for faster than current inversion pgm w. proof.

task③ If better than current pgm is found start implementing it. Using ½ Time share; change tasks 1 & 2 to ¼ T.S.

as soon as ③ is started, restart ② to try to find improvement w. proof / When one is found restart ②

Well actually if ② has found something, ① should stop! — So we only want 2 tasks on Time Share: ① the best way so far ② Attempts to improve the best way so far.

.10 Hur, note diffy of 594.28! We could switch back & forth betw execution & ~~proof~~ (proof improvement) w. each .001 of m exponent!

.12 Well, TM knows how big timestep is, so it can decide if switching is warranted — e.g. for the

.13 known value of N, would the new method beat the one now being tried?

→ This means that the method must be able to actually (w. some precision) estimate CC of soln, not just "order of $N^{2.8}$".

So .12→.13 could work. The statement of just how "Good" this General Prob. Solving

.17 technique is: ~~is vacuous~~ One might say that for a given problem type, there's a size of problems, N(20) (probably) — 599.25–.32 is a ... expected

.18 — Specifically if the method being implemented now has Time T_0 to finish, then, if the new method

.19 will take less than T_0 for complete calcn, we will switch to the new method.

.20 (.17) That is so large that the method will be essentially the "Best method" — it will be solved at a speed that is a small multiple of that of the best poss. method (if that pgm is findable & provable in finite time)

My impression is that .12→.13 would work but it would be useful only if ⓐ N, the problem size ... ⓑ We had rather good means for searching for problem solving method & proofs.

This ⓑ could be true in ~ "OATM".

So ⓑ is usually what People do (perhaps ⓒ) — The idea of using good methods to implement a task, to look for Tricks & proofs is human. Doing this will be implementation of an "organization" of Humans — or not "Human" but could be done by an "organization" of Humans.

.30 Still, I don't know how Hutter did it (on what he actually solved): I do have a hard copy of "The Fastest and Shortest....." (12 pp) (Mar 2001) — There is a shorter version at http://arxiv.org/abs/cs.CC/0102018

I looked at H's "The Fastest & Shortest..." — He has the idea of a "function specifn" — that one has to propose a function & prove it has those specs & prove it has a certain compn time for

.34 any x (as a funct of x): Say one (in L srch) spends all time searching for funct & proof. — Bug! → 599.15

As soon as one is found, we spend ½ time applying funct. & ½ time looking for a better funct.

.. As soon as a better one is found, (as opposed to .18–.19, which is faster, but harder to compute its

.37 total time), one switches to ~~that function~~ computing that function.

If one uses L srch in looking for funcs & proofs, The time needed to find a funct will depend on its cmplx × time to check proof (which Hutter says is ∨ (length of proof)^2).

4/19/02 LSRCH PAPER .07 596

ED

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 595.40 : Since one only spend ½ time on computing t. funct, T. total time needed will be

.01 2 × [(Time to find proof of best way thus far) + (Time to do function)]

If 595.18–.19 is done, it must take less time. ∴ < (.01)

I. forg. is much simpler & faster than H's method, but I proby left out a lot of detail that

.04 may make a lot of difrnce. → Woops 599.15 →

T. way H. talks about proofs, it would seem easy to Mt Carlo generate valid proofs only! See Bottom of on P60f H's "T. Fastest & Shortest". The valid proofs will be like valid functions in AZ141! — So no need to check them. We use t. random proof generation for ...

((Mt. Carlo Lsrch (Much Memory Needed, hvr).

T. search for proofs for many proofs can be much speeded up if we use t. GPD to Guide t. srch (rather than Raw Lsrch) — so it can be practical.

NB A Thm can be short, yet have arbly long proof: e.g. Term: Inputs to "UMC" U, gives "b" as output; Thm derives a', the b can be small, yet t. proof would follow t. action of t. UMC & can be ... Very very long! So: Generating only thms that are true, may be posbl. for only a restricted class of thms. → Hutter "T. Fastest Shortest": Has a Section on Lsrch that is NOT BAD

The Hutter's "The Fastest Shortest" Has a reasonable discussn: — Be noted!

.07 SN Perhaps Write Paper on Lsch: There has been no discussn in Literature of Lsch for OZ probs.

Discuss ... T & KT; || Lsrch; || random Lsrch.) Ito factor of (n!) in || Lsch. Many ... Poorer Approxs of Lsrch (Schmidhuber) (why it works anyway)! If P0 is Pc. of soln, one will hit it x fraction Pc of time. & See ... use Random CB. — But look at his papers on Lsrch.

Also discuss ~70 yrs to do $(b+\sqrt{b^2-4ac})/2a$ $(b \pm \sqrt{b*b-4*a*c})/2/a$. 16 symbols ≈ 10 symbol types

Use $\prod_i \left(\frac{n_i}{\sum_j n_j}\right)^{n_i}$ as approxm. or $\prod \left(\frac{n_i}{\sum n_j}\right)^{n_i}$ to get ≈ pc.

Discuss necessity of conditional [redacted] PC.

Re: Hutter's Paper on Fastest Shortest; Perhaps Most impt idea (for me) is idea of a "Function Specification". ... T. function be specified by its inverse, or any other properties of it. It's like a more General View of this "Function Specifn" idea. Also posbl extensions on kinds of problems we want TM to solve. In particular, t. idea of devising a function & a proof that it does what is needed, should be an impt part of TM's ... repetoire of skills. Hvr, my understanding of "Func. Specfn", is, like now, very A.H.: I think of function inversion, I think of functions given as functions — but we want faster functs that do same thing or ≈ same thing. This losses what Matrix Mult. is about — But somehow H's applying it to specific Matrix Mult problem is outside t. kinds of problems I'd considered before.

I have t. idea that I'm missing some Big Impt idea!

In my "Discovery of ALP" paper, I did use this idea! That in R Bound ALP ...

For Expo: Explain what it Means to use t. "Seed" GPD for all problems — i.e. that t. GPD has as 3 Levels: all into trivial problems, & sub-problems.

Expo: What a PD by compressing data, there is always a tradeoff factor. precision of a pd. & ...

You don't ...

Expo: ... Finding needed ... to compute t. values of ...

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

00:596.40 : The problem was to prove ~~[illegible]~~ lower bnd for the pc of the corpus. There ~~[illegible]~~ One
Ways to display one or many short codes — but there are many other ways. For SM
probs, displaying functions of past data that predict w. small MS error (or other Error criteria) are also
"acceptable" "~~Exact~~ Solns." →

.03 **SN** Why "Cross Validation" is not so good:
(1) In SM, we waste ½ of data! Since we are interested in "Accuracy" this reduces our size a lot!
.07 — (2) Ideally, in "Cross Validation" we can only test one hypoth/strategy. As we ↑ no. of hypoths
tried hypoths, we get into SOY problems. I'd like to be able to propose any no. of hypoths.
— ~~[illegible]~~ While .07 is common to "cross Val" & ALP, I ~~[illegible]~~ think in ALP it's more
.10 continuing.

→ **SN** Look at Jurgen's Q's about S 86 & 89. Just how are they relevant to Present Model?

→ Back to the Hutter & "Function Specific" I could progress fairly well w. a really understanding
it properly! It's just one more ~~[illegible]~~ impt. kind of upgrade to the system. → 598.31

.14

.15 **SN** Re: the convergence to zero of the probability of "U" for a long ~~[illegible]~~ stochastic Binary Seq.
.16 In simpler terms, what it implies is: .165 If we have a long binary seq. X that was Generated
by a stoch func. w. dependence on a few contin. params, then if we have a code for a sequence, X'
that has X as prefix, it is very likely that X' will continue past X or it. It will have
0 or 1 following X. Given a sequence X, whether or not a contin is likely, depends
.20 — only on X & on the reference machine. It is indep of the stochastic source that ~~generated~~
X. But still .165 – .20 is true, since a stoch source of that sort is very likely
to generate an X w. this property.

.24 In theory, it means that ~~[illegible]~~ if p & q are the two probys of 0 & 1 following X,
then we can get lower bounds for p & q. We can also get upper bounds. ~~[illegible]~~ If p' & q' are our
approximations to p & q, then ~~[illegible]~~ (1 − p' − q') ~~[illegible]~~ is the maximum error in p' & q', & this
.27 (1 − p' − q') = P_0 ~~[illegible]~~ will be very small.

Unfortunately, if p, say, itself is very small, the error 1 − p' − q' can be >> p', so the relative
error in p' can be very large. This sort of thing occurs in estimating pc of
.30 unlikely event — like meltdown of nuc. reactor, failure of National defense system (?),
and general estimate of failure of technology (airplanes, space prog, etc).

(.24 – .27) Seems wrong! p' & q' are not known. All we know is N_0 & N_1, the no. of codes for
strings having prefix X that have 0 or 1 (resp) for next symbol. I ~~[illegible]~~ haven't shown a way to
estimate how close this is to ~~[illegible]~~ $N_0 + N_1$ is to the total ~~[illegible]~~ wt. of strings that have X as prefix.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 597.40 : [SN] On Minzn of cc for IN Duction: If one has no aprioriA other than that provided by t. umc, Then raw Lsrch using Lsrch probly about as good as one can do. Only by "improving" t. guiding p.d. can one reduce t. [redacted] cc of t. soln. (If we don't consider things like (Quick Abort).

So, say we use a TSQ: It is not clear how much time one should use updating t. GPD. When we do updating between or during problem soln., it is not clear as to

.08 what fraction of time (cc) available to use in Updating. Also, Updating is usually "selective" — One updates part of GPD. Presumably, in updates during prob solving,

.10 one tries to update "relevant to current problem". The "localized" update is a special case of t. more general induction problem, in which usually for a specific area of induction [redacted] its usually best to use only part of t. corpus for update — (unless CB=∞). So for larger cc available, one might want to update GPD assoc. w. other aspects of corpus (partly in view of their possible relevance

.18 to t. present problem).

I do have t. "50% soln" & its "proof" — but it does not deal w. t. more complex (see it might work in this case) distribution of time over concurrent updating, (

.20 or t. difty of (.08–.18) of "selective" updating of GPD.

Another (Perhaps More Serious?) problem is whether it Might not be better to vary t. amt. of time spent on Updates depending on t. "Current Situation". — On Q is does t. "Proof" apply in this case as well: that for any distribution of time betw. direct problem & Updates, a Machine going at twice present speed, using 50% algm, will always spend at least as much time on to direct prob on update as any possl. division of time on t. slower Machine. (This "of course" is not t. same as saying t. "50% soln" is faster than 2 times as [redacted] fast as any (including optimum) allocation of time.

.30

.31 : 597.14! Hutter Considers "Well defined probs": These are (1) probs in which an acceptance function exists. These are Inv. probs — but generalized so t. acceptance funct may be very costly. (2) The acceptance funct may only be able to tell if one proposed soln. is better than another. If t. funct works for all pairs of solns, a Scalar G funct. exists. — It may take long to evaluate, hvr. expensive

In general, if cc of Evaln. funct is impt., reducing this cc may be a New problem! This Occurs in Koza's use of SPICE for an evaluator — or solving ord. dif eqns by stopping — as Gore for GA: Also, of course in H's Multiplicn. of Matrices.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 59840: The proof of equivalence of (criterion) functs is usually expensive. It can (in Kozo's case) be used statistically, for approxns. For some OPTZN problems — in hillclimbing, an approx. Gove can often be used. In neighborhood of (local) peak more precision is needed, but one may never know one is at or near a (local) peak.

.04 So is that the entire Question?! That when evaln is expensive, Hutter proposes [illegible] inventing equivl. evaln. funct. & proof that its the same. In case of OPT probs, "the same" could be w. known Error.

Hutter also Considers OPT in which G is opaque, expensive. We can define [illegible] (but then cost ALP → ∞) so we have a regular problem as best w. "20 evalns." Hm, the distribution

.10 OZ problem w. 20 evalns is a CB on calcns for ALP. Hm, the distribution ... It reminds me of the CHESS problem! — perhaps

.12 of the cc between trials! "Solve it in a way ⓖ."

.15 : 596.04 ; re: [595.24–.37] : If one finds a sequence of functions that are all very slightly better than the previous one, we would restart many times; perhaps never actually finish! 595.34–.37

The speeds could approach a limit Assymtotically! ⟶ .25–.32 solves this}

The 595.18–.19 does not have this difficulty — its harder to say just how good it is — for large enuf problems, it is very good —

.19 I think one can show that (major stumps?) On the Goodness of 595.18–.19: It picks an p iff at time $2(2^{\ell(p)} + \text{time}_p(x)) = T_1$

20 T. algm being tested — which has run time T_1:

.19 If 595.18–.19 is v.g. for large enuf N, then this may mean that we can simply add a constant [illegible] 2×(execution time of the target Algm).

An algm p will be found (indep of N) at such time T(p).

.25 If system switched to execution to p, it would take time $2(T(p) + \text{time}_p(x)) + T(P(x))$ to Do the total calcn. If we do not switch at that time, it is because/or other algms would do it even faster.

So, using 595.18–.19 the upper bnd on execution time is $2(T(p) + \text{time to calculate } \text{time}_p(x) + T(p(x)))$ [time to find the proof; time p(x); execution time for x; T(p(x))]

.30 GOOD!! 2·(Time to discover and check p's proof & assoc $\text{time}_p(x)$ function + time to calculate needed execution time of p(x)

.32 + Execution time needed for p(x)) = upper bnd on time to do task.

So .04 may be not needed "complete search" of Lsrch for all well defined problems

In the case of G(x) ●OZ: G_ov = G(x)·[illegible]T( ) we could try a "theoretical analysis" ("proof") type of soln.

Hutter mentions Mat Mults Dynamic Prog, and decision probs (SAT) as amenable to Lsrch.

Q: Can a proof reduce cc in ordinary Lsrch Problem?: Say we want x

.39 ∋ $\sin^2 x + \cos^2 x - 1 + x = 2$; Given a.b find x. We can prove x=2 is the same problem. f(x)=2

4/22/02
Evergood for Bread!

6:00

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: $599.40: [illegible] could we save cc by (knowing, recognizing, Hunting for this identity & proving it)?
Remember, Lsrch for inv. probs looks for a function that maps f(·) and S into x, ∋ f(x) = S

[~SN] Note that (f: funct of 599.39L) being equivt to x would make it very likely that we would use a funct that recognized that "f(x)=x to find x ∋ f(x)=z" had a simple soln.

.07 [SN] In ordinary, || Lsrch (Time Share), If we rent hardware for cost per second of use, cc = Time × Memory: For || Lsrch: Time = $\frac{Time_{soln}}{pc_{soln}}$; Memory hard to estimate.

.10 If we consider trials in pc order, there are $\leq \frac{1}{pc_0}$ trials w. $pc \geq pc_0$.

~~[illegible]~~ In worst case we spend same time on all trials, In worst case storage (Memory) needed for trial = time for that trial. In " " this is $cc_{soln} = Time_{soln}$. so $cc \propto (Time_{soln})^2$ and there are $\leq \frac{1}{pc_{soln}}$ such trials! so total cc $\leq$ $\frac{(time_{soln})^2}{pc_{soln}} = Lcost \cdot (time\ of\ soln)$ in steps or "clocks".

.15 So || Lsrch has extra cc of mult. by — NO! see .31

Woops! No! Total $ needed = no. of trials ($\frac{1}{pc_{soln}}$) × $ of each trial ($Time\ of_{soln}$) × time for entire process ($\frac{T_{soln}}{pc_{soln}}$) which = $(Lcost)^2$, which is enormous.

20

.21 Hvr using present day prices, maybe cc is not much > Lcost!! 1 GBy of Ram cost ≈ cost of fast CPU: If necy, we can use 2 hard discs

.23 (or even just 1) for swapping.

I got this bound of "[illegible] × (d−1)" for how much faster || srch was than T ← 2T. (was there a "2" factor here?)! If D can be 10 (~ length of string describing soln) then it's a serious factor.

Hvr. the main reason I was interested in || t.s. Lsrch was t. rand version since that it facilitated the comput of t. integral needed to compute

.30 the second order GPD Probs and to control most (if not all) Lsrch.

.31: (.15) (.35R) maybe wrong!

$\left(\frac{Time_{soln}}{pc_{soln}}\right)$ (= Time to solve soln.) $Time_{soln}$ (= storage for soln.) } this is cc of one trial:

.35 Mult by $\frac{1}{pc_{soln}}$ to get cc for all trials: so $\left(\frac{Time_{soln}}{pc_{soln}} \cdot \frac{time_{soln}}{pc_{soln}}\right) = \boxed{(Lcost)^2}$!!

From a practical standpt, (.35R) may be way off! Note .21–.23 — also 602.11 & 602.05
Note that (.35R) is absolutely worst case scenario! All pc are same.

4/23/02

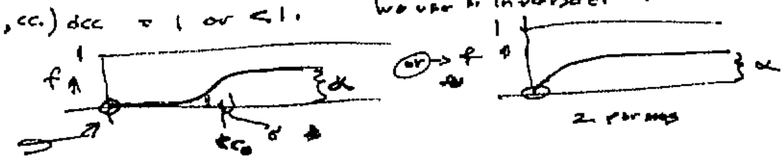
JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

[redacted]00:600.40: The main attraction of the ll search is possibly a (d+1) speed up for soln's of "depth" d. But most impt: I'd like a Monte Carlo output to use for Dnto grammars — GPD guiding d. A for lsrch.

Typically, what I will want: Say for INV! As corpus, a bunch of PST's applied to INV probs: & their soln times (sometimes lower bnds on soln times — since those PST's didn't solve the problem. From this {Inv_i, PST_j, cc_{ik}} data, we get a PD on (can also be func of) (Inv_k, PST_m, ...)

The PD's that we get from t. IND problem soln. can be in any form: — I could perhaps arrange for it to be Mtcarlo form.

