Bost

[illegible]

IDSIA Talk 1

Introduction: General Goals, Method of Project

A Progress report on a system for Machine Learning that I've been working on for many years.

System for ML: Able to learn to solve practically any kind of problem, after having a reasonable "training sequence" to teach it to solve problems of that kind. As the talk continues, it will become clearer as to just what I mean by "Learning", "Kinds of problems", "Solveable" and "Training Sequence."

I will describe the system briefly, to give a general picture of how it works: then I will go over the sytem in more detail, telling just how the parts work.

This will be followed by a discussion of extensions of the system, present limitations of it and weak points in the present state of the system.

The system starts out pretty much like a newborn baby - with very little specific knowledge about its environment, but with some simple, very general, *learning* algorithms and *problem solving* algorithms. We give the machine some simple problems, which it solves easily. The solution techniques for these problems are then integrated into its problem solving Algorithms, so it can solve the next set of more difficult problems. After it solves them, it again integrates the solution methods into its problem solving Algorithms - So it can work even harder problems - as we continue our training sequence of problems of increasing difficulty, the system becomes capable of solving problems of greater and greater difficulty.

A bit more detail: What kinds of problems does it solve: At first:

Two kinds - Inversion problems and Optimization problems.

Inversion problems. Given a function f defined by a specific computer program. The program has input x and output f(x)=y. The Inversion problem is given some "y" value, to find x such that f(x)=y. Here x and y may be digital strings of symbols a/o numbers.

A simple example is f(x)=x^2: If f(x)=9, to find x, or
f(x) = sin x + x^2 + ln x; y=3.
A more complex example: given a theorem in Math, we are required to find a proof of the theorem. We are also given an algorithm that is able to look at any theorem and any string of symbols that represents a proof of that theorem, the algorithm then says whether the string is a valid proof for the theorem:

Alg(Theorem, Cand) 1 or 0 (Yes or No).

INVersion problems correspond to the P and NP problems of Comp. Complexity Theory.

Another kind of problem, is the Time Limited Optimization problem. The formal form of this problem: given a function whose input(s) are strings and/or numbers. It's output is real numbers G(X)=Y. To find in 10 seconds, say, an x such that G(X) is as big as possible. If we were given 20 seconds, we could probably find an x with larger y. An example would be to design in 6 months an automobile with certain characteristics,

Second [illegible]

IDSIA Talk 1

Introduction: General Goals, Method of Project

A Progress report on a system for Machine Learning that I've been working on for many years.

System for ML: Able to learn to solve practically any kind of problem, after having a reasonable "training sequence" to teach it to solve problems of that kind. As the talk continues, it will become clearer as to just what I mean by "Learning", "Kinds of problems", "Solveable" and "Training Sequence."

I will describe the system briefly, to give a general picture of how it works: then I will go over the sytem in more detail, telling just how the parts work.

This will be followed by a discussion of extensions of the system, present limitations of it and weak points in the present state of the system.

The system starts out pretty much like a newborn baby - with very little specific knowledge about its environment, but with some simple, very general, *learning* algorithms and *problem solving* algorithms. We give the machine some simple problems, which it solves easily. The solution techniques for these problems are then integrated into its problem solving Algorithms, so it can solve the next set of more difficult problems. After it solves them, it again integrates the solution methods into its problem solving Algorithms - So it can work even harder problems - as we continue our training sequence of problems of increasing difficulty, the system becomes capable of solving problems of greater and greater difficulty.

A bit more detail: What kinds of problems does it solve: At first:

Two kinds - Inversion problems and Optimization problems.

Inversion problems. Given a function f defined by a specific computer program. The program has input x and output $f(x)=y$. The Inversion problem is given some "y" value, to find x such that $f(x)=y$. Here x and y may be digital strings of symbols a/o numbers.

A simple example is $f(x)=x^2$: If $f(x)=9$, to find x, or $f(x) = \sin x + x^2 + \ln x$; $y=3$.
A more complex example: given a theorem in Math, we are required to find a proof of the theorem. We are also given an algorithm that is able to look at any theorem and any string of symbols that represents a proof of that theorem, the algorithm then says whether the string is a valid proof for the theorem:

Alg(Theorem, Cand) 1 or 0 (Yes or No).

INVersion problems correspond to the P and NP problems of Comp. Complexity Theory.

Another kind of problem, is the Time Limited Optimization problem.

The formal form of this problem: given a function whose input(s) are strings and/or numbers. It's output is real numbers G(X)Y. To find in 10 seconds, say, an x such that G(X) is as big as possible. If we were given 20 seconds, we could probably find an x with larger y. An example would be to design in 6 months an automobile with certain characteristics, having minimum code.

This is different from the optimization problems in which time is not specified, and we want the value x such that M(X)=max. In general for most of the optimization problems we will work on, there will be no knowable value of x for which f(x) is max - that in the specified time we will have to do as well as we can, but expending more time usually means better optimization.

That talks about the problem class: As we continue, you will hear more about the problem class.

The next question is "How does the system solve problems"? The simplest version of the system uses a variety of "Blind Search" called "Levin Search" or "Universal Search".

The way it works: say we are solving INVersion problems. Any finite binary string (which represents a computer program) is a "candidate for solution". Lsearch uses a probability distribution to guide its search over these candidates. Candidates with high probability are tried first.
- But since we are interested in solving problems as soon as possible, we don't spend much time testing a particular candidate, unless it has high probability.

Roughly speaking we test condidates in P/T order, P being probability of the candidate, and T being time needed to test it. Using Lsearch, if a candidate has been given probability P and it takes time T to test it, Lsearch will find the solution in total seach time T/P. If there is > 1 solution, it will find solution of *min* T/P).

Note: the probability distribution guiding the search, is a *Conditional Probability Distribution*. So input to P.D. is problem description. Output is list of candidates and associated probabilities of each candidate.

P (problem description) (cand sub i, p sub i) in rough p sub i order.
The function P contains all of the problem solving skill of the system. P will change as the system learns more and more.