I had thot of a soln to t. IND problem using 3 input vars, — this would easily enable Mtcarlo output — by putting random "R" into it. In t. case of (Inv_k, PST_m) we (another, Apparently Very Useful form of IND output: normed) want a continuous function of cc as output P.D. We can do this by specifying a distribn of 1, 2 or 3 params: It may be more useful to specify t. integrated pd, so $\int_0^\infty f(Inv_k, PST_m, cc_i)\,dcc = 1$ or < 1. We use t. inverse of $f(\cdot,\cdot,\cdot)$ to get t. desired Mt.c. dif.



This is a nice 3 param distribn.

We can adjust t. 3 params so that t. [illegible] 414.18 of t. corpus is Max.

Maybe back to 593.40 ! A review, outline of "T. Report".
Reread early stuff on the report to get into t. spirit" !
I think one Impt Idea was that I started out w. a few Good PST's, & they were good enuf to lrn a fairly useful instructive TSQ. — & that I simply added on new TSQ's either by Normal lrng, lrng w. strong hints, wiring in, ...

[SN] for Hutter's "The Fastest, Best": An actual proof may be onerous! A few well chosen examples or random examples may be good enuf to make a TM. Very likely — Good enuf for a practical TM!
Sometimes t. function given by TM will actually be wrong — but it may be possl. to Make t. system
⇒ This is very unlikely ("Very" is adjustable — too extreme "Very" is very expensive! — So there may be an optimum to chose)

[SN] Re: T. GPD (Grand or General PD): It contains most of TM's "SMARTS": it runs t. show: It is a Recursive Bounded Rec. Bdd. Pd (Hvr. Note 60631) but we give it an input: It has (usually) a quick response, but if we wait, it will give corrections, & (presumably) progressively better outputs.
The mode in which this is done, will depend on t. [illegible] given to GPD.

When a problem is first given to TM, it goes to GPD, which assigns t. PST. (This done probablstcly — usually, it gives t. "Best" Guess — but not necly, if a M.Carlo mode is being used for that GPD Argument). After t. PST assignment, t. PST works on t. problem until it "solves" it (if this is meaningful) or until it gets an "interrupt" from GPD that is simultaneously working along w. that PST & watching its progress.

4/23/02 ID

BBN .13 Boolean Belief Nets.

6.02

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 601.40: while GPD is Doing this, it updates itself (also in ||, simultly) & at a certain point, it may become clear that the PST working on the current problem is no longer the "best bet", which is when it issues an "interrupt" & the new "Best Bet" PST

.04 starts to work on the problem (or continues where it left off). → 607.10

600.02 – 601.10

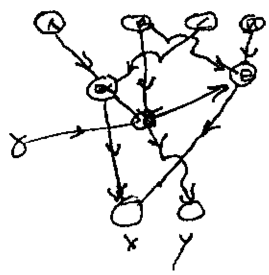
.05 SN HW: For || Lsrch (which may or may not be feasible; see ...)

I had envisaged dividing RAM into 2 parts. One is used for || Lsrch while the other is being "swapped" wrt the HDD. This arrangement may be unnecy w. new kinds of RAM, that are magnetic so they don't have to be refreshed. Hvr, the cheapness of Disc Memory is

.10 a rapidly moving target, & it may not be overtaken in the next 10 or 15 yrs!

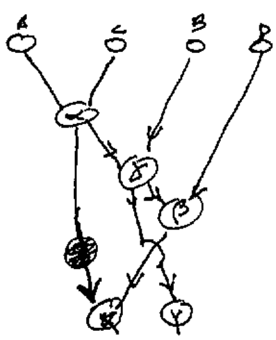
.11 If swaps betw RAM & HDD are used, the cc of 600.35 is certainly completely wrong!

.13 SN On Bayesian Belief Networks (BBN's) (J. Pearl).



ABCD are input nodes: X, Y output nodes.
If inputs & outputs only have 2 values & "hidden" nodes can only have 2 values and probty of the state of each node depends only on states of its inputs, then α and X (2 in, 1 out) can be characterized

.20 by $2^2 = 4$ reals, — one for each input config.
for 3 inputs (α, β) 8 input configs ∴ 8 reals.
But α has 2 outputs so it has $2 \times 2^2 = 8$ params.
Y has only 1 input so 2^2 params.



α $2 \times 2^2 = 8$
γ $2 \times 2^2 = 8$
β 2
X $2^2 = 4$
Y = 2

So 24 params to give d.f. of X, Y as a funct of A, B, C, D.

In general ABCD has $2^4 = 16$ states; for each of these we have a proby that X = φ & 16 probys that Y = 0

∴ 32 params.

.30 We can do induction by using a corpus of I, O pairs; then find a network & set of params that gives max pc for the corpus.

.35 For a brief (too notessy to understand) summary of BBN's see Evol comp. Ap 2002, p 116

This paper also discusses cc of various ways to approximate a good set of params.

T. general problem is said to be "NP Hard", in "Worst Case".

So this is a compendium of ways to make induction models. — I have no idea as to how good they are.

4·24·02

ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

00: 602.40: the Goal of a fit is to ~~[illegible]~~ Maxz:

pc of discrete structure of net; dcrn of what nodes → what nodes.

mult by Σ ~~[illegible]~~ h volume in continuous param space of the "peak".

Actually, we want to ∫ over all params of the pc of the data, but usually we can get a good approxn

by integrating k exp−(quad. form in the params). k is value at peak.

It might be possible to get this via a [Mt. Carlo integration] — even for a large no. of params.

It ~~[illegible]~~ looks identical to problem for feed forward neural nets, using a (relatively) small no. of wts.

I ran into the Evaln of this multidim. ∫ some time ago. — again it was optimization of a

bunch of continuous params: I don't think I considered Mtecarlo Integration, tho.

The nice thing about it is that we get a mean & var. for the current result, so we can stop

when we have enuf precision.

Re BBN models: we can write an ~~analytic~~ algebraic express for each

output pc as a function of the inputs. This is most easily done by expressing each

node's ~~[illegible]~~ pd. as a fn of its input pd's.

$$P_\alpha(0) = P_\alpha(0|A,C) = a_\alpha P_A(0) + b_\alpha P_A(1) + c_\alpha P_C(0) + d_\alpha P_C(1)$$

$$a_\alpha P_{A,C}$$

The idea is that BBN is a reasonable Eln of a General Probabilistic function ——

The "Elements" (Nodes) usually have individual meaning, so we can estimate/approx. them.

~~[illegible]~~ BBN does seem very similar to feed forward ANN. We have a set of continuous

params that we want to opt. In ANN we approximate D-functions. —

In BBN we approximate S-functs — But I'm uncertain about just what kinds of S-functs

are reasonable "Targets" for approxn. I think initially, the idea was to represent

human "Causal Reasoning". One way to Elz this: say we have this module w.

5 binary inputs: It's supposed to recognize a certain type of object — we call them "α's"

(i.e. α or not α?)

we have a bunch of ~~[illegible]~~ I/O pairs w. Objects 0/1

we then want to put a small BBN in the detector w. a few params &

optimize these params to give max pc to the I/O answers.

We can make β detectors, γ detectors, etc. Each is a small BBN that's

been suitably trained. We can then combine these nets in a deterministic way,

if we have deterministic D-functions on α, β, γ that can be used to detect, say ε..

In BBN's as opposed to other S-funcs; the inputs are usually strings on

discrete/binary/Boolean. Output are pd's on 0/1 or on ~~[illegible]~~ a

set of binary or Mary symbols. These ~~[illegible]~~ output pds are completely dscrbd

by a set of reals (usually nos. betw 0&1 - pc's).

So we have these models on discrete inputs & S-outputs on strings or discrete variables

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: ~~00~~ .40: We can then simply combine those modules to make ~~[redacted]~~ larger modules w. same I/O characteristics.

.02 Actually, ANNs can be used in a similar way! We can put Logical or strings in inputs to ANNs to get real nos out that are betw. 0 & 1 — which can represent the pc of a particular binary bit. (Boolean). ANN's can also be used to go from ~~[redacted]~~ D. Real nos. to t. same real outputs that ~~[redacted]~~ represent PC's. We can train them just like we did from 4.14.08.

· When a module has ≥ 1 Boolean output, (say 2 outputs) we have to express t. outputs as 2^2 nos, because of correlns betw. t. outputs.

10 It would seem that t. way BBN's & ANN's work is similar enuf so that I ought to be able to mix them into something that is either one (as special cases).

If we have 2 BBN modules operating on t. same input string! Each has a single Boolian output. looking at the ~~[redacted]~~ real nos (pc's) — one couldn't tell that they were (probably) correlated. Hvr, if ~~we~~ we put random ~~[redacted]~~ (common) input to t 2 modules we ~~[redacted]~~ could notice that t. outputs were correlated. T. correlations betw. outputs would depend much on t. probabilistic ensemble ~~[redacted]~~ one used (Mt Carlo-wise) on t. inputs. More exactly, if P_1 & P_2 were t. output pc's (reals) of t. 2 Modules, then w. ~~[redacted]~~ t. same stochastic input to both ~~[redacted]~~ modules t. 2 nos. P_1 & P_2 would be ~~[redacted]~~

20 — (usually) "correlated" Say we have a binary k vector as input. 2^k different possys. If we assume equal prob'ty for each of t.

.21 ~~for~~ each of t. 2^k cases we have a P_0, P_1 pair.

.22 2^k configns, we can calculate t. correln. betw. P_0 & P_1.

If we ran those 2 outputs as input to another [redacted] BBN module, — how would it treat the [redacted] "(P_1, P_2) 2^k - vectors" of .21 - .22?

I'd like a nice way to combine modules; so given t. "characteristics" of several, I could get t. "characteristics" of an combination of them. ~~[redacted]~~

Some specific problems: ① cascading ② 2 ∥ll feeding into a third.

30 ~~Clearly~~ The 2^k I.O. vector ~~[redacted]~~ does seem to characterize a module.

.31 Also 2 ~~[redacted]~~ Modules in ∥ that ~~[redacted]~~ share some but not all, input bits. How do we characterize ~~[redacted]~~ the I/O of t. pair of modules, if we have t. k_1, k_2 ~~[redacted]~~ I/O vectors of t. 2

.32 uncombined modules? [I.e.] This is related to Relational data bases? Overlap of inputs into correspond to t. "join" operation.

I'm not sure of DataBase terminology: Is a Column a "Field" & a row (≡ line) ~~[redacted]~~ in ~~[redacted]~~ "Entry"? Each row corresponds to a data chunk. The cols tell which kinds of data ~~[redacted]~~ t. elements of t. row are. ~~Certain rows, columns~~ Certain columns are input & other columns are "output" (the ~~[redacted]~~ assignment seems Arbitrary!

.40 Anyway, if we have a theoretical I/O function ~~[redacted]~~ we can simulate of

ED

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 : 604.40 ; by an file w. infinitely many rows, and a proper Monte Carlo distribution of I/O cases that describes the

.01 desired I/O function exactly.

If one has 2 such functions we can cascade them by "joining" the output cols of one to the input cols of another.

If we have 2 stochastic functions (like 603.31) with a few, but not all, inputs in common, we create the 2 functions indep'ly by 604.40 - 605.01, then we overlap their common inputs via the "join" operation. The outputs of the resultant module are then properly related to their inputs. 92.0 2004 64.

.10 On the Model construction problem (414.18 ;) : Say one has an empirical file f_1 w. Q_i, A_i pairs. One has a "theoretic" file F_2 w. Q_i', A_i' : If we join these 2 files on both Q & A columns, then if F_2 is a good model for f_1, the case count of the resultant file will be big. ~~poss!~~

In .10 consider : say only one Q column: Q_0. The data is big so several duplicates of each entry line. When we join 2 files on Q_0 & A cols, we get a ~~dot~~ the case counts of the A column. Say f_1 has n_{11} cases of A_1 & f_2 has n_{21} cases of A_1, then the case count of A_1 in the join of f_1, f_2 is $n_{11} \cdot n_{21}$.

In general, say one has many theoretical models for f_1, these are F_i, $i = 1, \ldots$

If they all have a large fixed case count for Q_0, but n_{ij} is the case count for

.20 F_i w. $A = A_j$, then the best model is the one for which $\sum_j n_{1j} n_{ij}$ is max.

This is a max of the dot product & occurs when $n_{ij} = K \cdot n_{1j}$ — i.e. the vectors are pointing in same direction.

What would be significance of the matrix $M_{\alpha\beta} = \sum_j n_{\alpha j} n_{\beta j}$?

The matrix given by $n_{\alpha j}$ is the case count of A_j for Model α.

Well. .20R is just a dot product: No need to bother w. Mat. Mult. (But poss!).

---

2 ways to tell if 2 pc vectors are close: Vectors $p_1 \cdots p_n$, $q_1 \cdots q_n$ ≅ 414.18

.28 In .20 it's $\sum p_i q_i$ is max; in Info theory it's $\sum p_i \ln p_i'$ is max (or $\prod_i p_i^{q_i}$).

– Say $\sum q_i = 1$, $q_i \le 1$; p_i are positive nos., $\sum p_i$ can be > 1. p_i can be case counts.

.30 – In any case, the $\vec{p_i}$ is the "model" & we want to pick a $\vec{q}$ → .28L or .28R is max.

I have been using **.28R**. This is ≅ (≅) 414.18. The advantage of .28R is that we know how much to penalize the selected model for the pc of the model itself.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 605.40: From the Appendix of 602.25 on BBN's [illegible] I understand them: There is an error in their computation of the no. of params needed to specify a probabilistic state of nature. After my correction, I think I understand how they are meant to work.

Each module node has a discrete set of params that tells how it expects Any input coming into a pc. on its output. — usually a discrete set.

One way to learn in these nodes: Like in AZ141, we have nets & subnets. In BBN (& in AZ141) we may use the same subnet as a component in various different runnings. Suppose we find a certain subnet has been useful up to now, in several problems, w. a certain set of params designated. Then, as we use that subnet in more new problems, we "trim" its params so it fits the larger corpus better. (The Gorc for this "trim" is that the pc of the corpus be Max.)

As the corpus grows to useful subnets (modules) can be used as trial components of larger nets. They themselves keep their params fixed, but the params of other nodes can be adjusted to give best pc to the solns of some new problems.

The heuristics for constructing new trial nets from old useful old subnets needs to be discovered. But: Other workers on BBN have hopes for forming BBN's & optimizing params.

A possible example of the form! We have a module whose input is a 2 dim raster. Its output is the pc that it "saw" the letter "A". Modules of this sort can be combined to recognize words, sentences, etc.

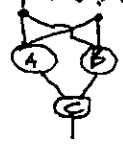
We can use simple nodes of completely adjustable params, to connect these useful "Modules" into larger, more useful, Modules.

I should actually write out the boolean functions small enough to see how the params entering the final output pcs → 607.00

.31 : 601.31 SN I've been saying that all of TM's SMARTS is always all in the GPD. Well, for (EXPO explosions) I think like much info is in the form of PSTs. I view these as part of the GPD — since when it is updated new PST's can be created & "added" to the GPD (or rather to its "Bevy of PST's")

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

:00 : Spec 606.30: Each node · is determined completely by ti no of inputs & their radix à ti no of outputs & their radix & ti assignment à ti radix of output (only 1 output is assumed) So ni input, wi ri radix $r_i^{n_i}$ input combns; for each there are (r_2-1) params that detb its output #. so $r_i^{n_i}(r_2-1)$ params for that node.

I suspect that consider  ti outputs of A & B are usually correlated, But computing C's output by taking that correln into account, may be complicated, analytically. I suspect that they approximate by acting as if ti outputs of A & B were indep.

:10 : {601.24 602.04 593.40} : 601.35 – 602.04 is a reasonable dcrn of how GPD (à, essentially, how TM) operates: 593.40 lists # kinds of pds : kinds of syste of PDS (note 593.05)
operations forms of PD's, kinds of PST's.
It would seem that this would be an essential dcrn of TM!
One needs in addition # more PST's: Various ways INV probs are solvd.
Also ti initialization — How TM starts out.
Then, of course, ti TSQ. My feeling is that ti TSQ wouldn't be as diffl to write, once I have a very clear picture as to how TM works, — and 601.35–602.04 + 593.10 ff seems like a pretty good outline of how it works.

$\int_0^\infty e^{-x}$.
$d\,x^n e^{-x}$.
$n x^{n-1} e^{-x}$
$- x^n e^{-x}$
$\int_0^\infty = 0 =$
$n f(n-1) = f(n)$
$f(n) = n f(n-1)$
$\therefore f(n) = n!$
$\int_0^\infty x^n e^{-x} = n!$
$\int_0^\infty g(x^2) =$
$\int_0^\infty g(x) e^{\pm x^2}$

:20 A possl course: To write up some detail on how ti early TM works: Then list some advanced problems, advanced techniques.
:22 Start w. d-END problems. — ANL #13 (possibly) a place to start. → 608.30
:23 How do Rolo differ from INV probs?
Well, in d-END : Given X, ylo find M(X) so M(X)=y & M(·) is minimal:
Say M is a somy, then x & y detb ti unknown string, M. (in ANL, key is lay, X is long random string)
To find an M that satisfys this dcrn is a INV problem. Ti goal in INV problem is to find M w. as little cc as possl. But if Length used, M tends to have hy PC, & ∴ PS ∴ a good bet for a soln. to ti INV problem. So ti difference then, is that in INV, we want w. as little cc as possl, But in END, we want a M that fits, w. Max PC, w. cc ∈ CD.
(woops! → 608.15 → 608.25 intuitive version also 608.26)

In a (mildly) advanced TM, the GPD's used in ti INV v.s. END problems; will differ, since we are optimizing different things.
When we Run these problems, we can give TM ti problem first as INV, then as END or ti other way around: Or just say ti 2 problems "indisply" so ⬛ when TM works one, it doesn't know about soln. to ti other: ⬛ Look for differences in TM's behavior in solving ti 2 types of problems. I suspect it will get same soln. in both cases.

Unlikely! see 607.15–23

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street, Cambridge, Massachusetts 02138 (617) 495-7908

.001 607.40: There are 2 elementary ways to work ~~[illegible]~~ INV problems. Given M(·), c to find ~~#~~ x ∋ M(x) = c!
.02 A very ~~elementary~~ elementary way simply trys x's in ~~[illegible]~~ ≈ Lcost order. This is rather poor, but ok for very simple problems — particularly if the GPD is not yet much good. Maps
.03 The more advanced way (true Lsrch), is to hunt for f(M(·), c) that maps them into x.
~~Also~~ It may be that .03 would have solved ANL a lot easier than the method I was using!
— I think I may have been doing .02! ~~[illegible]~~
In Search, on ANL, I just tried functions from Q to A in ≈ pc/Lcost order.
In .03 I would try to find a function that looks at both Q & A as inputs & has as ~~outputs~~ trial functs that map Q to A.

.10 [SN] In Lsrch, I want TM to return trials that "worked" but were, at the time, of ~~[illegible]~~ "not top pc". This is mainly for "Theory Revision" — But in addition, for "Theory Revision," I'll want these "suboptimal codes" to be as "difrnt" as possl. from the formerly "Best Code". TM will have to Learn a good criterion for "diffrnt".

.15: 607.30 Hmm! Thinking About D-IND v.s. INV probs (607.23–.30)
.16 As a INV problem: TM looks at Q & A & problem is to get a funct that maps Q into A:
.18 { A do so at low cc: There is a trivial soln: The function looks at its inputs; if it is exactly Q then
.19 { output is A. : Well, usually the ~~[illegible]~~ "IND" problem has a larger corpus, or, the Q is a ~~[illegible]~~
.20 Very long random no. — so the function in .18–.19 would be very long. But still TM could easily map from Q to A w. this ~~A~~ A.H. function — and even if there was a larger corpus of ~~[illegible]~~ QA problems (~~[illegible]~~ (No Q has >1 difrnt A), TM could make this simple A.H. function.
.24 [The Real difrnce betw INV & D-IND is that INV looks for best soln, IND looks for hyest pc soln. in CC & B. The Goals are quite difrnt.
.25 [
.26 On the other hand: If INV solver looks for trials in ≈ pc order, the soln should be ≈ that of the INV problem. But INV solver, in the most elementary version of Lsrch, looks for functions in pc order! but in true Lsrch it does .16 & then .18–.19, so it would not be like ~~[illegible]~~ D INV.
.29 But drop this for now; I have more impt stuff to work on!

.30 : 607.22 I want to work on minimally diff't IND problems, because the extension to S-IND problems is very important — it is ~~[illegible]~~ Essential part of UPDATING.
On the other hand, Using only D-IND, could TM learn a useful, signif't part of Math? Seems Unlikely, since much of ~~[illegible]~~ Hours of Math is Stochastic.

.34 Hvr, in this early D-IND [illegible] set of "STUDY Problems", I will be doing "Updating", by by using "canned" induction techniques like AZ AI & looking for common ~~[illegible]~~ Sub-functions (≈ SubGraphs). Also (perhaps) considering simple "contexts" for pc's of concs to be used in ~~[illegible]~~ cond construction.
.38 After .34 ff, I may want to try S-IND problems: Getting a useful Lbry of S-funcs & learn to combine them into ~~[illegible]~~ Conds for S-IND.

25·1 ok
24·15 ~~[illegible]~~
(9 [illegible] ok
78 [illegible]
78 [illegible])

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

00:608.40 : W. regard to (S. IND / S. funcs) (608.38 ff) etc. I expect that when I give TM these probs, I will look at how I seem to solve them so as to get ideas on how TM should deal w. them

Re, Incremental Updating & IND in General: I want TM to be able to look at an IND problem & have ideas as to what kinds of regularity to look for — I had this idea of a PST looking at a prob. & having a probability order set of techniques

.06 for solving it. In the case of IND probs, one way to (partially) realize this is by associating w. the Observed problem, a set of expected (or hyper/pc norm) regys to look for in that ENV corpus. This seems to be pretty much how humans Operate.

.10 A very impt. Heur in IND is the Partitioning of the Corpus. While I have written a few comts. about it, along w. some ways to implement it: I don't feel that I understand it well at all! ← Well, in partitioning Corpus; T partitions are sub corpi that look like they may have certain regyes. A subcorpus can be characterized by the set of regys it is likely to have. } Good!
Perhaps .06–.10 is part of it! (← later ... what was I thinking about??) maybe!

[SN] In the GPD since it will be looking at all aspects of TM's behavior, as well as external problems, It is likely that there will be serious "Double Counting" Problems in the GPD "array".

.19 / .20 [Talk Title]? A Set / ensemble of Efficient Mechanisms / Algorithms for Machine Lrng. (Some) Efficient Algms for M.L. [see 584.08 for possl. titles]

"Start out w. a very general picture of how it works: ≈ 601.55–602.04
Then more detail on PSTs: what GPD is: & how it is "Updated"
That this General system is of the "wrapper" class [Look up wrapper on Web again; got good Refs]
So reader/audience has general idea of how system works: then go into more

.25 detail on mechanics. — perhaps describe "steady state" first, then → 610.00

.27 [SN] In Report, Discuss "Adaptive Lsch" tell how a Berno seq of symbols can be put in pc order (using Huffman coding)
→ It may be possl. to devise a fast way of doing this based on the way "Arith. Coding" works. ← Likely, is pretty Fast!

30 — Just do all codes n bits long: For each code, find assoc. Arith code using the variously ped symbols.
[I suspect that there are as many solns to this problem as there are proofs of the Pyth. Theorem]
I think the arith code works fine & fast — I don't want to write it up just now, tho.
The idea is this: Say we have this n bit binary code. It corresponds to a specific interval on (0,1) & is of length 2^{-n}. We list the symbols in some fixed order (e.g. pc order)
So pc's sum to 1. If the 2^{-n} interval is not on any pc boundary, then the interval is in its first symbol. We then stretch the pattern of boundaries (pattern of boundaries) so it fits exactly in the interval of the selected symbol. We then select the second symbol, depending on where the 2^{-n} interval is. — We then loop to α
→ What we do if the interval is contains a boundary, is not clear! This is one of the diffys of arith coding! — we could just neglect these cases: they are (much less likely) (see 610.11)
after $T = 2T$ we just do that up to pt. where it's ambiguous.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: (spac 609.26): Derb. GPD in more detail & how it is updated. Then how the set of PST's is updated — then the more general problem of both creating (new) & modifying (old) PST's & assigning better pc's to them (i.e. the "PST Grammar")

.03 [One problem: "PST Grammar" is Derived by Unconditional pd. — Not at all what we want. We can get this set of PST's w/assoc uncond. pc's — & have the pc's modified by individual users. This essentially separates the Generation & assignment of PC's into separate tasks. Ideally, for each (new) (problem) applicn. (or to TM) of the Possible set of PST's, a separate grammar

.07 would have to be formed.

Discuss how various INDuction Methods can all be used to implement this system

.09 [I have big list somewhere: I think in a "recent" report: I recently added "BBN" to the list. — That's 593.17 → .29: it listed derbs & most of finds.

10

.11: (spac 609.40) SN This Arith Coding Method looks Good! Can be very fast, & can be adapted for some inter-

— Symbol dependencies.

.13 A (New) Very simple soln: List all symbol sequences in pc order; cut off when pc < T 2^{-n}. NB this Method can have pc's of cores (change dynamically) during seq. We can also do certain intersymbol dependencies this way, & is still (sort of) work. GOOD!

An advantage of the Arith coding method, is that we actually get a (binary) code — tho I don't immediately see how to use that! ☺

Using .13, it may be possl. to use a pgm like "OSS 86" to do Lsrch. So far, a main trouble w. this is that my way of Realizing a UIO was via AZ141, seems very inefficient. Hvr. the 3 input uni form of S-function Does need to be worked on / fixed — but this is a "well defined problem" & I may be able to get others to work on it, if it is clearly a bottleneck.

20

.22: (09) To start off: Derb. 'INP, INV, (OPT) problems: [redacted] They can be [redacted] (formally) solved by Lsrch. But Not all problems are of these kinds: H. has extended Lsrch to solve some of them. In both L's & H's problem types extended Lsrch to solve some of them. In both L's & H's soln. methods, the algms used are optimum within an additive c/o mult constant. in reasonable times, using In both cases for both kinds of algms, the consts are too large for them to be executable present day computers.

Since computation speed is doubling every 18 mo. — could we not expect that we will soon our computers will soon be able to use these algorithms effectively? Apparently not: — for the relatively simple problem of finding the general solution to a quadratic

30 equation, using Levin's universal search, we must wait about 70 years for computers to be fast enough. For marginally more difficult problems we must wait much longer.

I will describe a class of general problem solving systems in which the execution times for Levin's and Hutter's search algorithms are much reduced. Using present day we expect [redacted] they will be able to solve most serious problems computers or those of the near future. (Footnote: This is not the same as saying I expect the system will be able to solve serious probs in near future — I mean to say only that the bottleneck (in AI) is not in computer power but in finding appropriate algms), (in Machine Lrng)

Look for Algms that use multiple Matrices as well as a proof that the approx bad for a section of this algm is accurately computable for the size. [redacted] call this Cond. Algms.

612.00 spec.

4.28.02 (611)
1D

THE STACK 6.62.17 INDEX

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 610.40 ; This "Stack" is a list of items that I don't want to work on just now, but that are eventually of much import.

1) Details of my demo of "How to do Hutter's Srch better"; Also I need a better understanding of just what class of "well defined probs" it will & will not solve. Hutter: 522-524, 557, 576, 595.18-19, 599.30-.32 Mom Theorem (— use it Lsrch to find (soln algm + proof + time bound))

2) Discussn. for report on How Prz system can be extended to "Hutter Srch"

3) Discussn of Generality of Methods — that other induction Algms can be used, other PST's can be used — Giv. partial list: GA, ANN, RANN, ID3, BBN, CV[illegible]M's, ALP (MML, MDL), sim annealing, Lempler-Ziv, Gramm induction

4) 610.03-.09 : S Gramm of PST's is an uncondl pd. How to get needed pd conditional on problem defn.

5) "Anytime probs": 612.00 : still some confusion in my mind about the exact defn. of problem.

.10 6) On the meaning of & guiding pc in Lsrch (what's it the pc of?). 619.05-.19
This is not a critical Q: I could use a short form "or 2 stage form. — Some of cc saved in short form compensates for loss of accuracy" see (9) (17) See 674.28-.40 (33 interesting): 672.16-.20

7) On a possl. very serious Bug in Z141 & AZ141 (620.22).

8) Fast N Dim continuous Hill Climbing (say for Protein Folding): Needs More analysis 587.08-.30

.17 9) Easy integration of & 2 step GPD needed for guiding Lsrch in OPT & INV problems. (i.e. all problems). 628.38 ff: I'm not sure this (is possl. / will work) — but I should look at it more carefully. I. Monte Carlo method would work, hvr. , & may be good enuf for most cases. 639.17-.19

.20 Another possy is 641.23-642.02 : 644.12-.14, 644.30-645.03
This last approach doesn't do Lsrch at all! — just picks & best (problem) cond & works on it until it is clear that it's no longer best cond (or until prob is solvd). (Also see (6) (10) ... 619.5-.19)
Note: 674.00: This 2 step process needs only low precision & & can be done fast! 674.09 ff is v.g. discn. ... very Serious Diffy

Later: I used WON (12) instead. Seems much Better!

.24 10) Integration, Genzn of: AZ141, V.S. "Jurgen" method of assigning pc's to pgms:
[illegible] - 670.02, 657.33, 658.21 ...
669.30

.27 11) How to Deal w. Logical reasony: 673.24

.28 12) WON: See NIPS 166.00-181.40 for fine developments. Won replaces Lsrch for INV, OZ and can solve Time varying OZ!

30 13) (Genzn of Gambhouse Rm F): Applied when conds are correlated": we have a joint P.D. on all conds of each cond being a "win": >1 win is possl. 685.26 ff: Not much success! 686.16 try to define Genc for Z cond "Lookahead". (k cond lookahead is main goal.) Confused.
[688.14 then 689.04-.17 seem to be a formal statement of a soln. criterion]

.35 13) That sequential coding of sequential a/o Q A corpus may be "not bad" for a lang. that has defns of functs, a/o n tups; & defns of contexts!
See 768.20-.40 for prelimy argts | Also Note NIPS 184.10-.24 & 184.25 ff on Backtracking"

.38 14) Akaike factor: Akaike factor is a — $\frac{\text{mean}}{\text{num}}$ for continuous params & n data ... values. : Can I extend it for discrete symbol preds (say binary)? Genz
See NIPS 192.20-194.24 ; 194.18-.26 defines prob[illegible]
NIPS 186.16-.35 ; 187.20-.23 is v.g. (186.30-.35 is Main idea) → 611.00

I'm not at all sure what "Akaike factor" is. See idea of NIPS (186.30-.35)

5.22.03

ID

THE STACK

611/10

00:611.40 (15")

10

20

30

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 610.40 [SN] On [Anytime Problems] If ∥ Lsrch is used for OZ probs, then we are also solving Anytime problems (usually). [redacted] Many OT's obtain a sequence of "Best [redacted] soln. thus far" objects — sequentially — getting better & better as time goes on. For Anytime probs, this is fine. Hvr, some OTs [redacted] don't work that way: They solve OZ probs & they must be given a CB as part of the problem. If we restrict our solvers to OTs like in .01–.02, we will be solving Anytime problems, but if we want to work on Anytime probs, & we want less restriction on OT's, then we need a better defn for "Anytime". If we solve OZ probs using T→2T, then we will also be sort of solving an Anytime problem. Probly I shouldn't spend time on this now. Just treat probs as OZ probs: Use ∥ Lsrch or T→2T, and keep track of "Best Soln. thus far".

Well, not so clear! Even for ∥ Lsrch of OZ probs; One has a given CB: One runs the system for time T, then stops. During that time one or more of the OT's may or may not come up w. a "proposed soln." — or none of them may. So one increases CB — But some OT's may be able to simply continue. Others may have to start over because of the change of CB.

Reminds me of the 3 input UMC to realize S-funcs: The [redacted] AZ141 can only do functions that are of finite defined length. They do not give partial output for partial input: Just as (perhaps most) OT's don't give partial output for partial times.

AH! for ~~simple~~ AZ141: Say we feed in partial input: the Algm can output partial output. I.e. all bits of output that would remain the same, independ of the rest of (the unknown) bits of the input. While AZ141 could be made to do this, it would do it very inefficiently — spend much time doing calcns. that I would hope to be unnecy! All I want to do is

Perhaps a better way: Try to realize S-funcs the way humans seem to. Have O" [redacted] realized by inputs to UMC: One seeks O"; Another is Q; the output is a ... P.D. on A: this can take many forms: 593.17–.29 is a little list of forms: ~6 forms listed.

The BBN way of approximating S funcs of strings to strings (discrete to discrete) is something I hadn't considered. Even the individual nodes are of interest! The K input string input w. single binary output: The [2ⁿ parameters] are a complete descn. of all possl. funcs of this sort! A kind of "Universality" — But it seems unrelated to UMCs!

Hvr, even in discrete to discrete S-funcs, the no. of possl. outputs is often ∞, so the finite no. of params in the BBN may not be adequate.

One universal form: UMC, 3 inputs Q, A, descn of function: output is binary expansion of P(A|Q)! Since most functs of Q & A will not be probabilistic — (they will not be ≥0; they will not be normed.) — this would seem to be an inefficient way to make trials (i.e., trials for a dsn. function). Since the output is not normed usually we (probably) couldn't use it in 4 (dse).

Hvr, since the function descn is an adaptive P.d. — when we search for good s.funcs, (613.00)

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 612.40 we will (eventually) find ways to (probabilistically) restrict trials to funcs that are normed or easily normalizable. It may be easier to get Az.141 to deal (easily) normed S-funcs. There may be a way to take a small no of S-funcs (maybe just one!) & ~~...~~ do functions of it that always yield normed Pd's — or when they don't, it's very unlikely that the unnormed part will be large. (also, say the d.f. w/s ~~always~~ ~~subnormal~~ never supernormal i.e. normz constant is always ≥ 1)

Anyway BBN is a good thing to keep in mind when trying to generate S-funcs.

[Remember: I want TM to be able to use all conceivable forms of S-funcs because each is really useful for certain kinds of problems, & TM should have access to these hours (as well as any other hours).]

→ I really want to do ~~...~~ [612.36] Get TM to realize S funcs the way humans seem to.

So, when I am constructing the TSQ I should be mainly interested in trying to explicate my own methods of inducing S-funcs as solns to INDuction problems.

[SN] More impt than "the Report": a clear statement to myself, of just how TM is going to work, the list of various unsolved probs & references to how to solve them (if possible!) 601.35 – 602.00; and ~~...~~ 602.10 & 593.10 which also deals what needs to be done now.

For purpose of Exposition ("T. report") I want a descr of how TM works "At the Beginning". What PST's it has; How it works IND probs, How it updates. → .34 "Encyc"

.18

.19 [SN] AH! A kind of Breakthru! "Cure Cancer" was a strong step in this direction ~~...~~ (584.33 – 585.02) ~~...~~ was a soln of the "Encyc" problem, but it also includes the general case of a very large corpus of data.

The new item, is that ~~...~~ the "Encyc" can be R.W. In general, ~~...~~ TM has had no direct contact w. most of it. I had hypothesized TM treating data in an Encyc. by telling where things in the encyc — giving coords or methods of generating coords.

For Encyc ≡ R.W., we can do ~~...~~ similar thing, by giving instructions on how to make observations in R.W. (an "Observation" is an Analog perception in R.W converted to digitals for TM input.)

I discussed the R.W. data problem in the Aust paper "Does ALP Solve the prob of Induction?" I think in that paper, I didn't consider it to be included in ALP.

The Larger formalism would enable one to answer the Q: What is the density of Hg? — if the density of Hg had never been measured before — (or never written down for public access.) TMS

It enables TM to answer Q's involving ~~...~~ (Observations / measurements) in R.W. It means that TM can ("command" / request) observations in R.W. I sort of got into this in the "Cure Cancer" problem, but I didn't really think about the TM—R.W. interface much.