Suppose the system has solved a few new problems. How do we modify the system so that it is able to solve more difficult problems? We modify the Conditional Probability Distribution so that usally the solutions to the new problemms are given higher probabilities than before. Since solution time = T/P and T remains invariant for that particular solution, increasing P will decrease solution time.

Our method of modification of the Guiding Conditional Probability Distribution will be dealt with more fully, after we discuss the estimation of probabilities.

The particular method of *probability estimation* we use is called Algorithmic Probability. Suppose are doing sequential prediction: we have a long binary sequence, S, and we want to know the relative probability of the next symbol being 1 rather than 0. -

Let us assume that we know an apriori distribution on all possible Binary strings. P(R) assigns a probability to each such string. Then the relative probabilities of the continuations 1 or 0 will be P(S1)/P(S0). This would solve the sequential prediction problem, if we know P(R).

HOW TO FIND P(R)

For a heuristic understanding of what follows, consider Ockam's idea:
That simpler hypotheses are more likely than complex ones. We will quantify this idea and describe some results on the accuracy of the resultant system.

To quantify Ockam: Suppose B is a finite binary sequence. - Then D sub B is "description" of B if M sub r(D sub B) = B.
Here M sub r is an algorithm or computing machine that is able to take the binary string D sub B as input, for which it gives B as output. D sub B can be regarded as description of B in terms of Machine or function M sub r.
I use the subscript r, because M sub r is to be regarded as a *reference machine*. Clearly, by changing reference machines, the needed descriptions for B will change. The shortest description will be the one with fewest bits. - containing least information. We could approximate the probability of B by 2^-D sub B. For more accurate estimation, we use Sigma sub i 2^-D sub Bi. (Also mention *theory revision*". Here we are getting the total probability due to all possible descriptions of B. We will call Sigma sub i 2^-D sub Bi the Algorithmic probability assigned to B by M sub r = p to the M sub r(B).

Thre are a couple of loose ends here: first, what reference machine to use for our *probability calculation*? There is a group of machines called "Universal Machines", that are particularly good for describing things. They have the property that if we compare 2 of them, there will always be a constant factor that tells how much they differ from one another.

Two machines M1, M2 then P to the M1 (b) and P to the M2 (B) are always with a constant factor C1,2 of one another. This constant factor depends on M1 and M2, but is independent of B. The constant factor can be quite large - so it usually makes a lot of difference as to what reference machine one uses.

In our system for Machine Learning, we will be periodically changing distribution that guides searches for solutions to problems *is* an induction problem. How is this so?

Consider all of the (problem description i, solution program i, time i) triplets that have occurred this far. What we want is a bunch of relatively short codes to describe all of this data. From a set of codes of this sort, we can then extrapolate the data to any new GPD sub 2 (problem, solution) -> probability distribution on Time to solution. From GPD, it is possible by a process of integration, to create GPD sub 2 (problem, solution) -> probability that this is the fastest solution to the problem.

This last distribution is the guiding distribution for our search for solutions to new problems.

I will now discuss Levin's universal search procedure: suppose I'm in a Gambling house and there is a kind of lottery with a single large prize. I am the only custaomer. Each lottery ticket has a certain probability of winning - which is printed on the ticket. In the first kind of lottery all tickets cost $1. The best ticket to get would be one with maximum win probability. If it doesn't win chose next largest - and so on.

In the next case, each ticket costs a different amount of money, (Pi, Mi) are the probability cost associated with the ith ticket type. The best choice to make is the ticket with maximum Pi over Mi. You get the maximum probability of winning, per dollar spent. If you continue to buy tickets in Pi/Mi order you are certain to have least expected money spent before winning.

The gambling house can be translated into a problem solving environment. Instead of money for a trial, we spend TIME. The best trial to choose is one with maximum pi/ti.

Normally, in trying to solve problems, one may know pi, but one doesn't know ti. In this case, a very good strategy is Levin's time share strategy - (which I think is likely to be the best possible). The way it works: you work on all trials simultaneously, but you work harder on trials with large pi values. The rate at which you work on trial i is proportional to pi. Supppose we use this strategy and after a while the jth trial gives a solution after spending a total time ti on it. Since other trials have amounts of time spent on them proportional to their pis the ith trial will have Pi/Pj, tj time spent on it.

The total time spent on all trials will be Sigma sub i Pi/Pj tj= tj/pj sigma pi. Sigma pi will in general be 1. So the total time needed to solve the problem this way is tj/pj.

So: this is Levin's time shared seach and I think it's as fast as is possible for the kinds of problems in which it is used - i.e. INVersion problems.

This technique can also be used for time limited optimization problems.
In this case our trials are not programs that attempt to solve inversion problems: They are *optimization techniques* each one takes as input, the problem description, G(X), the function to be maximized, and the time available for solution. As before, we have a conditional probability distribution, that looks at the problem description and assigns probabilities to various optimization techniques to be applied to that problem.

The way we do the search: say our time limit is T for some small tau we spend time tau times pi in the ith O.T. and slowly increase tau. We keep track of the O.T. that has best G value of all the O.T's when tau pi = T for any OT, we stop spending time on that O.T. When tau pi = T for any OT, we stop spending time on that O.T: When tau pi= t for the best O.T., we stop.
This is our best value. Total time =< Sigma tau pi = tau sigma pi = tau : tau = T/pi, so time to obtain best solution in time T, is T/pi, so we have efficiency factor of i/pi - as we did for INV problems.

~ D6.00 Bolg.

.04→.05 : T. idea that in Lsrch, t. main prob is to Modify t. Pd. for t. problem ⇒ in Lsrch t. factor 2^{ℓ} is not large. — ideally ~ 2 or 3 or even one! ⟶ but note : (.15) !

.15 Q: Is .04→.05 Literally correct? One doesn't ever do Lsrch w. large CJS? Just exactly what did that "small 2^{ℓ}" mean?? I think t. idea was that Lsrch was "optimum if all t. info was in t. Pd". — If a Long Lsrch (w. large CJS) is t. best way, then 2^{ℓ} will be large.

I think 2^{ℓ} is usually not very large — that an impt. part of TM's work is to modify t. P.D., so that 2^{ℓ} usually is small (but not nearly as small as 2 or 10 !).

Another Q: I had t. idea that Skinnerian TSQ's (very small CJS) were not v.g. — That a large srch was more "educational" for t. Student. This would be true if t. student tried "interesting" trials — if t. soln. was really "educational". To what extent is a normal, large CJS Lsrch "educational"? — How is it "educational" — ~~what~~ in what specific ways?

This seems like an impt. Q! Perhaps CRITICAL!

An Approach: Consider 2 ways of ~~wotting~~ working a "large CJS" problem:

1) straight Lsrch 2) Normal method(s). (a) Do they really differ (b) if so, how does t. "normal method" give more "training" to TM than straight Lsrch.?

E.g. Consider a student.

In "straight Lsrch, w. a "Big Problem": T. soln. is pretty much as Envisioned in S89, using a "Conc. net". A Big Problem is solved no differently than a "small" problem, But t. cc is larger for a large problem. After solving t. large problem, TM isn't nearly any better off than if it had been solved in a Skinnerian TSQ. — in this latter case, TM would have ↑ pc of t. concs. used, but if t. prob is solved as a single large prob., then t. concs used don't nearly have cc ≥ ? & ~~so~~ need not be recognized as "useful". So normally a Skinnerian TSQ with t. ~~[illegible]~~ concs in it will normally be better than having TM solve t. final problem at very large CJS (w.o. a TSQ. leading to its soln, as in t. Skinnerian Case). →2.01

D30.00 Bulg.

SN

from "Expo [redacted] 1.40".

.01: On other extreme Situation: Consider very large problem: One starts out by doing much "Meta-lrng": [redacted] i.e. Hyer level lrng. a/o learning in general related to final problem. E.g. Going to College before trying to solve Energy Problems.

["Meta lrng" is not t. rite term " ~~Meta~~ duetolrng" is about how to lrn; about how lrng works (& fails!). For a larger problem, one mite do some Meta lrng. in preparation for it, but one would mainly do lrng on T. field of study & related fields, (which is what I'd like a term for) T. main idea here is that This study is Non-greedy — not directly trying to directly solve t. problem.

TM$_2$ could learn to design ~~do~~ this "Pre education" as a trend to "Self taught TSQ".

.10 → A perhaps better Tack: Say t. P.D. for Lsrch is conditional on ① t. initial

.11 problem descr. ② T. trace thus far of t. work on that problem. This Certainly seems like a reasonable way to ⓐ do prob. Solving ⓑ take advantage of previous experience.

So t. P.d. is on "what to do next" in view of ①(.10) & ②(.11).

.10 seems like a more general method of prob solving than a "Simple Minded" Lsrch.

.15 While "Simple Minded Lsrch" does give a factor of 2^{l} slower than t. "Best Method", t. Q is: Does it give us any way to make l "small"? Could .10-.11 be regarded as a possl. way to do this!? I.e. t. P.d. of .10-.11 can be regarded as a P.D. on prob. solving pgms! So, a $\boxed{\text{N.G. TM}_2}$ ~~Simple~~ would design a P.D. for Simple Minded Lsrch, that was really t. same as .10-.11.

In .15 I think t. P.d. on prob soln. methods ~~does~~ is not conditional on t. "trace thus far" — So perhaps TM$_2$ couldn't use that info for a "Simple Minded Lsrch"

— In .10-.11 one has both t. original p. solving cand pgm. and a modifn. of t. pgm as it evolves (= runs a trial soln). — So proby of soln. could drop during t. trace — This could be a way of Implementing "QUICK ABORT" !!

.25 (Hvr. it would seem to be out of t. "spirit of Lsrch", because, during t. trace of a trial, pc would not ↓ monotonicly, it could ↑ a lot if it began to look like a Soln. was "Near". Would .10-.11 have this diffy?

One way to deal w. .25: For Lsrch control, (i.e. choosing next cand.) use only initial (pre execution, pre-trace) estimate of pc of cand. For discontinuing a trial, use recent (trace) info. [I'm not sure this ⓐ will work ⓑ is well defined/(meaningful!)].

Another possy; that t. Cands themselves are ~~Probabilistic~~ "Problistic" — meaning, what?

.32 Stochastic — Monte Carlo-ish? (I May have once used an idea like this to deal w. t. difference betw. pc & bc in Lsrch) → see .38 hvr.!

[SN] Define t. Main problems Clearly!, Exactly!

One attempt at one problem! Re: .10-.18: In order to express many Heurs, we need to use trace info from each trial. Hvr, this screws up Lsrch, because pc is not - monotonic during t. trial, it can ↑ as well as ↓, & is meaningless? impt!

.38 On .32 we discuss Mt. Carlo: Hvr, by use of Lsrch we can Make it deterministic, if we like!

D31.00 SN Bulg.

Solomonoff.com
.org.
.net.

Remember that t. "soln." of a problem is a "trace" (= method) rather than a simple final answer.

Still, a [redacted] soln (trace) to a problem can be completely desrb by a finite, deterministic string.

.05 Another way to desrb. a soln. is by a probilistic string, in which t. pc of t. string ↓ w. leng th. I Think this ends up w. a probilistic d.f. on strings that has "t. sequencal property" — i.e. $\sum_i P(a\,b_i) = P(a)$ a is a string & $[b_i]$ is t. set of all single symbols that can follow a.

⟶ So it looks like using .05 is essentially t. same as using Lsrch on a P.D. of solns

Best to look at 1 or more prob solns. to see how Lsrch fits (if, indeed, it does)

.12 "How to solve linear equs:" Move all terms to LHS. of eq. : Then simplify as much as possl. Try to get eq. into form $ax+b=0$ Then $ax=-b$ Then $x=-b/a$.

"Simplifn" is a rather complex operation, & has to be lrnd from examples: Perhaps learn what term "Simplify" means by examples — ⟨Tho in many cases, "simplifn" may have special meanings.⟩ → .30

Links on IBM TP:
1) Rt side under Bar: 3 FN Jacks
2) Novita on/off sw!
a) ⏻
b)
So 5 items.
3) Back edge of top cover: 3 lites. Battery
Ⓝ
Ⓝ goes Green when machine "on".
may tell battery in, & if Batt is charging.