I didn't really think about the TM—R.W. interface much. T. Big Idea of the "Encyc" Soln. is that TM avoids coding the entire Encyc, by coding only the cond'l QA's

.34 (.18) To start: TM works ~~...~~ D. IND probs. (These look like INV probs, but they are not!) [609.05 – .29 see ~~...~~ 607.20 – .30 for relation to INV probs]

The induction method is to find max 414.18, but all $O^n(A|Q)$'s are 1, The only thing we need to maxz is the PE of ADfunction). At first this will be done w. "pre-Lsrch, in which we search directly for the Dfunct. in pc / Lcost order. I may have done this in Search, ANL TSQ; A more advanced Method (Maybe Used in 586 for ANL) ~~...~~ looks directly at both Q & A & try's to find a function that maps this pair into a function relating A & Q. (i.e. $f(Q) = A$).

It has to do this for all QA pairs, so it has to be able to look at all QA pairs, & from this info find a suitable

4.29.02 6 K1
ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 61340: D-funct soln.

So: BIG Q: How to solve D-IND probs.? (symbolic Regressn).

I've certainly written a lot on this: but I don't remember much!

One approach was to [illegible] ↑ uncond'l pc of solution functions by ↑ pc's of its component functions.

This was done via TSQ.

The other BIG PROBLEM is soln of S-IND Probs: These 2 probs are related: Update of D-IND prob GPD is a kind of S-IND problem — but I may be able to devise simpler solns for it than the General S-IND problem.

.09 Back to D-IND: One way was to get/funcs (different functions) that solve each of the QA's in the corpus,

10 — then try to "integrate" them. — This makes the induction a 2 stage process — which can simplify it. This was sort of done in the S 86 treatment of ANL

Testing conds is somewhat faster (tho, because it is D-IND, a trial for the entire corpus will tend to be aborted quickly if it doesn't fit — no need to test the entire corpus each!)

Tho, we should be able to find functs for the individual QA's much faster than for the whole set.

Rgs initial fitting to individual QA's is BLIND (No FR) srch. Also Unconditional srch.

On big Q was whether this Blind uncond'l srch, guided only by a TSQ, could give TM enuf smarts to begin using cond'l srch & Non-Blind srch (i.e. feed back from previous successes & failures). This learning could be via [illegible] update of GPD —

.20 a/o (preferably) by the content of stuff it has learned.

The content dependent pc's of concs in a cond, could be uncond'l, or we could expand it so the content includes the "conditions" (e.g. the problem descr, or the local sub-problem descr.).

.24 [→ Anyway, just write down how I seem to solve the probs & then design a TSQ so that TM can do same thing.

My remembrance of this is that I had trouble doing this & that I wanted to write a "Reasonable TSQ" (that ought to have needed info) & then let TM work on it. —

So I'm not sure I can do .24!

It may be that now I will allow more complex Heus. to be used — since I may be able to get TM to learn them by long dating & various QA tricks (cindarey "— ...whatelse?)

30 Anyway, try it again.

BTW. I think I did use a kind of S-funct as an intermediate step to a D-funct! Before learning how + → sum, — → subtract, I did (sum sub mul div) each in pc=.25.

So 3+7=10 wasn't hard! but nested ones later, were hard ... & I started trying recursive functs, since humans do learn them easily. I did want TM to learn idea of "Quantity" That stuff inside parens could be evaluated (if possi.) & substitution made. → 6/6.00

.37 One way might be to show TM example of sequence of operations in evaluating an expression. — This would be a Very GROSS "Hint"! — But it may be a way to get over a bottleneck.

4·24·02 615

ID JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00; 614.40; Would I need a ~~special~~ learning feature to get TM to watch/a (reinforce) an accurate — Or can I do it within the present structure? Perhaps study learning by example" & see if I can fit it in w. less "Skinnerian" lrng. (.14)

.02 [SN] Re: Hutter: List various kinds of problems in which Lsrch doesn't seem to work; Show that in some cases it does work anyway & "constant factor" in can be ≈ 2. — This is using a Hopkins pst's.

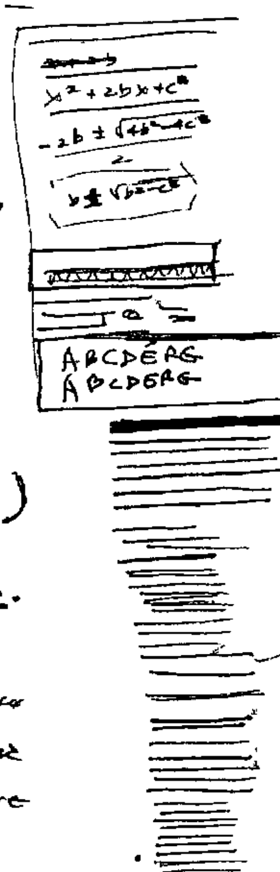
So cases where Lsrch doesn't seem to work: 1) Min time for product of sparse, large matrices.
2) OPT probs; Greedy Gen take very much time (Best landing, SPICE (for ea. sim. G.A.)
3) INV probs in which verif. cc is very large. — So ~~its better~~ one way that's feasible is to find probably faster verification routines.
4) One trick (≠ Lsrch) that humans use; try problem soln. for small N, random params: if it works, try for a few other small N, random params: then it's likely to work for larger N.
The small N examples may suggest proofs.

Newton / Newton-Raphson code — Yallo 1) ~~[illegible]~~ Shorter inst. Shorter run 2) ~~[illegible]~~ ~~[illegible] line~~ 3) ~~[illegible]~~ but looks O.K.
Newton / Laguerre code

.10 (.02) Well one "Skinnerian Soln": Give example of correctly worked problem: Give trace of soln. This trace is part of the Q. The Q consists of problem + trace of soln. plus a new similar problem.
≠ This particular format has an index that TM lrns to recognize. TM is also given the answer — which is ~~a~~ some modifn of the trace in the Q — is applicable to the new problem in the Q.
If the problems are all Inv. probs, so TM recognizes the soln., it can search over various modifns of the ~~trace~~ "trace" in the Q for trial solns — it can do a "search".
After it finds several solns to problems in this way, it will try to find a common, general, way to map from the example traces to the correct solns.
The "Skinnerian soln" would be to give a sequence of problems of this sort, in which the soln. to the problem & the trace in the Q differed by more and more in successive problems — so TM would gradually ~~adapt~~ adapt to this small difference — & eventually, to a large difference. (I'm not so sure this is really as bad as normal "Skinnerian [illegible]")

Another (apparently conceptually diffrnt way): Put the traces of the examples in Q. This makes it harder for TM to lrn to use that info.

A somewhat better way: Q ≡ (problem, "Hint ≡ Trace"). TM must lrn that the trace of the soln. to the problem is ~~a~~ "closely related" to the trace in the Hint, so it would be a good idea to try modifns of the Hint trace as cand. solns. The hint trace & the soln trace are perhaps "Analogically Related".
The "Hint" info could be regarded by TM as ≈ "Context" info.
Perhaps Get that Book "Your Wish is My Command": Prog. By Example: H. Lieberman, ed. (#50) Morgan Kaufman Pubs 2001, 448 pp
Drop this for a while Back to 614.37, → 616.01



4.30.02 616

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

6 spac 14.37 | 6 15.904 [illegible]

Back to 614.24 : One offspring of this idea is to write a TSQ that I would think to be "Adequate for Me": Then see what ~~assumptions~~, what background knowledge is assumed — & see if I can teach that info, via TSQ on ~~[illegible]~~ "Brain Surgery", or whatever.

I had that idea of linear eqns → quad eqns → cubic eqns soln, using same hay — I wonder if I could really put it into a form that TM would Understand & use. 42

"Understand" means ~~[illegible]~~ be able to ~~genz.~~ to other situations — either Generalize parts of the idea or just the Whole thing. 40 52 69

"Understand" — completely — to know all possl. codes of the (idea/subcorpus), to know all shared codes w. all other possl. ideas. Implies knowledge of all analogs w. all other concs. possl.

An offshoot of this: ... seems very reasonable! I may have tried it in the past!

If a TSQ seemed "reasonable", but there were parts that I didn't know how to teach TM — that I should clearly state what the diff'y is: and write the idea into TM, & see how it works. Try to figure out a way to ~~[illegible]~~ do it better ... "Later"

The list of "unsolved induction problems" may suggest way(s) to deal w. some of them.

It may (indicate / bottle / reveal) areas where my ideas of induction are inadequate!

Simply being able to express these Lacuni in TM's knowledge in an exact form — would be ~~[illegible]~~ very useful — toward dealing w. them. I have to put them in forms so that TM can use them.

Consider one bottleneck in ANL: Learning that things enclosed in parens can be replaced by what they equal, & ~~TM has to know how to parse~~ (in)equality of the ~~[illegible]~~ parent expressn will remain invariant.

This seems like a very complex statement!

We usually use the more approximate idea that if $a = b$ (a & b can be any expressns) then we can legally substitute a for b in any expression.

Factor it into 2 ideas! (1) ~~[illegible]~~ $a = b$ (2) the substitution idea.

In the ~~[illegible]~~ Pure. I haven't yet introduced the "Parenth" idea!

I think that with ANL, One heuristic "Hill climbing" idea, is that we want to end up w. a single number, so when we substitute say "10" for "3+7" in an expression, we make it smaller — so ~~[illegible]~~ "closer" to acceptable final form (using a "natural" ~~[illegible]~~ size metric.)

Humans probly have that metric (of size) "Built in."

One way to do it, would be to write out the conceptual sequence for Humans in English — then try to fill it in for TM.

Another idea was to write a sequence of problems that I would expect TM to be able to learn! Then see what concs are needed, then fill in "inter-concs" then devise examples to lrn the concs.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 :616.40 : Some stuff on TSQ's! 520.00

Another remembrance: that I thot of just writing a soln. for ANL problem (t. recursive soln.) and seeing how much CJS it had — & whether I could reduce it. At that pt., I found that I didn't have clear, exact Model for t. needed induction: So presumably, I now do have an adequate model.

As it is, I think I have a good model for t. "over-all structure" — that what is Really Needed is ① TSQ ② Heuris, concs for TSQ ③ forms of S-Functs for S-INDuction.

A Main, Crucial concept in this TM. Model: GPD = Trainable Condl. P.D.

T. Swiss Army Knife of AI. — Give Google as example of

.10 — input & come but output has no pc's: A's in ≈ pc order.

Cyc mite be similar.

SN What about use of ≈ APL for TSQ's? Does it have good string processing?

SN What May have happened: as to S89, I spend so much time on TSQ's that I really forgot t. details of how S89 was supposed to work! So: I do want to write clear that "t. report" & my own "Footnoting" of it, is adequate & easy to understand.

SN In INV probs (& perhaps IND & OZ) If we look for a mapping function that goes

.17 from problem dern. to soln — Is an uncondl P.D. adequate? What we are looking for is to some extent, a condl P.d.. This is Confusing!

.20 T. condl. P.D. would seem to be doubly dependent (redundantly so) on t. prob dern.

If I could use an uncondl. P.D. here, it would reduce (complexity/diffy of problem) considerably!

It would mean that I'm essentially (asking for a single Grand PST, t. same for all problems — but this PST would, presumably look at each problem & decide what kind of "More narrow" PST should be used on it.)

.26 That uncondl GPD still is "adaptive" in t. sense that it is modified as t. TSQ continues. → 619.02

Does this modify (simplify) IND problems? were and to Marz 414.18: We want a function that is able to look at t. corpus (& perhaps previous traces of TM's induction behavior & result) — & propose

.30 — Various O''s as conds. We may want to have t. function not look at previous INDuction traces — & (instead) have this info somehow integrated into t. GPD! This seems Likely an IMPT Q. Is this poss'l.? .30–.38 is a relevant Remark

.38 In t. case of 419.18: T. problem is "Well Defined": TM can use any informational a/o cc that are available to solve that problem. Hvr, t. boundary betw. what activities can be regarded[redacted] as "part of GPD" v.s. those activities regarded as "part of t. problem soln. algm" — unclear!

.35 { I want to make t. division in a way that is "Best" in t. sense of long run low cc [redacted] a/o

.36 relatively simple pgmng. a/o simple "data structures". This is t. Q of .30–.31

In general, we have Q of allocation of activities betw GPD & PST's.

The PST's can be regarded as "part of" GPD.

It may be that t. only difference in "allocation" of this comp. depends on whether ↳ ≈ "Naming" as in GPD or Not in GPD →

5/2/02
ED

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 617.40 : [redacted] SN Re: Updating; A PST can (1) decide about whether to wait for end of prob. to update (default), (2) wait for seq. of probs of "this type" before updating, (3) update after each trial (within t. PST), (4) update during trials. It's not clear how t. PST gets feedback from updates after probs or after [redacted] seq of probs of "this type" — i.e. so it can "improve itself" — or even initialize itself!

.05? (617.40 spec) : we will need to report t. calcs or whether it's very "problem specific" to t. present problem. (for some other problem) If t. latter is true, we should see it; but in some cases slightly than t. stuff we allocate to GPD.

So: RE 617.17: Try TM w. uncond'l GPD. GPD looks at problem & gives/ordered list of PST's — (pc of each being t. "Best"). I want to be sure this really works)

[redacted]10

The GPD Does seem to be a cond'l. P.D. we insert t. problem desc. & we get a P.D. on PST's. For a INV prob: when TM is somewhat young, T. output(s) will be a sequence of attempts to map M( ), s into x. So GPD will guide Lsrch, directly.

When given INV problems later (a more Mature TM), T. (M( ), s) input will give several possl. search types: (1) regular Lsrch as younger TM would do (2) General OZ problem — that devises a GORC for t. problem & calls on General OZ problem Solvers.

(3) Devises a Vector Gorc & uses a GPS type srch or modified GPS.

[redacted]20: SN "Proof" that $k(x) + lg_2 ALP(x) \leq 1$: If ALP(x) were a Comp. (usable) Prob. measure, we could use Arith coding to show this. (except for "occasional "bug" in arith coding.

Hvr. consider $ALP_T(x)$ — this is ALP w. c.B. = T — which is computable Prob Meas. for large enuf T $ALP_T(x)$ is arbly close to ALP(x) ; So there exists a code (constructively) so that $ALP_T + k(x) \leq 1$.

So t. big Q is to compare Sum of pc's of codes thus far, w./ pc of shortest code thus far. If sum of codes is much better than t. other (May are both usually easy to compute) use ALP; if not don't. [e.g. Using all codes within 10 bits of "best code" in ... expensive]

UR
DL
LR
LL

UR
LL

5 LR (both "longer" Gold Crown)
6 UL "Choose"

.30 SN On Partitioning & indexing: Normally, t. corpus will be Partitioned by indexing a/o by nature of applicn (such as "updating" vs. external problems). So each Partition could be initially regarded as a separate Induction problem w. its own assoc. 4.14.18 to Maxz.

If we want to merge 2 Partitions, we pool their corpi & maximize common 4.14.18. When this is done, we usually try to take advantage of info in t. original codes of t. separated corpi.

.35 → We'd like to use t. original "regularities", plus find some new ones. T. method of doing this trick is yet to be discovered! I Guess it depends on just what t. regularities are: some are easy to merge, others aren't. The easiest is when t. 2 subcorpi have common defined concs, so we save t. overhead of defining.

Normally, one does optim Maxz 4.14.18 over all subcorpi — but t. value of 4.14.18 is often

5/4/02 ED

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00:618.40 : The product of the factors for individual s.c.'s: Hvr, I expect that they parts allways use a common "Definitional Pool" — which gives minimal (many of all s.c.'s.

.02:617.26 : I think the GPD is always "conditional": The condition is always the problem defn; "what is being asked?" of GPD. GPD is like Google — it has to have an input (its condition) in order for it to have useful output.

.05 [SN] (EXPO) In explaining why we get the PC for Lsrch in 2 stages: We want the pc that a cond. is the Best soln. This is the product of 2 pb.'s ① The proby that the cond. is a soln.

② The proby that if it is a soln, it will be the "Best" soln. The first factor is usually cf I's ~~the second~~ [illegible] The definitions of these 2 factors are unclear —

10 Well, the def. of ① is clear — I can see it as an reduction problem on the basis by defined corpus.

I AM confused about this! (put back to this later!) → Stack (ED 611)

↓ On second thots on Reading Jaegan's stuff: he does seem to use the previous successful solns as a corpus to get a PD for subsequent Lsrchs: We mite say — that the corpus consists of min Lcost solns of problems — which would seem to be a reasonable corpus!. But not so! For the various solns, the pc's are from a different P.D.'s so the examples in the "corpus" are not v.g.

20 → Anyway, what we want is a characterization of the successful cand itself — cand
i.e. how much cc to solve its problem, not much on the pc it was successful.

On the other hand, using previous soln.conds as "the corpus" is easy to do & seems to point in the "rite direction"! Tho it may not be exactly rite, the speed with which it can

25 be implemented may override its other "imperfections". → put (05–25) out stack → 611.

[SN] On Convergence of $(pc(u))^{2^{-n}}$: The signific: If a code gives upto n, bits long works, then the proby there continues with U, is very small (as $n\to\infty$). Of course it depends on the Generating Dist μ.
~~[illegible]~~ For fixd c, B, T_0, perhaps one could guess at the form of μ!

30 For fixd c B, T_0, There are corresponding "Convergence Rates" for 0, 1 & U. One mite be able to estimate "no. of continuous params" in ~~[illegible]~~ μ(!). Hvr for linear Regresn (say) if one used integer coeffs, & the usual methods of linear pred'n, one would find no. of coeffs, not the fact that there was a finite dsn. (Tho actually one could get the no. of bits in the dsn!

Say we knew "a'ne" what ~~[illegible]~~ how the generator was! its integer coeffs: We could use "near by" integers & fractions to approx the true integers — this would give the expected Convergence rate.

Ah! Say for codes of upto n bits, all but .001 of them continue to n+1 bit (at least)))??
Then this is evidence that the errors for the pc's of 0 & 1 for n+1 bit are ± .001 ??

5/6/62 Z141; Wolff 122 620

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 6/9:4 SN Cover's "Extension Complexity" is a v.g. (maybe exact?) approx to ALP. So it discreates Univl. d.f. is adequate for all induction!