SN) One of t main things I had against Skinnerian lrng, was that we might not include certain very impt. concs. that were needed for non-Skinnerian lrng. Sk. lrng is a lot like Lenat's teaching Cyc by "brain surgery": by carefully giving it "facts" that he thinks it may need. I can't give example of such a conc. — Perhaps in "normal" "Expert Systems", t. "brittleness" is due to lack of these concs.

NB) In OZ probs (which are most problems) TM spends very little time in "Lsrch". Each OT is "its own thing" & may or may not use Lsrch in itself. T. set of OTs themselves are orderd via Lsrch, & t. "master P.D." has t. job of matching [redacted] an OT to a given OZ problem (or to find a pd over t. OTs for each problem).

INV. probs are often (maybe usually) converted to [redacted] ≈ OZ probs via t. "GPS" method of a Vector Conc. (.12) could be viewd as a kind of "OZ" soln to an INV. prob.

.30 "Simplifn" is a kind of OZ problem, but t conc. often has several components. Certain components are more impt (maybe essential) in certain probs.

Getting back to 1.15: An assoc. problem is: If Lsrch is ≈ optimum, then most/all heurs have to be inserted into t. P.d. I need some good examples that are easily gened.

.34 [If a heur. reduces/compresses t. srch space, this would seem to be expressable as modfn. of t. Lsrch Grand P.D. It would seem, that Many heurs would be of this form.
.36 ⟶ see 4.13 for Possl. Serious objection/diffty.

.37 So there seem to be several general schema for probSolving. There is .34, which assumes all needed heurs are expressable as GPD modulus and (This GPD is conditional on nature of problem only ⟹ (?)).

.39 There is another (perhaps dfrnt?) method of prob-solving in which one solves a

1.3.0~~1~~ 1Bulg.

\ Abcdefghi Abcde ABcdefg

Abcdefghijklmnopqrst . ABCDEFGHIJKLMNO

4

.01:3.40. Problem by Lsrch, using a Z141 type P.d.: we then look for ways to compress t. G.P.D. — using t. latest prob Solm. Soln, (but not merely using it: One could simply try to compress whatever part of corpus one sees ~~reppys~~ in: but using t. ~~[illegible]~~ ~~latest~~ prob soln. as part of t. Compression is better because its more likely to be relevant to future (& newer future) problems)

● [T. Main Prob] I'm Working on now is [redacted] "What is t. over-all Irng Algm of T.M?"

~~There~~ is 3.34-.37 & 3.39ff so far.

Perhaps related to 3.39ff: T. idea of my working a problem, then trying to break down my soln. into a "concnet" — that would [redacted] implement its soln. via Lsrch.

Re: T. Argt of 3.34-.36 on why more heurs can be implemented as Modifns of t. GPD: Some possl. Objections:

.11 1) Most Heurs are ~~[illegible]~~ not for ~~[illegible]~~ Main problems but for Sub-problems, so they aren't (immediately) relevant to Lsrch for t. "Main" probs. 1□ $\int_{-\infty}^{+\infty} e^{-\frac{x^2}{2\sigma^2}} dx$

.13 ●→ 2) Many of t. Modfns of (G) Pd's demanded by 3.34-.36 may not be easy to implement for Lsrch: i.e. Th. modifns may give a GPD in which it is very diff. to put cands in ≈ pc order. Looks VERY IMPT!

[≈SN] Re: .13 PD's come in various forms of defns of them. ~~Some~~ In Some forms, certain operations are easy, other operations diff. Xfmg. from one form to another is sometimes a way to enable certain operations (like putting Cands in ≈ pc order for Lsrch). ABcdefg $\int_{-\infty}^{+\infty}$

A few types of Pd defns: 1) In terms of a U10 machine w. random input: Could be a Mt Carlo type, 2) Input is string, output is pc of string. 3) Input is N (an integer) output is Nth most likely String. (This last is useful for Lsrch). 4) The Z141 Model: Is this a variety of ③? & c.f. {Z141 Model} ABcde 5) General Mt. Carlo: Pc of output = pc of that string. While t. Model of ① does this, we need a conditional Mt Carlo: e.g. Given t. first t data of a T.S. (condition), to get a Mt Carlo d.f. for t. next data pt. (This is what a "Summarizing Machine" does ~~[illegible]~~).

Can I put some (or All) of t. Heurs implied by 3.12 (How to Solve linear Eqns) int. form of Modifns. of t. GPD? (Note .11 of hvr.)

* 1.4.01 Perhaps look at older A.I. literature for a system to do Alg. probs: Look at heurs used, & see how they can be put into form of modifn. of GPD.

Perhaps look in "Encyc of A.I."

.31 The idea is: All A.I. probs involve search: Any heur must narrow t. search: If t. search is characterized by a Pd, then ~~an~~ any algm. that narrows t. search can be interpreted as a modifn. of t. Pd used to guide t. srch.

From .31 t. Q. then seems to be .13 ... ordering t. pc's for Lsrch.

see ~~[illegible]~~ 99 TM 9.05 (1.15.99) for a detailed proof that "All heurs can be expressed as modifns of t. P.D.

BBA

Boston Bicentennial Associates, Inc.

102 Mt. Auburn Street
Cambridge, Massachusetts 02138
617-491-8007

Maybe good idea to write a few "Internal Reports" (≈ "Papers")!

1) On Practical Optimality of Lsrch. This is an explanation of Why Lsrch, properly done, may be close to ~~optimal~~ the best possible achievable with the given informational and computational constraints. Computational constraints are cc. [Give (many) Examples!

Informational Constraints are the TSQ., "heuristics", mathematical techniques, general knowledge of the domain of interest.

a) That, by Lsrch, I mean the Lsrch guided by the Cond'l. pc [redacted], P(α): Given the problem descn, α (a string), It is an output P.d. on strings that are possl. solns. of the problem, α.

Give examples of kinds of probs; (T.S. extrapoln, bag extrapoln, OZ probs, INV probs) and what kinds of strings, P(α) is a d.f. over.