This discrete d.f. does not need a UTD machine, but can use an arbitrary Universal machine, with the width of a code being ϵ^{-1}, n being # of input bits "looked at" by the machine.

Whenever one does use a UTD machine for Extension Complexity, its CC is very large! >>> Max needed for UTD machine.

That the convergence of definite ratios wrt. the Kolmogorov complexities is surprising!

An important application is to Universal Machines. Whatever one's TRMs: Most of modern computers CPUs are not "Like" TMs at all.

A Big problem here is that for a sequential corpus, Extension Complexity is a very inefficient way to do induction. Each possible continuation must be individually considered/evaluated. I ran into this in the "BES" (BDS) Algorithm for detecting Nonlinearity in a time Series of reals. I also ran into it in using AZ141 & its "Lisp" interpreter.

I probably will want to use different methods to implement S-functions. — However, I do want those methods to be Universal. A possible course of action: Just work problems in which I do know a reasonable form for the S-function solution. After working on this a bit, I may find a way to make the system universal. I can start with "frequency" model of pc. I can put the system to invent new objects to make progressive runs on — using Z141 (or AZ141?)

.22: SN My Z141 model may not be so good for parsing English! My model considers that the corpus was generated from primitive characters by chunking them together, & chunking the resultant new characters together. This results in very many logical parses of a particular string being analyzed. However in English, there are relatively few logical parses of the text. (I'm talking about the word structure, not sentence structure — Even in case of sentence str., English has, again, many fewer possible parses than a PSG, say). Will this also cause difficulty in function definition in AZ141? SG4b = Z141

One Big Trouble in early work on Z141, was that I always assumed the every "AB" that I found would be parsed as "C" (if CEAB had been defined).

What about the "Super L-Z" method that I invented? Would it be useful for inducing functions? This problem is very serious: The whole basis of TM is defining rules, & defining other rules based on them...

.38:59&.32 Re: the difficulty of .22 Af: I may have not properly considered (1 coder)! This might account for the "½ factor" disparity! In "reparsing" one only get one of the (1 codes). on making too a near data definition

622.35 Space

5/7/02 621

IQ

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 : 620.40 : [SN] Is "Adaptive Lsrch" Uniformly Better than GA of any kind?
Perhaps it depends on how one "adapts" — how fast, how cleverly the update models are.
It also may depend on how Good the "GA" models — what kinds of Muts, crossovers are used.

Also note: that GA uses random trials with replacement would seem to be a flaw of extremely — but apparently not so! [In GA, we normally have ... % of cases of "duplicate trials." — we depend on a very rich set of || solns.

.06 So: [At present,] I have GPD which contains "½" of TM's info & [PSTi] set, which contains (most of) the other ½ (another "½" contained in "Encyc".)

.10 As the STQ is fed to TM, GPD & [PSTi] are continuously improved.
As probs come in, GPD assigns the PSTi — then as PSTi works GPD watches & may switch to another PSTi.

The exact structure of GPD, & [PSTi] will vary during TM's life, in response to the trainer's direct manipulation.
Partly to y. TSQ & partly to the trainer's ... "Details" have to be (decided on/discovered.)
So The initial content of GPD's PST set & ...
Also Some Early TSG's designed.
Earliest TSQ's: ANL: simple Algebra; But D-IND. So we want to ... Length of

.20 Dern of function that maps Q→A. We can use Lsrch.
Next maybe: Simple S-IND: Mainly using frequency model of induction — perhaps some sort of English tables?
Say Lap's rule of ... ≈ Z141. or ≈ AZ141. From Ris? it may be poss!. to Improve GPD.

Next: More general solns. of 4141.8 — Nonadaptive Lsrch / now conscientiously cost more & more.
Say Simple vme.
Discussn. of "Scaling" 3 kinds of probs: 1) solvd by TSQs. 2) by Context dependent ... concs,
Chunking models corpus & as ... R.

.28 3) By Partitioning corpus / statistical factory. ("Indexing" of Q's helps TM lrn "partitioning").

.30 Go back to earlier reviews: Perhaps Augment .06–.28
Some reviews 512, 515 || 552 on proof (...) A || Won 538A || 532.20 Scaling || GA 517 / ... D-IND 614.09
BGN 602.ff || Chaos 599.12 || [592–593 has lots of Good Review Stuff] ... 583.00 ... better more detail
The last 583.00–.06 clarify General method of operation: 592.20–593.40 ; 601.35–602.04

5/9/02
IQ

'622

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00:621.40 (593.29 Spec) SN on kinds of S-funcs : There are methods of making "vectors" out of text, to compare texts samples for "similarity": The cut text up into sequences of say, 3 or 4 characters, — T. resultant vector is than dot products w vector from a different data set to compare the data sets.

A (perhaps related) method is used to somehow evaluate the "Goodness" of an "English Composition" for Grading College Tests : Essay type Questions

for mappings from Text to vectors or reals (or complex nos!)

perhaps Generalize these kinds of mappings

SN2 Also note use of complex (etc.) wts for certain ANN problems. I saw a book about this. QUATERIONS Hypercomplex nos; Base 4 dimensional things. Hurry note about: perhaps possible to use other vector wts.

.10 SN3 Instead of "updating" t. GPD; If we consider that it is a Rec. Bnd. P.D., then we would "update it" every time we use it! T. GPD itself would be defined by t. reference apriori (or umc) plus t. "Corpus" on which t. GPD is based.

SN In t. 2 stage method of Generating pc that a cand will have max G : This may take so long. But a different, cross Approach to this ideal P.D. might be used! E.g. using just successful cands as Corpus!

.18 SN On ALP v.s. Ockam : ALP: Before seeing t. data: we have 2 proposals for prediction technique: P_1 & P_2. We run P_1 & P_2 on t. data! P_1 has a small m error on t. data than P_2: we then expect P_1 to get better future error than P_2.

.20 That shorter codes for same .18–.20 is essentially what t. convergence Thm. implies — "apriori" imply better future predn. (Not entirely clear!).

.22 T. arguments of .18–.22 has to be made clearer. .18–.20 is correct, but its relation

.24 to .21–.22 is not "compelling" ⊙. (28)

.25: 620.40 Spec : After "reversing" we only have one code for t. corpus, yet t. pc's of t. (from a declined) symbols are all based on many ((codes! — ∴ many possible parses. How to deal w. this?

.28 (.24) .18–.20 is correct, but .20–.22 is not t. same as t. "conv. Thm." Say we both contenders agreed to use t. same umc (= apriori), but each contender found different regys in t. corpus. I'm assuming t. set of regys + t. umc amounts to an apri choice of apriori. [Chosing a umc w.o. giving CB or having or regys, amounts to no statement of apriori] . What we are interested in, is not Sol 78T3, But 575T2, corollary, which does not mention umcs.]

Dat So t. corollary (which I will call "FTM Fine's Theorem") would say that t. pc of t. corpus wrt a param, held by (pc from) is a measure of its expected res (or K-L measure) of its errors on a ner known corpus (in t. past). T. only Q is "Can we legitly include t. factor pc(param) in t. pc of t. corpus?" It is clear that some such factor is needed: otherwise t. ML Pd. that gives pc = 1 to t. corpus would win!

5/11/02 622

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

00:622.40: In order to see a logit a priori choice of Pam's the wt. pc(Pam) wrt some time, seems like a reasonable choice.

- One way to Look at it: Each contender has an a priori d.f. to help evaluate tools, i.e. a set of tools to search for good cands. Indep. of the above, if a contender under these conditions comes up w. a model, then pc(Pam) pc_{Pam}(data) is the correct P.D. i.e. "Finds Trm" applies. Def [FTM] = "Finds Trm" = Sol7STs(?) category

The "Universal P.d." comes in because a person in deciding on a personal a priori — would tend to use a Universal pd in (deciding on a vouching) his own personal a priori.

.09 T. proof args. are "Not Bad" but still need to be Tightened up, evaluated, criticized, Proofread, Debugged!

10 ON "Updating": GPD can be updated after each problem soln., or, it can be updated before &/or during each problem soln. In this latter case, TM can decide on what parts of the GPD to update: Also it can update wrt previous problems, if it feels it is useful.

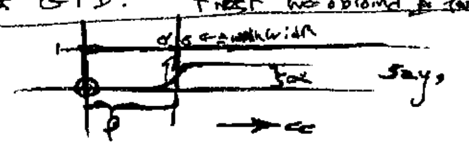
So in general, the "just a time" spent on updating is rather unclear. Ideally, it should be decided on by TM, somehow, on the basis of "Previous Experience": But, in general, this is the information → 624.10

.15

.16 [on WON 542.40] → [SN WON] On the probable ordering of PST's or other Cand types to be "Lsrchd on" } Not So! See .24 No So!

.18 I had this 2 step updating of GPD. First we obtained the fixed c.c. the P.D. on G of each PST_i = $P_i(G)$.

20



Say, ↑ prob of soln. for cc = cc

well, I think to compare 2 PST's, comparing α_i & α_j would be usually decisive: this is prob of soln at cc → ∞.

.24 ! On second thot, I'm not thinking about PST's in general, but maybe INV problems! — and this looks like WON type curves! In this case comparing α_i w. α_j is correct. If they are ≈, then compare β_i w. β_j. If they are ≈, I don't know if $\alpha_i \neq \alpha_j$ makes any difference.

Actually, comparing α_i & α_j may not be so simple, since usually α is not known

30 exactly, by any means!

Usually, the way INV problems are solved is by hill climbing or GPS (Vector Hill Ht.). At each time, for each PST, one has some idea how close one is to the Goal. — This "Measure" can be used to compare "where one is" "how far one has gotten" in any of the PST's being pursued in ||.

.36 So, as one works on each INV problem, one has a P(G) curve for the future, that is continually changing. A Q is: do they ever look like the curves of [.20?] → Yes! See 624.00-.04

.38 Perhaps one has in mind (at each time) (a) Will this technique (PST) ever solve it? — The prob of

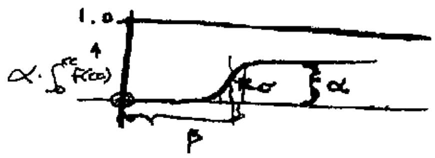
.40 Soln. ever (b) If so, how long will it take? So maybe we want to

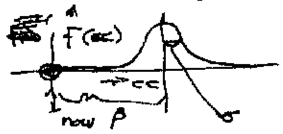
5/11/02 ID ID ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 : 623.40 product of these 2. 'I first part is a scalar betw 0,1'; + second is a P.d. that $\int^{\infty} = 1$.

.01 If α is t. first factor & t. product $\int^{\infty} = \alpha$ F(cc) $\int_0^{\infty} = 1$





.04 So [623.39 - .40] is ≈ [623.20]

10 kv. 3 uf / 1 us = 30 k Amps
200 cell
cal = 4 pts
3.62 liter
720 liters
2,700000
2.7 M
.720 k mA / 2.7 K h
.08 / .3 = .27 ml/drk

It may be poss. to solve this problem more or less exactly for certain curves of t. forms .01 ≡ .04.

Whether curve .04, or t. "S" curve ($\int S \, dcc = e^{-\int F(cc)\,dcc}$) should be used in making cc allocations among PST's is unclear. [Right S curve becomes relevant in computing

.10 : 623.15 of GPD being a "Resource Bnd PD". So one has to state how much cc one has available before it is meaningful to ask for an output value! I like this view. It does take into account t. amt. of cc needed to use a P.D., & in perhaps t. most General way poss'l.

It does sort of simplify t. dscn of how TM works": [when Given a problem] we just use GPD to decide what PST to use. Normally well: Maybe not so simple if updating occurs along w. prob solving!

So perhaps any PST (if it uses GPD) has to tell how much cc to spend on "updating" GPD & when to spend that cc. ⇒ Perhaps a PST would be "selfish" & not want to spend cc on [updating GPD in ways that] deciding whether TM should switch PSTs! → 630.11

.20

.21 So: There are (at least) 2 items of poss'l (≡ Alternative forms/choices) Modify in System:

1) Doing regular 2 stage updating of GPD : Or using simpler, not nearly so good updating algm.

2) Doing Update before each PST a/o during; Controlled by PST; Vs.
" " after (or before) each PST or continuously during, 50% of time: (improvements)

Chose simplest one for Expo, but mention poss'l. Variations / improvements.

? → A "Big" "impt" way Present System differs from S86: [In S86], S86, TM would ask for best OT for OZ problems only (most problems). Now TM asks for best PST for all problems.

30 .... (Since almost all problems are ≈OZ anyway, this isn't a Big Difference — but my picture of how TM works may be somewhat different.

So t. main Difference from S86, is that I've Gone into more detail on how to update various parts of GPD — partly by using Mxd Corps Term also other ideas on forms of inductive functions.
(S-functs v.s D functs).

Another Difference : Present TM doesn't ever do "Self improvement" as a "special problem". Updating means "improving t. GPD" & this can include any conceivable form of SI, since a PST can be any thing — including a new global system for TM

5/12/02
ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

Wolfram wrote about these — TM — 40 : 330 P

00 : 624.40 . [SN] on "Intractable Sequences": ~~[illegible]~~ Presumably sequences for which the way to compute the N^{th} bit takes a certain $F(N)$ of ≤ (say $F(N)$ is linear in N): The idea is that there is a minimal way to compute the N^{th} bit & one can't do it any faster. Furthermore, it may be that most computable sequences (i.e. seqs w. finite [illegible]) will have a min cc that is easy to compute. [illegible] the value of i.e. All Most all computable seqs, to compute N^{th} bit takes the same time as computing first N bits.

— That there is no "shortcut" to get the n^{th} bit, e.g. for π, it is perhaps necessary to compute the first $n-1$ bits, before the n^{th} can be computed.

The (argument/proof) would be n to the proof that there can't be using "Non-random" numbers.

It isn't yet clear to me how to show what I want: Say $F(\alpha(\cdot), n)$ is the min cc to compute $\alpha(n)$ (= the n^{th} bit of the seq. defined by α (a [illegible])). So $F(\alpha(\cdot), n) = F(\alpha(\cdot), n-1) + g(n)$, where $g(n)$ is a nonneg funct of n. Is it clear that there are any, ~~[illegible]~~ a few, many, very many, functs ~~[illegible]~~ like $\alpha(\cdot)$ that have this property.

Consider $\alpha(x) = x!$: we could calculate $x!$ from $(x-1)!$, but there are approxs, that are e.g. much faster. $\frac{x^{x+\frac{1}{2}}}{e^x} \cdot 2\pi \cdot e^{\frac{1}{12x}}$?

If there is a proof in this area, It will prob'ly involve a "counting Argument" (like is used to show almost all nos. are "Random". "Random".

5/14/02
ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 : 624.40: The idea is that the System is "Very Simple": We have a Cond'l P.D. & a set of PST's. They are used to solve problems. The updating of the (cond'l) GPD accumulates knowledge, as does the expansion of the set of PST's & their probabilistic associations w. various problems. For ME — For my own information: that it seems to follow how I think & solve problems w. easy transfer/to larger TSQ's.

I will have to make it clear as to what the cond'l P.d. is; & in particular what the GPD is, how it is used / what it actually means: & how it is able to integrate/synthesize different areas of research/inquiry (also aspects of its own operation).

An advanced concept is the idea of "Resource Limited P.D." & how it incorporates updating into its "Normal Operation".

So, draw picture of TM in "Middle age", steady state:

Then describe early TM, then evolution of it: modifns in the PST set & in how good the GPD is. Then discuss Mature TM that has effectively modified (optimized) the method of search & General operation. 592.17 – 593.40 } Has much Good Stuff, Review of this stuff. → 628.00

[SN I had trouble getting a pure QA TM to understand what optzn problems were. The way this is solved is by optzn problems automatically using the 2 step process for updating GPD.

So TM doesn't ever "understand" what an optzn problem is; we use its (second step as) optzn scheme in a special way to get update for optzn.

Q: Could we devise a "special update" for "L.A." (Lookahead)? More exactly, is there any xtnn of the normal GPD. Well, in the "Cure Cancer" problem, It might be possible to simulate the way humans seem to design "Discovery Research" PGMS. —

e.g.: Making Rsch subgoal of "Understanding Mechanisms of Cancer". Understanding how Cancer Cells differ from non-Cancer Cells.

After a lot of Q's of this sort have been answered, we can begin to ask how to destroy, slow down, ... cancer cells & leave others intact — or somehow Get rid of effects of cancer w.o. otherwise hurting person too much.

5/17/02 FD

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

On Grammar Discovery:

For Discovery of the type of Grammars discovered by ZIFF! Perhaps best to consider the "words" = "sub words" [≡ "Ntups"] that have been discovered so far, and to have only tentative PC's.

When we get a chunk of corpus, we try to use these "Ntups" to parse that chunk, and use the usual LZ's rule to get a tentative pc for that chunk: we then multiply it by the costs of defining the various Ntups; for this last, there may be several ways to define them, a network of definitions. The overall desired pc of definition is the sum of those, the pc's of each of the defin. techniques. — but we will (usually) be mainly interested in single methods if hy pc (unless we find a very hy population bunch of definition methods — "degenerate" methods of defn. These include trivial variations: e.g.

AB → C; DE → F   AB DE → CF; but we can also concatenate) so pc + 2×pc.
DE → F; AB → C   AB DE → CF

CFG discovery: "A new method for discovery of Grammars of CFL's" ~$, 60.

To discover the Palindromic structures in a corpus: look for correlations between the corpus & the corpus written Backward.) So our new Macro Corpus consists of the same original Corpus String, & that String written Backwards. We look for common ngrams in the Macro Corpus. When we make a tentative def. of an Nbop defn, we reparse the corpus! (& its reverse is automatically reparsed)

While this approach would be fine for many CFL's, it would probably not work for English text. Perhaps try it on "Fortran" or other Computer Languages that are CFL's.

→ My impression was that "Ngmsets" was a useful conc. ← I think this is V.G.