.15 b) Heurs can be defined as techniques to speed up (more exactly, make less costly) the search for solns.

c) Because of b), heurs can perhaps always ← No, not always (really) see 11.03,.05,.18 for why not! be expressed as modifns of the P.d., P(α).

They are in classical A.I. modifns of the search; instructions to first look in certain regions of the srch space [redacted]"

In Lsrch, the ordering of the search trials is governed by the form of P(α):

.19 ~~So we can implement heurs by modifying P(α)~~ So, if we use P(α) to guide our Lsrch.

d) Though a particular heur may, indeed, be implemented by modifns of P(α), those modifications could take a form in which it is very diff'lt to order the P(α) trials, in the pc order, to implement Lsrch. How can we make modifns. in P(α) that ① will implement the heur ② will make it easy to order the trials in the pc order.?

.25 One possl. way: Any particular heur must have some reason for existence:

ⓐ Either thru the empirical history of its discovery (i.e. what solutions to what problems suggested this heur), or ⓑ the heur might have been obtained by Analytical/Mathematical reasoning.

If ⓐ is true, then we can ~~use~~ take the set of past examples (or invent a suitable

.30 set of "pseudo past examples") and use our normal induction algorithm → 6.05 to modify P(α) in view of that part of the "pseudo TSQ".

This "normal induction Algm": has to be spelled out first. See 6.05 on this

.32 If ⓑ is true, then I assume (but have not yet proved) that a similar modification of the P.d. P(α) can be made — that Mathematical reasoning is a kind of probabilistic reasoning, but with probabilities of 0 and 1 only, ... Then why consider it to be probabilistic reasoning? — the answer is — "So we can apply the methods

.37 of ⓐ to it to modify the P.d., P(α').

.17.01 1 Bulg Expo.

BBA

Boston Bicentennial Associates, Inc.

102 Mt. Auburn Street
Cambridge, Massachusetts 02138
617-491-8007

.05: 5.30 On "T. normal induction Algm": This is Given a P.d., $P(\alpha)$ and a set of examples (a BAG) assoc. w. a particular value of α [illegible] ($\alpha \equiv$ problem dcrn). Or ~~[illegible]~~ $P(\alpha)$ and a set of α_i & a BAG for each α_i. The α are t. prob dcrns, t. BAGs are t. problem solns.

[I Guess t. elements of t. Bags are traces of prob-solns]

.10 The problem is to modify $P(\alpha)$ ~~[illegible]~~ $P(\alpha) \to P'(\alpha)$ so that $P'(\alpha)$ best expresses

.11 t. past corpus plus t. new augmentation to t. corpus of t. $[\alpha_i, Bag_i]$ set. → see .15

~~The~~ The problem dcrn. for $P(\alpha)$ is: {$P(\alpha)$ and t. past corpus plus t. new augmentation of t. corpus.} — And t. standard request to do

From this, $P(\alpha)$ outputs a p-wtd ordered sequence of ~~[illegible]~~ OT's (optzn techniques) (These OT's p[illegible] order — if t. pc's are Given)

.15 .10 — .11 is t. problem of modifying $P(\alpha) \to P'(\alpha)$ so that P.c. (P'dcrn) · [redacted] $\prod_i P(\alpha_i, Bag_i)$ is Max

I guess t. Expert problem is to get a good set of OT's for probs like .05 — .15 ——

Or perhaps a single good O.T.

A possl Approach! Do some simple problems in Algebra: See how t. techniques of .05 — .15 Map into t. way I solve t. problems. well, maybe: but t. kinds of problems dscrbd in .05 — .15 seem more dift.

1·31·01 1 Bug.

7

maybe also discuss F.N. 15 of Sol 89 (.26)

An attempt to outline, in more detail, what I will say at Lugano (IDSIA)

Some impt. items:

.02 1) Demo that the system is within factor of 2 or 4 of optimum for the cc & incremental constraints. Most Give several examples. Give 3 ways to show optim. of Heurs to modifications in the P.D. (Give counterexamples, too.)

.03 2) Explain, & detailed descr. of meaning of 3 GHT's.
(GHT3 is .02)

.05 3) "Quality" of a P.D. — what it means to "improve the P.D.".
These P.D.'s are partially ordered (maybe > 2 dims) — maybe tradeoff betw pc [redacted] of cong's & time to compute cong's. We want $\frac{pc}{T}$ to be max (≈ 1/cost!)

In Sol 86 I discussed "compressing the P.D." There's only one (but major) aspect of "improving the P.D."

4) Discuss varieties of P.D.'s, how they are often wholly or partially transformable into each other. One compl form: to output strings in pc order (w. pc's): Max pc first.

SN "Quick Abort" "Quick Abort" may be simulated by assigning a (nominal) ~~(token)~~ low pc to aborted trials.
(My mind is not clear on just what this may mean!).

.13 (5) [redacted] Say I have 1 second to work on "improving the P.D." If I had a well defined Goal for this, it would be a regular OZ problem. Now, I do not — the Goal is partially ordered. Is (such meaningful in this context)? Do I have to convert it to a scalar Goal? Also, "Horizon" is relevant here: one can "improve the P.D." in various areas: which are more impt. & how much more impt.?
But .05 & .13 is a separate problem area involving TM2. A MAIN

Q is to solve .02→.03. The "normal" workings of the system (= TM1).
One way to do this: Make a concept net, as in Sol 89. Some of the "conc" can be heurs.
I have in mind (apparently) 2 difnt ways of solving problems: One way, one assembles various concs (sequentially), to form the soln to the problem. As an example, perhaps the probs used in the early Search pgms (ANL), devising pgms, using a grammar of "subroutines" or of "functions".

.24 Another way to solve probs, is sequentially: One is in an initial situation, what to do next? One does it & one is in a new situation, what to do next? — loop to α

.26 ΣSN Another Aspect of .02→.03: To show that "Blind search" can → heuristic search, if P.D. is suitably modified. (perhaps this is what .24 is about!) (This is the F.N. of Sol 89, 15)
— so on/4 aspects.

[redacted]

C

I should be able to take any prob solving method that I can think of & show (via the "4 aspects") how TM would go about discovering that method.