E.g. Grammar rule $A \to \begin{pmatrix} C \\ D \\ E \end{pmatrix}$: The set (CDE) could appear in other parts of the Grammar.

or $A \to \begin{pmatrix} C \\ D \\ E \end{pmatrix} \begin{pmatrix} F \\ G \\ H \end{pmatrix}$: So we want to find a pair of ngmsets, α (= $\begin{smallmatrix} C \\ D \\ E \end{smallmatrix}$ say) & β (= $\begin{smallmatrix} F \\ G \\ H \end{smallmatrix}$, say)

∋ $\alpha\beta$ is "maximally predictive" — This means we have constraints on the largeness & smallness of the sets α, β. If α & β are very small (cardinality) they have small ssz, but predict well, when they occur. If α & β have large Cardinality they occur, have large ssz but predict less exactly.

The ngmset data cons was a principal heuristic in SOLSB (an ind. Machine).
After getting a few Ngmsets & good ngms, we can try concat rules, & addition rules.
Additions: Boolean addition of several ngmsets. We see if they can ↑ pc of codes of corpus.
Similarly w. concat of ngmsets. : to see if the concat is useful to ↑ pc of codes of corpus.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 (spec 626.26) : I think t. main problem in writing TSQ's, is t. find ways to get TM to acquire useful concs.

I have (will have) this sequence of stages in TM's development: Better & better PST's; How to get it to acquire various hacks

I did have that big problem of getting TM to acquire t. idea of a "quantity": My ideas on what this actually was were unclear. Better: Decide how TM should behave if it had acquired this conc.

So, we get TM to learn what ev(3+7) means (i.e. "eval").

So it must learn that ev(3+el(8-2)) = ev(3+(8-2)) (transcaple)

But many other examples.

.10 TM search t. "Eval" function in LISP, but it doesn't know that "3+7" is somehow closely assoc w. eval(sum 3,7) . ; Tho it may have learned how to solve "3+7=?"

does it automatically since PST's that use t. "eval" function calculation gets more G per unit time.

Well then, is TM done more or less finished?!

.17 Write down just what needs to be done! → 592.17-.40, 593.10-.40 } has Good Rcvr of Some of this

.18 SN Re: t. 3 input unc as a form of Universal P.D. I work out a method to use A2:141 (≈ LISP) → 447.224 To realize such functions: I had the idea that t. method I proposed was

.20 Very inefficient : I. idea was that One pattern a bit: A2:141 gives an output: If t. output's initial bit is Φ, then that means t. rest of t. output (if any) should be disregarded & we should concat a new bit unto t. old, & input. If t. first bit of output tells if we need another bit in put. ... When t. first output bit is 1, then t. rest of t. output is "t. actual output".

In fact, this system may not be so inefficient: we just look at t. first output bit & given another input bit if t. first output bit was Φ. We don't have to actually look at t. rest of t. output if t. first output bit is Φ: → (see FN(.00-.33)R ← This seems to solve t. problem! >

Trouble is, we only save time in executing t. final output function of A2:141....

All of t. calcns. before that take an amount of time that is indep. of t. bit that we are (usually) only interested in: t. first bit of t. output.

.30 • IS there some way we could teach TM to somehow only spend time on t. first output bit, if it were to be Φ ?!? This is a More General Problem in TM system → YES! IS Automatic! → We want TM to be able to do (w/ this kind of action) spend time on t. that calculation rapidly; will get a lot more PC (in GPD).

.31 .32 • I also must write up arguments as to (just why / all both) I use t. 2 stage GPD when looking for an optimum Soln (I do this for both) INV's often. problems.)

.33 .17 In Infant TM I may not do it because GPD is not good enuf to make this a useful technique.

→ 629.11

.37 • FN(.00-.33)R is an impt case at which this "trim" is important.

.38 Re: Design & Integration implied by t. 2 stage GPD & of .33: If I assume t. first stage dist. only has 3 parzens:  ∫ω(·) = Φ, say.

Then I can make a tables of pairs, say; But more generally, if I assume that two combination

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

00:62840 of Two cols (first-order U.F.'s) yields a first-order d.f. w.r. α, β, β that is a func of the $(\alpha, \beta's)$'s of the combined d.f.'s, then I can combine any number of them.

But look at the actual integrals; there may be an easy way to do integration by parts, th some trick, because there is a certain function product of d.f.'s of each of the PST's): we divide by f_i & integrate something dependent on t_i.

Say $(\Delta t \cdot f_i(t))$ is the p.c. that PST$_i$ will solve this problem in time $(t, t+\Delta t)$ — so $f_i(t)$ is a prob density.

[Note that $\frac{dx}{x}$ is the derivative of $\ln x$.

$d\,xy = x\,dy + y\,dx$

$xy = \int x\,dy + \int y\,dx$

10 But put this on the stack (P611#9)

11:62837 So: Perhaps Abstract/intrdn of Report:

We report the continuation of research on machine learning Sol 56,57, 62, 86, 93, 94. The procedure model we are developing is close to that of Sol86. The system starts with an initial set of "primitives" (the universal) instructions. We initially present it with a small set of problems, which it solves easily, using Levin's Universal Search on combinations of its primitives. The search is initially guided by a universal probability distribution induced by the initial primitive instructions. After each of the first problems are solved, the initial probability distribution is updated, so as to speed up (post hoc) the solutions of the problems solved. This is done in an optimum way. Algorithmic Probability tells us how to do this in an optimum manner.

Having solved the first set of problems. After updating, the system is able to solve more difficult problems. After the problems are solved, we again update the probability distribution. This problem solving and updating are continued for problems of increasing difficulty. We expect that the system can be brought to a high degree of problem solving skill, using this learning to, I hope, a suitably designated training sequence of problems.

In a more advanced system, the updating will be done more frequently, possibly at the same time as the problem solving — using either parallel computers or time sharing on a single computer.

Bionic 8.

$p = 5.85$

$L_0: 8.92$

$AL = 2.7$

$5.6 \approx ?$

$S_n: 5.95$

2.29

$\#\ 7.44$

2^n

$p \approx 6.07$

5/27/02
ID

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00:629.40: Plan of Report: 1) Abstract: a summary of entire:
Intro: Tells what kinds of probs solved; Gives motivation for this project as "A.I." [Maybe why not a machine that one can talk to "as a friend".]

Give order by list:
Discuss form of soln of induction probs (Q A probs): Soln is P.D.
How GPD is "Like Google" [Great variety of Q A types]
" " " Unlike (a) paths (b) kinds of Q's that Google does get involved in — "What's a best PST for this problem": (c) "I am trying to integrate this expression: what would be a good substitution to make?" (d) What should I do to simplify this expression?

.11:624.20 [SN] I had the idea that a RLPD is never "updated": that using the RLPD automatically updates it: that "updating" is part of the process of using the RLPD: Hur. not so clearly true!
Unless we understand that using the RLPD automatically modifies it, (when the updating occurs)
Before, I was worried that this view of RLPD's would make updating during probsolving impossible! — but not so: the PST can determine when (& how much time to use in simulations) & how much time to spend on the update.

But say we just use GPD as a UMC & look for short codes for a particular SubCorpus.... Just how does this "Update" GPD? Well, if we found short codes for 414.18 we could certainly use these codes for induction — as part of "updated" GPD.
Actually (I conjecture) we use GPD, we either use short codes that were found in the past, and/or find new short codes (≡ update). This would be true if we always used GPD as a (Q A) machine — which I guess we always do — its a Condl PD. —
So we always use 414.18 to update it. — Strictly, we can use it w.o. updating it, by using the s-functs that it already has. — But normally we will want to integrate into it the problems solved after the last update — i.e. the actual Qi, Ai pairs that occurred: The Augmentation of the Corpus.

One Big Q was how 414.18 was "updated", in the sense of modifying the previously found "soln." (≡ S-funct). I.e. how to update 414.18 w.r.t. an Augmentation of the Corpus? Well: (not using the simple umc model, but continuous param model of S-functs)
(First) we use same old S-functs but update simple continuous params like pc's of concs. (This is like updating of A≥141 but note 620.22ff, 620.38 in particular....
(Maybe Serious Difty!)
Third? (Second) we define new concs that are now "above threshold".
and Second (Third) we may reparse the Corpus (in view of new pc's) — then update pc's if reparsed.
Also, maybe undefine some concs & define new ones. (undefining shouldn't be done much — just for recently defined concs — or concs that are not part of other defns.

620.22ff on "Big" in A≥141 use, updating.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 : 630.40 : T. "update techniques" of 630.30ff: I've ~~forgotten~~ how this is done: I must find references
2/o new analysis. In AZ141: as ssz of corpus, we can afford to define new concs.
Reparsing, hvr, is within t. function (or function set) that implement t. predictive d funct or S funct.
In t. case of ~~[crossed out]~~ pure D prediction, only t. pcd t. predn function must be f, since t. corpus pc = 1 always.

.08 [SN] on Updating: T. formulas used in Time series prodn. are in ≈ QA form! They
Express t. output Df. as a function of k previous ~~[crossed out]~~ inputs. That t. input sets
Overlap .... is this info used in anyway? — for linear regression, it simplifies
t calcn. a lot. T. imptc. of this is that I had been having trouble finding examples
.10 of stochastic (QA): Actually Curve fitting is another rather general example.

.11 For S-functs on Strings: If we have a "d-funct" that is almost always
right — say it's right (1−ε) of t. time) — then we can code it as that
"d-function plus" "corrections" for n Q.A.'s, t. pc is $(1-\epsilon)^n$ mult by
cost of corrections: Each correction cost ε mult by cost of some of t. A: The
~~[crossed out]~~ Corrections themselves form a Bern Sequ. & can be coded
.18 as such. →

.19 A genz. of forgg.: we have k distinct d-functs to do the corpus of QA's.
.20 → [None of them are right all t. time, hvr] (tho we may have some that are always correct but they
have low apnes!] Anyway, we code t. corpus as a Bern seq. using t. k d-functs.
.24 If 2 or more d functs agree on an A, we add their pc's! If all agree t. pc = 1 for that A. (632.00)

.25 [SN: Admin] Perhaps keep a file of references to [ways to realize S-functs.]
631.04, .11, .19 ; [632.18 − 633.?? : On How to implement t [3 input univ Model]] ← But note Q of 638.06R
636.03: How to Norz.
[See 593.17−.20 for a list of 7 formal S-functs.] 497.22 original 3 input univ Simulator viz AZ141

.30

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 (spec .24 : 631.24) : Hvr. 631.19 does sound [pc] exponsive / [→ But note .02 ↓] We have to code each of the d-functs. Because its a Bern seq, we don't have to say what the pc's [pc's expense] of the individual symbols are.

.02 The d-functs can share sub-codes. — so if [redacted] the s-functs are similar to each other — not so expensive. If several d-functs give correct predn, then we have parallel codes: so "doubling" ["multiplying"] pc for each such occurrence.

.05 : 631.18 → Another way to get "s-functs" ≈ the $(1-\epsilon)$ thing of 631.11–.18: For each Q, we have an s-funct that gives an A. When there is an error; say A_0 is the answer from the d-funct: we then use a standard $P(A|A_0)$ for the other possys. This is "standard" in the sense that $P(\cdot|\cdot)$ is indep of Q — it depends on A_0 only. How to get $P(A|A_0)$ is unclear; there is a "standard" ALP method. $P(A|A_0) = \sum 2^{-\ell(\text{code for } A,\ \text{given a code for } A_0)}$ summed over codes for A_0 [For given a short code for A_0, to obtain A from it.] since A are

.10 always harder, these must be self-del (prefix) codes.

→ A practical way to realize $P(A|A_0)$ is not apparent.

.15 Another way to think about $P(A|Q)$: First we have from Q a partial code. To obtain the rest of the code, we have to go to the "R" input. Q can actually get a self-del output. If then concat on the R input, to obtain codes for A's. → How to do this all is not clear → 634.12

.17

.18 **A Better way to think about s-functs via 3 inputs used!** ← Seems to WORK FINE! Use A 2141 (≈ Lisp)!

The 3 inputs are ① defn. of P ② Q ③ R: Hvr, P, Q & R are all self-delimited

.20 finite strings — normal Lisp inputs. The length of R (or $2^{-\text{length of } R}$) gives pc of output.

Since R are self-dels they form a prefix code.

A common way to get self-del codes is w. "end" symbol. I think I analyzed this some time ago!

A problem was the pc = $3^{-\text{code length}}$ ... since there are 3 symbols.

I think I did it this way: the symbol pc's were $\frac{1}{2}-\frac{1}{2}\epsilon$, $\frac{1}{2}-\frac{1}{2}\epsilon$, ϵ. 3 → 3.375, 4 → 3.16, 5 → 3.05

A string w. $\frac{1}{\epsilon}$ o's & 1's has pc $\frac{1}{2}(1-\epsilon)^{\frac{1}{\epsilon}}\cdot\epsilon \approx \frac{1}{e}\cdot\frac{1}{2}^{\frac{1}{\epsilon}}\cdot\epsilon$.

We can use many codes in ||, each w. its own ϵ — we integrate over all of them.

$2^{-L}(1-\epsilon)^{L}\cdot\epsilon = 2^{-L}\cdot\epsilon\cdot(1-\epsilon)^{\frac{1}{\epsilon}\cdot L\epsilon} = 2^{-L}\cdot\frac{\epsilon}{e}\cdot e^{L\cdot\epsilon}$. | max $\epsilon e^{L\epsilon}$ = max $L\epsilon e^{L\epsilon}$, $L\epsilon = 1$?

.30 $(1-\epsilon)^L = (1-\epsilon^{\frac{1}{\epsilon}})^L \approx e^{-L}$ $\quad \approx 2^{-L}\cdot\frac{1}{L}\cdot\frac{1}{e}$ $\quad xe^x$, $e^x + xe^x$, $(1+x)e^x = 0$, $x = -1$, so $\epsilon = -L$ as expected

L	$(1-\epsilon^{\frac{1}{\epsilon}})^L$
3	3.375
4	3.16
5	3.05
∞	2.71828

This gives the "peak" for ϵ, summing over all ϵ should give larger.

Actually, in spite of the "apparent" constraints on 0, 1, Δ [Δ = end symbol], when we look at the R's as a sequence of QA's: it is an ordinary 3 symbol Bernoulli.

If the mode length of R is L then Δ occurs w. frequency $\frac{1}{L}$ in the Bern seq of alphabet 0, 1, Δ.

.27 The pc of the R sequence is old Bern: $\frac{n_0!\,n_1!\,n_\Delta!}{(n_0+n_1+n_\Delta+3-1)!} = pc$. [3 = no. of symbol types.]

0, 1, Δ → 633.00

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

00:632.40: So this seems to solve one of the very Big problems! How to implement the 3 input func p.d.!

We use 632.18-.20, then the pc of any R_i will be $\approx \left(\frac{1}{2}(1-\epsilon)\right)^{L_i} \cdot \epsilon$ L_i = length of R_i

$\epsilon \approx \left(\frac{1}{\text{mean length of all } R_i\text{'s}}\right)$: The exact expression for the entire set of R_i's is given by 632.27

So we really don't need that UTM simulation with its apparent inefficiency! We just have 2 inputs to the func that determ Q ∈ R is our output ∈ the output.

The Bern formula of 632.27 is equiv to (interchanging summing) overall poss'l. pc values for 0, 1, 2 Δ.

.08 If the length of an R_i is zero (null string), its pc is still $\epsilon \approx (\text{mean } |R_i|)^{-1}$.

If we want the most likely A to have pc > .9, we have to have >1 way to code it. If we want to get pc ≥ .1, then just about all R_i values would code for the same thing. — Say the machine

.10 would always give Ao as output — unless |R| was >10.

.11 A trouble w. the Borgo: In many cases, most of A_i's have pc's ≈ 1 — yet it takes many R trials

.12 to verify that each pc in 4.14.18 is indeed close to 1.

Well, if most pc's of A_i's are close to 1, then $\overline{|R_i|}$ is small — say ≈ 2. $\epsilon \sim \frac{1}{2}$ $P_1 \approx P_0 = \frac{1}{2}(1-\frac{1}{2}) = \frac{1}{4}$; $P_\epsilon = \frac{1}{2}$.

null: pc: $\frac{1}{2}$; 01, 10 → (0, 01, 10, 11) have ∈ pc = $\frac{3}{4}$: {00, 001, 010, 0001} = $4 \times \frac{1}{2} \times \frac{1}{16} = \frac{1}{8}$

so total of 0, 1, 2 length codes is $\frac{7}{8}$. 1, 2, 4 — so 7 codes → $\frac{7}{8}$; 15 codes → $\frac{15}{16}$

so for pc 1-δ, we need ≈ $\left(\frac{1}{\delta} - 1\right)$ codes.

If $\overline{|R_i|}$ is larger, I suspect we need even more codes to get total pc close to 1.

.20 Another possl. trouble: (I examined this before): One of the pc's will always be maximum ∈.

(This is the largest pc) If the needed pc's are all < ∈, we have no way to express some of them. Well, we could do it by semi measure: ∈ i.e. |R| = 0 gives no output.

We would have to renormalize.

AH! This may be a way to deal w. .12 as well! Say we have $P_{A_0} = .99$:

We let a code A_0 & the codes for $\Delta\binom{n}{1}^{n}$ don't code anything if n < 7, say.

A trouble is that "not coding for anything" usually means "non halting" — which is expensive to verify ∈ in L search. Could he use "null output" to be equiv. to "not meaning full output"?

No output "Null output" means Machine stopped w. no output. — Null output could take a short time,

.30 and TM would be "motivated" to arrange so that these Null outputs took little time! See arguments at 628.31-.32 and Note (628.00-.32) ff.

Still we would have to examine all these "Null output" codes to properly Normalize the output for R=Δ.

A big waste of time!

It may be that these difficulties are unavoidable: In a Universal D.F., when we get a certain output for a short code, the system has no way of knowing that there are no other (for present Q) short codes (of short length pc) that have output. (to some Extent)

The "Null output" situation corresponds to my previous 3 input func model, in which the func would choose which input sequences were (=0 or 1), by "asking for another bit" if they were illegal,

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 633.40 ; In the recent model, the system "prints" nulls stopped when an input was illegal (i.e. not part of the prefix set) so the previous & present models may not be so different! The previous model printed "φ" followed possibly by output, if the input was illegal.

In the recent model, the relative wts of input strings of length L was $(\frac{1}{2}(1-\epsilon))^L \cdot \epsilon$: The previous model had relative wts of $(\frac{1}{2})^L$. This does seem different! In recent model, the value of ε depended on $\overline{|R|}$ (mean R length) for the entire corpus. The old $(\frac{1}{2})^L$ had no such "Global involvement".

Hvr, the older Model did have a Normzn constant that had to be computed ; as does present model. The ε doesn't affect relative wts as it is fixed by Normzn constant.

.08 The main difrnce seems to be $(\frac{1}{2})^L$ v.s. $(\frac{1}{2}(1-\epsilon))^L$ — as relative wts of 2 certain R codes. The $(1-\epsilon)^L$ factor is a signift difrnce in the relative wts of 2

.10 R string, in the 2 systems. It would seem that this is a sig nift difrnce — so that one model

.11 would have to be wrong if the other is rite! ← Discussion on this: 634.11-.13 Sounds rite → 635.00 ; Z_Q is of some fairly length. "collit" Z_Q

.12 :632.17 — Dropping this for a while! → from 632.15-.17 The first output from Q (without R) is a "short code" for A_0 (the hypothetic Answer). To obtain A's "near" A_0, we use this short code plus other bits, to create various alternative A's.

$(Z_Q \frown R_i)$ operated on by some standard Algm. gives A_i.

$M_U(Z_Q, R_{ij}) = A_i$. (R_{ij} is the j^{th} R that gives A_i as output).

.19 Say Z'_Q has same (max) pc. as Z_Q. Then there is no way to get M_U to print Z'_Q ... R_{ij} < 1

.20 $\sum_j 2^{-R_{ij}}$ can't be 1 — if it is there can be no other legal A's — which is WRONG. !!

.19-.20 seems to demolish idea of 632.15-.17 — of a A_0 and other A's being various "information distances" from A_0.

Pilot V5. Process extra fine

Well, say A_0 has several codes — would that help? ?

Another possy: A_0 does have code Z_Q, but A_0' has code $Z'_Q \frown R$.

$|Z'_Q| < |Z_Q|$ $|Z'_Q \frown R_i| = |Z_Q|$.

Or: $Q \to Z$: $M_U(Z, \text{null}) \to$ no legal output.

.29 $M_U(Z, R_1) \to A_0$; $M_U(Z, R_2) \to A_0'$ $|R_1| = |R_2|$.

.30 NB These objections to various S-funct Models are impt — They should be telling me what Kinds of Models are Good! See

In using the 3 impt UMC model, which is Universal, But the commonest Models I use are :

.34 ① Bern seq. Z141 & AZ141

② Various real continuous linear & non-linear curve fit Models.

③ Maybe UMCs for d-functs for d-INDuction

④ Perhaps ≈ Bernoulli models of ③ (631.11) Maybe 631.19.)

⑤ See 593.17-.29 for 7 forms of S-functs.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: (spec 634.01) In general, the function (for single uses) must be able to decide which input R's are

.01 legal & which are not: {I think this is ney to get around the diffy of 633.20 (say we wanted 100 pc's of A's — all = $\frac{1}{100}$: — this would be impos. if we apriori had some codes w. pc's $> \frac{1}{100}$}. The older ("Give me one more bit or not") method & the newer method (all R inputs strings end in Δ) both selected out a subset of codes they decided to be legal" — but in different ways.

→ But whatever I do, if the machine has to decide on legality of an input: this has to take time.
In a proper training Envt., the machine will try to arrange codes so that total time spent is minimized (628.00–.32) a)
— Here, I'd like to arrange so that this "Learning" is as Easy as possl.!

The 2 methods assure Σ pc's < 1 by 2 diffnt means: The older method arranges so that the legal binary (or whatever) sequences are never prefixes of one another. The other uses the Δ symbol to terminate codes (so 3 symbols).

Using the "Older way": for R, we would feed it the binary sequences in Lexical order (= Numeric Order). If a sequence gives an output, we don't try extensions of that sequence.

((Since the set of codes is not normed, we really have to get "almost all" legal codes to ≃ normalize it! — which is a "Big Job". My Impression is that there is no way to avoid this!))

Because: 1) we need to use unnormed codes: This means we have to find the normzn const by finding almost all legal codes!

2) The reason we need unnormzd codes is 633.20 (≈ 635.01) — i.e. TM has to be allowed to decide which Codes should be Legal.

Normzn. is not easy: We really don't know how many legal codes to get before quiting. The main problem is 414.18: i.e. we need P(A₀|Q₀), say. We need at least 1 code for A₀. Presumably, we've tried all codes shorter than the one we've found for A₀. We could study how total pc f w. length of Code Considered — Then Guess at Assymtote!

Not normalizing doesn't seem to be a reasonable option: we have to balance the Normalized pc of the corpus v.s. the pc of the Model. The pc of the model is normed → (HA: How to Normz: 636.03)

On Second Thot, the foregoing Mess seems unreasonable! I certainly don't run into the normzn problem when I solve problems! It may be that the S-functs I use are almost all already normed. — (If this is possl., TM would try to use such functs because they are much less expensive (cc wise) than unnormzd S-functs.)

The reason may be that in science one normally uses prim. rec. functs. — There is no normzn problem — they are already normzd. Also, there is no halting problem (?).

So I will try to use Prim rec. functs almost all the time. When I do use partial rec. functs, it will be for very serious, diffclt. problems.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 : 635.40 ! I can probly get AZ141 to generate only prim. rec. funcs, but ~~The~~ I don't see how that gets rid of the normzn. problem. ! W. Prim Rec functs, we always know if an input gives an output — So we don't have to wait a long time for "Halting".

.03; 635.29 [N] [How to Normalize] Do [Random Trial Codes] Keep track of μ and σ^2 of Σ pc :

I think one should do trials w.o. replacement — so hy pc trials will be repeated. This is a waste of time, so I should find way to count such trials w.o. excessive Room. Hvr, this Monte Carlo Normzn. does need a lot of Thot! If I use the early #3 input enc model, I can't just try any string as an input — it may seem like a legal string, yet have a prefix that's legal — (so it couldn't be legal itself).

10 — Int. more recent 3 input enc (w. all binary strings = legal since they end in a, ~~stop~~ "stop" symbol.) — but

.11 I was wary of this model because of $\frac{1}{2}^L$ v.s. $(\frac{1}{2}(1-\epsilon))^L$ (634.08) — on second thots this may not be so bad! T. 2 systems could end up giving the same Pd's anyway! — They will have a different set of codes to express the same PD.

.13 To continue .10 : the most recent 3 input system. Here, all finite binary strings are possible codes : for Mt. Carlo trials: spend ≈ time on codes of length n, for each n. ~~Equivalently~~. For each n, total wt. is $2^n \cdot 2^{-n} \cdot (1-\epsilon)^n \cdot \epsilon \propto (1-\epsilon)^n$ — Unfortly, I don't know ϵ. ~~The~~ Since total wt. of all strings of length n is $(1-\epsilon)^n$, I have to · (Assume I do know ϵ.)

20 — General problem: T. interval 0,1 is divided into cells of length : p_i $i = 1 / \infty$, $\Sigma p_i = 1$. Some of cells are black, others white. I want to estimate Σp_i of all black cells.

.22 — Say I randomly chose a pt. betw. 0,1 & note color of corresp. cell. If the pc of choice of a cell is $\propto$ its p_i only, then I think it will work.

.26 (So: first chose an integer, n betw. 1 & ∞; probty of choice of n is $(1-\epsilon)^n$. Next, chose a n bit string at random : that's it!

.28 One way to do .26: probly of ϕ is ϵ, probty of 1 is $(1-\epsilon)$: do trials until ϕ occurs; # trials gives n.

.29 With each trial of .28, flip a coin to get a bit of the K (=n) bitstring.

30 .22 – .29 will give the fraction of codes that are "defined" v.s. "undefined". ("Undefined" means "null output". — So we have Normzn. by keeping track of n & ϵ, we we have [redacted] amount of uncertainty in this. T. Q is : How expensive (cc) is this?

Note: Use Lsrch CB'on trials. Say T < 2T if repeated error is not small enuf.

.33 Probly a good Method : Do all short codes systematically first, — say up to n = 5 : then do the Monte Carlo search on the rest. ~~The~~ I can use random choice w.o. remembering which cells were visited, since revisit is unlikely enuf to make it worth the bookkeeping to Remember! ABCDEfghijklmnop

[DROP] This for a while! There is a "Obit" of Peter Elias in IT ~~news~~ news letter; Refers to methods of coding that seem relevant to what I'm working on NOW! → (637.00)

5/25/02

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 : 636.40 : The Elias Obits # IES IT newsletter Mar 2002;
I have all these Docs on CDROM. (also most in Hard copy)

.02 Also note: I have Book: Probability Similarity Networks: Heckerman: This is useful one kind of [i.e. BBN]
BBN for decision making (Medical) — Has lots of Refs → Appendix A: (pp 163 ff is Best Demo of System!)
Also Note! The (1986 Book) (Uncertainty in AI) that has my 1986 paper has Good discussions (Pro & Con) of
Fuzzy analysis by Zadeh v.s. Cheeseman.

The discn. of 635.00 ff is applicable to t simple unc. w. all binary codes (= 632.18 ff) legal,
using Δ as stop symbol w. pc = ε. [struck] Since ε isn't exactly known
until we code t. entire corpus (Prob we can approximate ε earlier in t. coding)
.10 we don't know t. exact pc of any CRA until the end ... which
is o.k.
Also, t. fact that t. earlier 3 input unc. method for $pc = 2^{-\ell r}$ v.s.
$2^{-\ell r}(1-\varepsilon)^{\ell r}\varepsilon$ for t. newer method, does not seem to be impt. See 636.11–13
for Counter Argt. → [638.10–.16] suggests t. older [447.22] 3 input unc. may be fine! — (or good const)
So 632.18 ff will probly work. It is a universal d.f. Hvr, most of t. time, I will
.17 be using Simpler kinds of S funct like 634.34 ff also 593.17–.29. → (638.11) IMPT!

.20 : ! Back to 629.11 ff: 630.00 ff [Some discn of updating problems: 630.20 ff]
<In general t. update method will depend on Just what S-funct is usd.>
For London talk (& maybe Report) discuss Heckermans (02) analysis of Probly in A.I. (Ibid. p(1–3)) ← But Note this was up to 1990 only. Perhaps Much more work on Probty in A.I. since then.
i.e. What is "New" about My Approach.?
.23 I May want to discuss "Normal Problem Solving".
a) Given Problem : Chose method to solve it. We do this on basis of our experience.
b) Solving it may involve Choices. — Made on basis of experience.
GPD will do a) & also b)—
Also as (sort of) part of GPD, it will [redacted] invent new PST's, & evaluate their
.30 effectiveness's for various kinds of Problems

One of Main Probs is updating GPD, in view of its knowledge of solns to new problems. ?!
The basis for updating all first order GPD is 414.18, (Since first order GPD is a cond'l. P.D, 414.18 always appropriate)
Again t. exact method of updating will depend much on what S-funcs are
used.
Note: 558.04 : Lists kinds of probs Solved by TM in a crossed way 557.27 ff. (536.35 Ans TM probs)
How to write TSQ's: (521.26 / 520.00)

5/27/02 638

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00:639.40 [SN] On OZ problems: T. CB for such problems must be from: for || Lsrch, we just continue working until the trainer says "stop". No and up spending pot of time on the open market, Opts.

.02 I did work on this some time ago. A simple formalzn of the problem yielded a reasonable result. I scanned 578 to 558: so far found 534.28ff Mildly relevant: (Got "533-1 dQ)

[Scan 539 to present] → [538.10-.11] was the dffy Referred to (.20)

.06 Anyway: T. idea was that I was, in Lsrch, spending less time on "less likely" cands., so, if I was interested in the cand that would be most likely to be "Best" for t. Time spent on it, that this would bias the srch toward hy pc cands excessively: also

.09 T. meaning of the pc's of the cands was unclear. (.20)

[.06R | Q: can I show that for my pc, [.7] at least over 633 636.05 Normz which will do it?]

.10: 637.17 Re: Normzn: ① Note Q of (.08R) recent ② Is [414.18] correct? T. pc's are "semi measures" i.e. unnormzd? ③ using 6-/3 input UMC of 632.18 ff, TM doesn't know what is when it's making cdfs for a A's to a Q. Later Q's can change c!

→ Re ② I think 414.18 works for unnormed (Butcheck this) pc's as well! If so, I could use t. old [(447.22)] 3 input UMC Model, then normalize when we actually wanted a A for a Q. I could use that Mt Carlo Normzn • legal inputs. In using 447.22: Just put the inputs in Lex. order — Noting which are legal, & not continuing any. This would be relatively fast & Good & easily Normzd. via Montecarlo. Plus 636.33 — But 447.22 normzn may not be so easy!

.16 [SN] 582.00 ff to ~593.40 has lots of Good Review Material. continuing

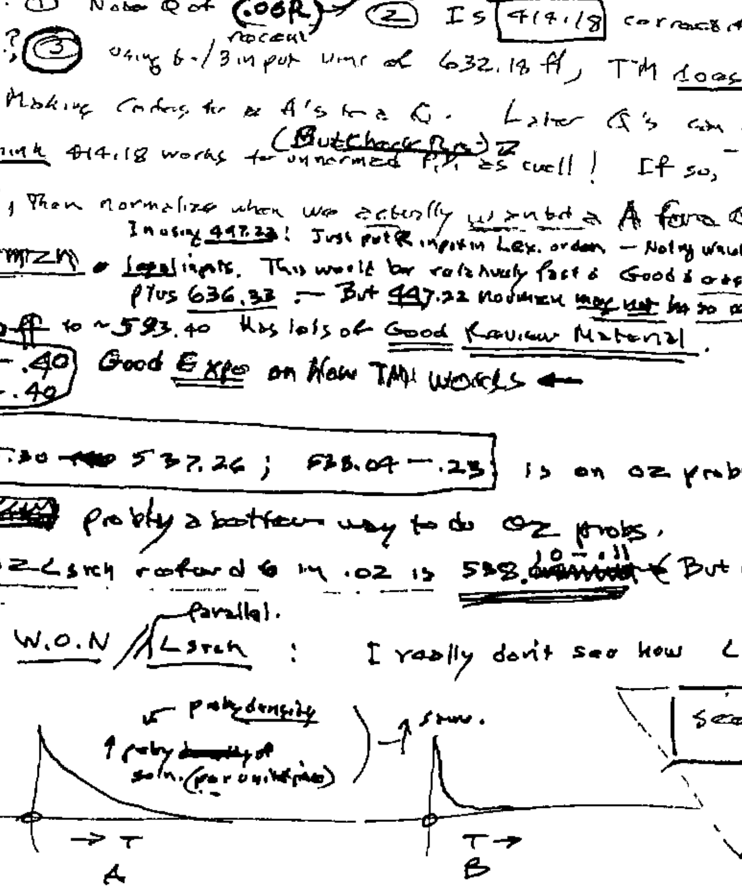
(518.29 — .40) Good Expo on How TM works ←
(518.29 — .40)

.20 (.09) ← [535.30 — 537.26; 538.04 — .23] 538.27 is on OZ probs: from there, it gets into WON as [illegible] probly a better way to do OZ probs.

The diffy in OZ Lsrch referred to in .02 is 538.10-.11 (But Note 639.24!)

.23 [SN] On W.O.N // parallel Lsrch: I really don't see how Lsrch could ever be done by WON:

One possy



[See 539.32 for the (.30) ff.]

.30 curves A & B are same except for time dilation of α

We spend α as much time on A, so as to stay on same level in both curves.

Total pc of soln for A = α times that for B.

Only iff t. A & B curves have this simple relation, will Lsrch be correct — i.e. we should always spend α times as much time on A as B.

Since the curve info will usually not be [illegible] in the form of curves that are related in that way: [Lsrch is usually wrong.] — If t. goal is to find soln [illegible] of

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 638.40: present problem, most rapidly. Hvr, if one is also interested in updating info on effectiveness of OT's (538.19–.21) — This is a L.A. (Look Ahead) problem. Perhaps that aspect of t. problem should be worked on separately.

T. only justifn. of Lsrch for OZ probs. is t. usual "optimum within factor [illegible] 2^k" — & k can be made small if we make t. guiding P.D. "adaptive". But, what is t. definition of t pc. that guides OZ Lsrch? Say we obtain it in t. 2 step way.

T. hope is that by "Adaptive Lsrch" that t. pc assoc. w. v.g. PST's will eventually become very large — so Lsrch will "pick" V.g PST's to have hype & much wt.

.10 Still, do we try non-"best" cands (enuf (or right amount)) to give them appropriate pc values?

So: I do understand my standard OZ soln. via "Adaptive" Lsrch — & its assoc. updating of t. GPD. — T. Q is — is it any Good" — & what would be t. "Best" way — & how can we improve t. present method — & how Bad is t. present method?

.17 Perhaps V.G. way: work on "Best looking" Cand. until it no longer looks "Best".
If cands make trials: T. existing trials can be used by other cands & changing their expected Gates. THIS DOESN'T USE LSRC at ALL! → 644.30

.19
.20 Re: OZ Lsrch (or INV Lsrch): I think one of t. "Basic Ideas" was to divide t. prob. solving into 2 parts: Part ①: "Chose (probabilistically) PST's to work this probem
Part ② Search (Probilistically) over t. PST's.
This "Ruff Guess" followed by "finer guess" was t. "Big Idea".

.24 → Perhaps very impt. point: That usually t. pc ordering of PSTs will not depend much on t. CB. If they were actually independent of t. CB, Then OZ Lsrch would (maybe) be quite reasonable (as reasonable as INV Lsrch?). SO CHECK This OUT!

.28 Another point: That the PSTs of hyest pc's will (in an adaptive system) be v.g. at problem solving. They may not be "T. Best", but they will be V.G.