.35 O.K.: Consider solving most linear eqns (w. 1 unk):
working backward: try to get eq. into form $ax = b$, then $x = \frac{b}{a}$.
Do xforms that put system in a "simpler" form o/o a form that seems easier to solve.
TM must have some concept of "simplicity" or idea of what forms of eqn. are "closer" to being in $ax=b$ form. So this looks like an AND/OR net problem.
The "parts" of this "heuristic" are of hy pc ... they are often used in eq. solving, & other probs.
Also, .35 is in the direction of GPS a fairly General prob solving method.

1·31·01 i'Bulg

Some weaknesses of 7.35 — .40: a) How did TM get t. idea that successive "legal" xfms on t. unsolved eq. is t. way to go?

b) In what ~~sense~~ can we say 7.35 — .40 is essentially guided by a P.D.?

c) What is t. Lsrch on? In 7.35 ··· ~~[illegible]~~ t. "cand", t. [illegible] 7.35 — .40 "pgm"?

(SN) Is Blind srch of [redacted] FNP5 of S89 discussed in Baum? — Near the end Near the end ! { Section 4.2 on "Levin". also 4.3

Well, first look at 7.35! In what way is it a "Search" problem? Well, we're searching for a seq. of operations that ends in $x = \frac{b}{a}$.

~~Esp~~ All Inv. probs can be regarded as "search for string that satisfies conditions"

In 7.35, ~~suppose~~ consider the "closeness to $x = \frac{b}{a}$" heuristic (or "simpler" form, etc.).

We seem to be "learning" during t. search, because when we find a cand that is closer to "$x = \frac{b}{a}$",

.08 we try modifying of it (i.e. GA. w. Mutation only). "Learning during t. search" makes it not Blind!

So how does this square w. t. Sol89 footnote?! Lsrch is (normally) for "Blind Srch" only.

What did t. FN say? — That any non-blind srch could ~~become a Blin~~ be xformed into an equivalent Blind srch "if one had enuf params." It's not clear how this is supposed to work: Maybe reread f.N. (By "Parameters" I may have meant "i.e. "Traces".)

Unless t. P.D. is modified during t. srch. — In which case, each cand is chosen "Blindly", but ~~each can~~ successive cands may have quite different P.D.'s.

If t. P.D. is different for every cand, then is Lsrch possl.? — or more exactly, Do we get expected cc for each of $\boxed{\frac{T_c}{p_c}}$? (or $2 \cdot \frac{T_c}{p_c}$).

Also, what about || (timeshared) Lsrch? — one doesn't really have completely seq. canonical trials in || srch. (Perhaps look at Li-Vitanyi Book for alternative Lsrch techniques.

.21 It is easy to Generalize 7.35 to a very common srch heuristic. We xfm t. Inv prob. into an OZ problem via t. "closeness" criterion, then use ~~[illegible]~~ proper Mutations-only ss in GA.

.26 Try this: Say t. Cand (a "trial") is characterized by t. trace. In all formulations as .21! The trace is a sequence of xforms. & t. pc of each xform depends on t. pc's assigned to previous xforms, so t. cand. traces are substrings strings in a stochastic lang. — so Lsrch is appropriate!

So cands are generated as follows: One has a finite set of xforms possible. At each stage in t. generation of t. trace, each xform has its own pc, & is a funct of "t. entire trace thus far" for that cand. This is certainly an ordinary P.D. to which Lsrch can be applied. This may have been t. idea of t. Sol89 footnote!

Woops! The pc's obtained as one generates a trace are not monotonic! — So they are not a real P.D. Could one do Lsrch anyway? Usually t. pc's ↑ as one goes along a trace, as opposed to normal generation of cands by stoch. Grammars)

-.01 Looking at P.N. §4, Sol 89 [Solomonoff]: "It is not diff't to prove that" ⊙: any heur Algm can be simulated in Blind srch,

.00 in a space of sufficiently powerful cands. A cand pgm must be permitted to have more args than

.01 Simply t. present prob cond'n: T. args may include any info obtained as results of a dvrg execution,

.02 of previous cand pgms.

' In .01-.02, I guess its t. pc of a cand. as a fnct of ① prob. dcrn ② other things.
≡ previous trial history

If t. P.D. is difrnt for each cand: ~~[illegible]~~ then Lsrch may not work!

Non-t.less, this P.D. that is updated after each trial does seem to be what many heurs do!

The seq. of trials is a progressive lrng experience.

In 7.35 ~~one~~ each xfrmn is/ usually ~~not~~ a trial at all — i.e. one doesn't expect it give a soln. ∴ Most xfms when executed, can be very useful as "Experiments": ~~a~~ Things done to get information.

Sol 89 my Santa Cruz paper on 6/13/89

Another Quote from ~~[illegible]~~ 2/2/85!

"Any Heur com or any other info should be insertable into t. pd, by assigning it a pc or an operator (w. a pc) that tells it when it is to be used."

"How to get better of heur 4,89 — so it was better Ran. 7/4/88
1:30"
40"
33"
43"
40"

Sent I thought been able to find t. summary of work out. proposed Hochet 1989. Jump in Search box:

.15 Well: $\boxed{\text{T. "Soln."}}$ In T. ~~[illegible]~~ the thing we are searching for is t. pgm that solves t. eq. This is always what INV Lsrch looks for. We are not searching ~~looking for~~ t. "trace" of t. soln. (The t. "trace" is a property of t. soln: it goes into a "slot" of t. soln. pgm.)

Finding & applying suitable heurs to get this pgm. is then t. main present problem.

It may involve Logical/Math'l reasoning ("non-probabilistic").

For ~~[illegible]~~ devising heuristics, any experience or t. past knowledge would be used by Humans.

Hvr, if we are to do Lsrch, Heurs can be derived only from info from "previous problems":

.23 If we also info from t. present problem, t. P.D. will change during t. Lsrch — which invalidates our CJS estimate.

Well, if we use the T←2T mode of Lsrch, any changes made between T←2T operating would not screw up t. Lsrch; It would only ↑ t. pc of t. soln. — presumably! — which is fine! So if we ~~find several~~ discover several modifns in t. PD during a T←2T "round", we do not apply these P.D. modifns, until t. next Round.

Hvr, since the T←2T doesn't occur many times during a problem solution — one could be rather limited in one's source of data for heurs. —

One help is to use T←αT w. α<2. Already α=2 is sub optimal (but only slightly so): α=3 is, I think, optimum. See how much worse α=1.5 is. (Making α smaller means the T←αT operation occurs more often (larger no. of times, per soln of a problem))

[The analysis for optimum α came about when I was using the T←αT method as a way to combat H.W. errors in large-scale parallel computer implementation of TM. The reference is within t. last 6 mo (I think).

Another possy is to restart t. Lsrch ~~[illegible]~~ whenever some v.g. (or big) ~~[illegible]~~ new heurs (modifns of t. P.D.) are found. (tho this might be too wasteful!), ~~Another poss~~

Another Possy: that it is unusual ~~[illegible]~~ for many (or even any) new heurs to be devd during t. problem soln. (?) — in which case we have no trouble here! ⟶ See 2.2.26 for More on changing P during t. srch.

2.1.01 1 Bulg

≡ "solution"

Of interest: At 7.35, I was interested in the implementation of Heurs in Lsrch: yet 7.35 is not exactly a Heur! It is a prob. that solves many kinds of problems.

So: finding out how 7.35 could have been devised, is now the BIG PROBLEM □

Actually 7.35 is a sort of Heur: it is a probsoln. that can be easily modified to solve a great variety of probs.....

Main Immediate Goal: To demonstrate (≃ prove) 7.02: first: that for INV probs, Lsrch is within a factor of ≈ 2 of true optimum "if all info is in the p.d."

Specifically, that any heur can be put into the p.d.: i.e. any heur is "learnable" by the system, if it is learnable by a human: otherwise the heur has to be inserted, or TM has to be helped in finding it via "Hints".

The 3 args that "any heur" can be expressed as a p.d. modifn.

1) Heurs are ways to speed up search — by reordering trials: This can be done by LSRCH changing P.D. on trials plus Lsrch. (5.15–19)

2) We can find out how to modify the p.d. by hypothesizing how the heur could be found, devising a TSQ (or a token TSQ) to get amount that it would change the P.D. (5.25–30)

(Also the heur can be obtained by logical/Metrical reasoning: (5.32–37 ... Needs to be worked out)
See 305.06 (Bulg2000) for some developments.

3) "All info is in the P.D." If a person has a better way than TM to work the problem, then that person has a heur that is not in TM's P.D. This assumes all Heurs are expressible as P.D. modifns.

4) The FN. of §4, Sol 89; [9(−01) — .02]: Hmmm I still don't understand this; I'm not sure it's correct!

.....:20 SN on: For some heurs to be "learnable": TM should be able to "watch" problems being solved, so that it can observe regys in the solns. Perhaps this observation gave rise to that FN in Sol 89! In General, it would seem that human based heurs could be based on any kinds of regularities observed in Problem Solving — so all of this kind of data must be available to TM if it is to be able to devise (almost) all heurs.

GOOD!

I think FN of Sol 89 is impt, but one must study .20 to understand why!

[illegible]:

2-2-01 1 Bulg **QUICK ABORT** — a Generalization.

for SEARCH of solns. to INV. Prob

Heuristics/ a/o pgms (codes), modifns of such pgms. that make them less expensive in time a/o memory.

They are ~~of~~ of 3 kinds

.02 1) leave cand. pgms of the search invariant but vary order of trials (only). ⟵ such is in this class.
Not doing ⟵ such would mean e.g. doing trials in Lsearch order w. fixed time cutoff for all trials, or Random ← trials w. fixed time cutoff. (find more interesting examples: this is the only type of heur I can prove the "x4" efficiency for.

.05 2) Leave order of trials the same but speed up some or all of the trials. Simplest examples:
a) Get faster machines. b) Quick abort: [trials themselves are modified but order of trials & invariants to be least cc.

This is an OZ problem

Example of Q.Abort: Say problem is to find short codes for string, S, using UTM.
Not using Q.A. Wait till ~~entire output or~~ machine stops before comparing output bits.
Using Q.A. reject trial as soon as 1st out bit deviates from that of S.

.10 [This is not a necessary heur: one could continue on till a certain ~~fraction~~ or > a certain no. of output bits were wrong: If < that no. were in error, then make code by adding in bit string telling where errors were: This can be done in (at least) 2 ways.

The never fraction would ↓ as we get better "Best" results for the search.

.14 See (~ recent) work on ZIGL: Re: Gerry Wolff:

Other examples would first examine parts of output likely to be in error, & reject them if they had more errors needing correction than the Best Cand.

Thus far

.18 3) Mixture of 1) & 2): Example/ for OZ problem, here. We want to find linear regression coefs a, b ∋ $a x_{t-1} + b x_{t-2}$ is as close as possible to x_t; for $t = 0$ to m (using Σ sq. error criterion: Non-heuristic way:

~~Do search~~ Do exhaustive search over a, b plane: start w. low resoln & $-R < \binom{b}{a} < R$: for successive trials use larger R & higher resolution (e.g. double R, halve the grid size — this mults by 16 no. of cands in each round!)

Heuristic of type 2: For each round, find best cand & its ms error.
For next round limit region in a, b space to best approx ± ΔR that gets value < best approx.

This is also an OZ problem

Unless the problem is to find the map R.

.27 A type 3 heur: find a, b by solving linear eq. involving correlation matrix. It's not clear that this is type 3: It goes directly to the optimum point, so we may regard the order of trials being as in the non-heuristic case but it eliminates all but one trial.

In all these methods there is an extra amt. of cc involved in computation — particularly in the linear eq. soln. "heuristic".

The method of .18 can be used for non linear regression, to narrow down solns in a, b space. Then when we are close enough, use linear approxs to get successively closer.

Another type 3 example: Using many Machines in ||: This usually changes order of trials as well as speeding up all over process.

Acct no.
Note a phone
Beacon
3391096745
Password
9792

2.2.01 1Bulp $\boxed{\text{QuickAbort}}$ → Generalization ~~genzn.~~

So far, I'm only saying that a hour is equiv to modifn. of the P.D. if the hour is of type 1.