.30 Another point & in line w. .28: After certain level of TM development, it will begin improving the PST's (as well as improving algm. to assign PST to problems). This should further & its efficiency significantly.

V.g. .30: That we have (in Lsrch) been emphasizing that TM will operate close to optimum: If t. optimum PST has initially a very low pc, it will be "exercised" ≈ ϕ so we don't get much info about how good it is.
(The "negative "Lookahead" L.A. effect")

Perhaps t. main way that optimum PST's get hy pc is by deductive & inductive reasoning — like (Human PST's Analysis Methods). I do want 640.00

To W Th F Sat
28 29 30 31

ABC

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: 639.40: to be modeling TM's PST's on Human PST's as involves possl. problem a direct (PST) a Human

■ Meta PST, level.

.02 [So: Make summary of ① Statement of t. OZ search problem ② Possl. solns. & approaches to solns.]

.04 Q: for INV Lsrch, Ross's "GHT" "a gambling House Arm" Rule says $\frac{pc}{cc}$ ordering of trials is Best. Is there a corresp. Rule for OZ probs?

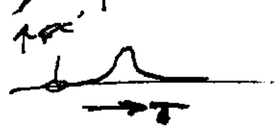
Hvr., infact, t. proof of t. Rule doesn't seem relevant to INV Lsrch! In t. proof, we have ordiy probys of "win" (I'm not sure, but I think t. pc's need not sum to 1 — Σ can be >1 or <1)

/10 (If I allow $\Sigma p_i > 1$ then) my rationale for looking for candidate "Best" cand is not correct!)

GPD can express this kind of PD. Its "norman" is simple conf: p_i = pc that cond. will succeed. $(1-p_i)$ is pc of failure by that cond.

If Σp_i not neclly $= 1$, t. $\frac{p_i}{cc_i}$ ordering is proby still correct — its t. pc per dollar of t. bet. Largest is best.

In TM we have an apparently difrnt situation: For each cond we have a pd over costs ($\equiv cc_i$) of solving t. problem. T. WON analysis is then appropriate.

Say we have a f'corre:  that gives pc of soln. per unit time as funct of T. For all conds, we move to pt. at which pc' are all t. same. We work on each cond an amount so as to keep them all at t. same pc' level. If this is impossl. at a certain pt., for a cond., — then we drop that cond. If Lsrch is to be t. best way, then $f_i' = F'(\alpha_i t)$:

I.e. same form of f_i (except for dilation factor) for all i. (see 638.23 ff on this effect)

■ While t. Q of just how to deal w. OZ/OPT/PST problems is certainly impt.,

I do have a sort of "pro-tem" soln. — t. 2 step Update of GPD. I should simply

.30 write t. report using that soln.

[In "My version" of t. report: Have refs. to discussns. of various problems; Also detaild defns of various Diffys.]

.34 [SN] On Updating On: T. problem is $O_n(A_{nn}, O_{nn})$ isn't "long enuf" — so, a Human mite look at how On processes Q_n+1 & see "why" t. pc of A_n+1 was so small, & perhaps from this, get ideas on how to modify On to get t. desired hy pc, yet not ↓ t. pc (much) of t. older Q_i, A_i pairs.

.37

I think there may be standard ways that a Human does .34–.37 — common Heurs for this task. So in actual TSQ's — see how I solve t. problem(s) → Spec 644.00

5/28/02 ED

Rev :00

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University

53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

641

.00: 640.40: Refs. to start of report! (637.23 - on "Normal Prob Solving").

628.11 ··· 630.00 : 637.20 - .40 ; 630.30 ; 628.17 ; 626.00 - .26 ; 624.21 - .40

(621.06 - .40 [redacted] review; review of reviews) Notes 621.30

Perhaps just write Introduction: Then review notes to see what must be added.

.06 → (START) We will describe a very general system for solving a great variety of problems: The classes of problems solved are, inversion problems, induction problems and optimization problems.

Included in this repertoire are solution of algebraic or logical equations, proving theorems, induction of languages

.10 Symbolic integration, real and symbolic sequence prediction, general Question Answering, and of most importance, updating its own problem solving techniques. As it solves problems, it uses the information it gains to improve its problem solving techniques.

.17 (.175 →) (642.31 - .32) When given a suitably designed training sequence of problems, we expect the system to

.18 attain a very high level of problem solving skill.

.19

.20 There are roughly three stages of operation of the machine: infancy, middle age, and maturity.

.23 When we present a problem to the middle aged machine it consults its (conditional) General Probability Distribution, (GPD), which finds a appropriate problem solving techniques (PST) that is judged most appropriate for the presented problem.

30 The system then uses that PST to try to solve the problem. While this occurs, the system is updating its algorithm that assigns PST's to problems. During the current PST's attempt to solve the problem, it may become clear that a different PST is more appropriate — In which case the system switches to the new PST, until the problem is solved or until the system decides

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

00:641.40 Post a different PST is appropriate ~~[illegible]~~ This ~~[illegible]~~ use of PST's
.01 one possible switching between them, occurs until the problem is solved, our until a time limit is reached.

Insert .11-.12 ~~The updating process consists of modifying the GPD in view of new experience data~~ ~~It also improves probabilities by increasing bonuses on old data~~ ~~Another part of~~ Continuing work

~~The updating process involves also creating new PSTs, and modifying old ones~~, and changing the updating process/involves ~~The algorithms that assign PST's to problems.~~ problem solving

~~assignments of problems~~

In the system ~~described~~, all of the ~~system's~~ information is ~~stored~~ contained in the GPD and in the current set of PST's.

The PST's themselves often have to make decisions ~~with~~ with incomplete

.10 data ~~[illegible]~~ and they use the GPD to help make these decisions.

.11 ~~The updating procedure problem involves~~ The GPD Updating is an inductive inference problem ~~[illegible]~~ involving

.12 ~~[illegible]~~ modifications in certain parameters/ ~~the~~ the creation of new induction models. to fit new data as well as

~~A mature system uses only one PST — that it has discovered itself.~~

Since the system has no restrictions on what kinds of PST's it conjures it is likely that a Mature system will find a single PST that is applicable to all problems — This PST might operate by examining

.20 each problem and assigning a PST to it — ~~[illegible]~~ as ~~[illegible]~~ does the ~~[illegible]~~ middle aged system — or it could ~~[illegible]~~ invent an entirely different system for solving problems

The infant system begins with a small number of PST's and ~~[illegible]~~ a simple algorithm for assigning them to problems. ~~[illegible]~~ Initially, the PST's are all based on Levin's universal search algorithm.

SN Perhaps start with infant system. — Then tell how improvements yield "more maturity" : so start w. 64 l. 06 → l. 9. (dcm of problems).

.30 .31 insert into 641.175 ~~Moreover, the system~~ In addition to solving problems, we want the system to learn from its successes and failures — so ~~it becomes better better at~~ ~~solving problems~~ with more experience, it becomes more competent.

The ~~[illegible]~~ initial ("infant") machine is designed to solve/induction a certain kind of problems ~~[illegible]~~

I Derb. Q A problem as in stuff I've written. "CPM" or Nov 8, 2001 Sept 8, 2001

Looking at CPM (Nov Sept 8, 2001) I don't remember just how the things was supposed to work, or what the main bugs were!

5/29/02
ID

4

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

642.40: Just why did I discontinue N8/02? Maybe because it couldn't solve optzn problems? — at 431.ff I had finished part of t report. There is on [433] a list of things finished & unfinished for t. report. (M5.01 447.22-.23 I found way to implement 3 import umc DF) lot of change in what & wanted for t. report (pp 433-580)!

580.37 Didn't find a whole lot of change in what & wanted for t. report ~~...~~ Unfortunately, t. contrary (if any) of ~~...~~ discussn. is 581 ~~...~~ (4/11/02 582.00-.20) seems to be a v.G. summary of T. present state ~~of~~ "T. report". How it differs from t. ~~...~~ N8 02 Report is not clear! — The it is a much more general system.

581.00-.10 reviews from ~ 461.00 to 486.00

492.04-.16 was a point of some excitement! I'm uncertain of what t. idea was! It seems to be a way to write a TSQ (a very effective way.)

Perhaps at N8.01 I didn't have good ideas on how to implement S-Functs. (?).

582.00-593.40 seems like a great summary of t. state of TM. 5.

[582.00-.21] is a particularly ~~...~~ good, short, summary

Q: What happend from 582 to present? : A BIG CHANGE is 639.17 NOLSRCH!!

What I can do now : Make a kind of report, but mainly for ME. Have sections (Including Introd., that tells what it is how it works, more or less.). Tell just what each section covers. Also, (for my own use) what are future developments expected? What are diffys? — How do I expect to deal w. them.?

Usually, write out answers to each Q. — but occasionally use refs, if t. refs are really clear.

I Guess t. part least clear in my Mind is ~~more~~ details of update of QA problem.

Note that for many QA problems, t. S-funcs will not be represented by t. 3 ingredients — but in other ways, & I will have special ways to update them. [~~T. idea at 582.00-.21 seems close to this, and method of 641.23-642.01 is closer. This last does not seem to use LSRCH at all, as far as I can see!~~ (This should not be here! its at 644.31-.33)]

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00 : (640.40) : One way to avoid disturbing olde Q,A pc's is by partitioning Corpus — so Qnew, Anew is in a relatively small "sub-corpus".

.03 SN On GBT (640,04). T. Thm is right. (probably even if $\leq p_i \neq 1$). T. Q is can I legitly apply it to Lsrch a/o OZ lsrch. If I can, show this, it would clarify things Q monsely! It still wouldn't apply to Irng during Lsrch, but that's not so Bad!

→ For OZ probs, hypothesize that t. P.D. doesn't change much with CB.
T. wierd thing about Usual Lsrch, is that t. pd is a same measure, I can beat measure.
For INV probs, t. 2 step GPD does yield a pd that I would want to bet on via GHT.
T. pd gives t. pc that a cand. is "Best." I really have to think about this a lot & write up

.10 conclusions & argts. Very clearly! T. wierd thing is that I simultaneously want a pure soln

.11 of small cc & I'm doing Lsrch so as to minz investment of cc. Tho, literally, it's (.17)

.12 SN In t. 2 stage update for INV or OZ probs: I could approximate t. integral soln. by picking t. cand. w. least expected cc for soln. A trouble is: this maybe OK for most cands, so they would get pc ≈ 0. ↑ This only gives t. "Best cand", not t. PD that I Need for Lsrch!

.17 (.11) t. latter (Min. ■ ≡ investment of cc) that I'm interested in, not merely finding Cand w. smallest cc. So t. GHT "goal" is correct.

The Applicn to the OZ probs SEEMS to correspond closely to t. INV. applicn.
Say as t. OZ problem is a Mmzn ≤ (like in INV probs).

T. GHT says Lsrch is ≈ best it we don't watch Learn during Lsrch.
Hvr., in fact t. info we have is not really appropriate to GHT — it's more usable by WoN.
It would seem that Lsrch is a simpler approach than WoN: But I'm not sure.

SN NB: One of t. IMPT (Post ≤ 89) ideas was that Irng during Lsrch was ≈ Essential: Along w. its synonyms "Switching Instructions", it was able to deal w. t. "correlation problem" betw— Cands.

.30 : 639.19 HA! So we may be able to use .12! (find single Best Cand): work on it unAll
.31 it is no longer t. "Best" cand. This is what I've been considering — as long ago as 582.00-.21 is close to it & 641.23-642.02 is closer. This last (is perhaps
.33 582.00-.21) does not use Lsrch at all! ← A Real Serious Change!
I think One Advantage of Lsrch was that it was rather "Non Elitist" —
That it did consider a Very Broad RANGE of CANDS.

Hvr, this serious updating during Lsrch does not normally occur in Infant TM — so we do use Lsrch at that point in the Life.

(Spec 645.00)

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

.00: (Spoc 644.40): Actually, Lsrch may not be eliminated by parallel updating! We can do continuous (perhaps Mt-Carlo) Lsrch and simply modify t. pc's during t. Lsrch. This would preserve Lsrch's virtues [Non-Elitist, "optimality"], tho doing 644.12 & not doing Lsrch at all, may be much faster! ←

SO: For Report: Write it up first as purely Lsrch w. sequential update, then discuss (.00) then 644.30 (No Lsrch at all) as "improvements" to t. System.

.09 → In t. Method used by N8.01 (Partitioning), we first "Recognize" what kind

10 of problem we have (w.r.t. t. set of all QA problems): Then we use a "PST" assoc. w. that particular problem type to solve it. Could this be a special case of t. more general techniques in which TM uses GPD to categorize a problem → chose a PST for it (or get a PD on PST's for that problem)? It would be neat if I could somehow integrate the N8.01 method into t. general system — that part of N8.01 seemed rather "special situation"/A.H.) — I didn't really have a clear picture of the Methodology.

In t. Method of N801: T. "Recognition" was B/W, not "Gray" i.e. a D funct. not a S funct. function. If they were S functs, that would ↓ pc of results a lot! Maybe not! We could have

20 various PE's for various possible "Recogns": Each "recogn" then has its own D.F. over t. set of all A's: We get Baysian || coding! As t. System "Smartens" the Recogn. gets sharper → D-functs. [w. assoc. t. in A's i.e. "SMARTENS"]

.23 Remember that GPD is a rather Complex (Versitile) object & can implement (Store, Represent) many different forms of Info/knowledge! — So it should be able to do "soft" Recognition. This soft recogn does seem to work O.K. GPD gives, say, 3 possible "Recognitions" of a Q_i: R_{i1}, R_{i2}, R_{i3} (with pc_1, pc_2, pc_3) : $\sum_{j=1}^{3} pc_j \approx 1$.

Each R_{ij} has a $P_j(A_i|Q_i)$: T. resultant is $\sum_{j=1}^{3} pc_j \cdot P_j(A_i|Q_i)$.

30 [The "report" N.8.01 has some discn. of how recogn. works — & t. handwritten IP notes up to & after N8.01 have more discussn. I should be able to integrate (.23) (into) w. that discussn. If I can do it, this will make t. "Recogn" trick much less A.H. — Much more "Natural" & probably easier to find/design heurs for it.

Well! I'm pretty sure it will work (i.e. .09 ff)! Normally, we have "indices" for TM to be able to categorize Q's — at first we do t. categorizing — & later TM must lrn to do t. categorizing of Q's. The Recogn. functs used in N8.01 are simply a B.W case of categorizing that I wanted TM to eventually lrn — But lrning usually gives S-functs.

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

00: 646.40 Corpus: ~~Usually~~ Internal problem solns. must somehow speed up or otherwise Better t. solns of External problems.

Internal problems can use concs from external problems, but ordinarily, not t. other way around! A scientist might make up an internal world (Mathematics, say) & use it to predict things in R.W. — perhaps t. reason This Works is because T. Mathematician is imbedded in a culture — that has many problems — Some linked to Math.

646.40ff seems very inept. I had been assuming internal probs for some ("crudely" "Cost") as external. I must find some way to deal w. this since (most problems) probs in a mature TM are probably internal!

.12 6/2/02 A possl Soln: That getting a good Move for 414.18 is regarded as "just another of TM's problems". This Does enable transfer of info (between) from TM's External TSQ to this particular "internal" problem. We may or may not want this problem to be part of t. corpus that 414.8 maximizes. If we do (we do!), Rqts is a bit of recursion, so we have to be sure a "Boundry Cond." is implemented. If we don't then TM really doesn't seem to be working on that "internal problem".

.20 .25 I think This is Easy: T. "Boundry Cond" is t. External Corpus!

The my mind isn't Entirely Clear on this (.12–.25) seems to solve t. problem! I think this approach will clarify t. business of [illegible] Context (local or Extended) dependance of Concs used to generate trials for 414.18.

The (646.2I). t. original disturbance about internal v.s. external problems — was concerned w. "Recogn. functs", t. Soln. of (.12–.20) would consider Recogn. functs to be part of t. External Corpus Dcrn., so It would not be a problem in this Area. — It will have to be treated in another (probably Simpler) way.

.30 I had been Concerned that if internal problems were mixed w. external, that internal probs could get arbly (argue w. à la "become a solid contemporary hypoth"): Hwr, It seems that t. internal problem of "Move 414.18" doesn't generally need to learn any single external problem. {Hwr, certain single problems can get a lot of wts. if they are suitably configured. ↳ 650.34

↑{Look into this!}

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

00: 645.40: So when a Q comes in, GPD categorizes it in a (S-funct) way. — as 645.23 ff. We do want to get ≈ D-functs, hwr, because they make it easy to check a new Out funct, since only a small subcorpus (i.e., t. recognition class, itself.) is affected when t. P-function for a Recognition class is modified.

If t. Recogn. functs are ≈ D-functs, we can! can so much cc in verifn. of changes of t. [illegible] P functions assoc. w. that (approx.) Recogn. class.

I suspect that I've written a Paramount about this Q, — Much earlier than N.8.01, hwr.! In an earlier TM model — Maybe when I was at IDSIA.

Some time ago — (Maybe at IDSIA), I had the idea of a great variety of IA's (induction

JOINT CENTER FOR URBAN STUDIES of MIT and Harvard University
53 Church Street Cambridge, Massachusetts 02138 (617) 495-7908

On S-fonts:

I can easily generate a universal d.f. on d-functs. Using AZIAI and MoCarlo — Laplace-Bernoulli method: i.e. after a string of n legal symbols (including "procorpos", the next symbol is a uniform random choice of all legal previous symbols. (The procorpos is occasionally augmented as new symbols become legal.)

T. forgoing realizes the S-function in the first argument of Rec 3 input one. Model of Q A induction. It gives a p.d. over d-functs w. any no. of inputs.

Publicize: S-font

Such things can be generated by the above M. Carlo process. We get a universal d.f. over strings. (a universal lang ... if the rewrite rules are "Turing Complete")

In fitting continuous params to S-functs: If SSZ is small (which is usually true for mini-TM) the d.f.'s are Broad. Easy to find the Mode & there is no reason to find the peak exactly. I.e. 11 codes would be a good approx.