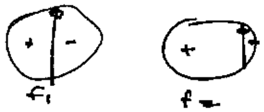
It may be possl. to show for ~~some~~ some type 2 or 3 hours as well: I don't know.

(SN) on Lsrch: Consider non-Lsrch of 11.02–.05: How bad is it? Say we do trials w. time limits T & 2T for successive rounds, but not Lsrch: Say we want shortest code ~~for~~ string s. We do all trials of length |s| to start. As soon as we find a code of length ≤ |s|, this becomes our new limit. If the shortest code is of length |s| − 10, (it would be a v.good SM code) ~~[illegible]~~ this ~~[illegible]~~ would ↓ cc a lot but still cc would be enormous.

Q: Is the search method of 11.10–.14 a "heuristic"? It would seem not to be a INV problem heuristic, because the goal is unchanged. Essentially, it's an OZ problem, anyway! In fact, most of the examples have been ≃ OZ problems

The problems in 11.27 could be made the soln. of 2 eqns in 2 unks.

Q: if $f_1(a,b)$ & $f_2(a,b)$ are both continuous, & both change sign in region R — Must there be a pt. in R where they are both 0? No.



For linear set of eqns, ~~[illegible]~~ if there is a soln, it is unique, so $\sum_i (f_i(\bar{x}))^2 \to 0$. One can find regions w. low Σ on the surface,

type 1 search (11.02) for INV problems Given f(·), to find x ∋ f(x)=a.

If f(·) is a UioMachine, Lsrch ~~[illegible]~~ using P.d. on x (P.d. is a funct. of (f(·),a).)

If the P.d. is difrnt for each a, then when one finds a pgm that gives x, it will not nearly give x for a difrnt a (!).

Say one is looking for a pgm p(a) ∋ ∀a, f(p(a))=a (when f(x)=a has a soln. ~~[illegible]~~)
f(p(a))=a

Trouble is, one can't verify that f(p(a))=a for all a (!).

See p.504 in 1997 Li-Vitanyi. (P503 ibid) lists a few interesting INV problems. 40

Using L.V notation, we want to invert φ(·) given x we want y ∋ φ(y)=x

~~If we then use Lcost(φ) only (Lcost is a funct of φ only),~~ Say A is an ~~[illegible]~~ inversion Algm

Then if we use a P.D. on A, as a function of φ alone, The total time to find y will be ≤ $\frac{T_A}{P\text{cost}(A)}$ This will be true for any x.

if ~~If~~ our P.D. is on A as a funct of φ and x, and A exists, then

$\frac{T_A}{P\text{cost}(A)}$ will not be a constant, but Pcost(A) will be a funct of ~~[illegible]~~ both φ and x. If we ~~[illegible]~~ inversions for x = 3, 100, 150, as a t.s.q., then ~~[illegible]~~ Pcost(A) will be, presumably quite large for x = 200 — based on the similarity (if any) of the solns. for x = 3, 100, 150.

2.3.01 1 Bug: Quick ABort

What I want is a good set of exemplars for each of 6. 3 cases of 11, 02, ,05, ,18.

2.3.01 ↑Bulg HORIZON problem   Partial v.s. linear ordering of P.D.'s → (31)

The way ~~is~~ to not solve the horizon problem:

For a horizon of H, at time t, TM acts as tho it had to ~~[illegible]~~ get ~~max~~ max total R.(reward) for (interval period) (t, t+h). "h" is the horizon. At time t+1, it acts to optimize (t+1, t+1+h). At time t+x it acts so as to optimize yield for (t+x, t+x+h)

If it did this it ~~would~~ would always work on "self improvement". It would never work on actual problems. It would always do S.I. first in any interval of length, h, since this would help more in solving main problems later. → (.22)

.05 Dal (S.I.) (S.I.)

A possi. way to get a finite horizon, is to spend a fraction f of one's time on self improvement. The effective horizon is then (maybe) ∝ $\frac{1}{1-f}$.

.09 → But "∝" ("proportional(")) to what? ~~[illegible]~~ How do I get "dimensionality" "scale" out of this?

Anyway, (in general), it would seem diff. to distinguish betw. normal problems & "self improvement" problems. Hvr., in my own ~~formalism~~ formalising, we have look for finite present problems using the existing P.D.; "Self improvement" consists of "improving" the P.D. — of finding shorter codes for it (mainly) a/o other means of "P.D." improvement.

Since the Goodness P.D. is ~~[illegible]~~ only partially orderd, this is not really a "well-defined problem". ABCDEFG

.17 In .09, one does not get a "horizon factor" what is obtained is a "slowdown factor" of 1−f in work on the present problem. If it would ~~take~~ take time T_0 to solve w.o. self improvement, it will take time ~~[illegible]~~ ≤ $\frac{T_0}{1-f}$ to solve it w. self improvement fraction, f. "≤" because self improvement will tend to ↓ ~~[illegible]~~ cc needed for the problem.

.22 An arg. v.s. that of (.05): Working on main problem ~~[illegible]~~ a bit before S.I. could help "direct" S.I. — ~~[illegible]~~ The P.D. is a "Big" object. One can improve "parts" of it. Working on main problem could tell what "parts" to work on.

But even doing it this way, TM would never finish working one problem!

ABCDEFG
abcdefg

.27 On the partial ordering of P.D.'s w.r.t ("Quality" "Goodness"): 1) It is good for the P.D. to give hy PC to the corpus 2) It is good that it evaluate this PC rapidly.

.31 ? [So ∴ Partial ordering because of 2 Goals: (Can we unify the Goals by using $\frac{PC(corpus)}{Time(to\ evaluate\ corpus)}$? Essentially Min Cost for the corpus. ($\sim K_T$.)] ?

Well., .31 doesn't seem adequate; the P.D. is a Cond. P.D. Perhaps min total K_T K_T for all of the problems thus far? MIN total K_T for all probs thus far? If corpus was only INV probs.

Perhaps look at stuff on "TM$_2$'s Goal". Maybe from that Saarbrucken Box. → See 20.25 for Review of this problem w. a sort of Soln.

2.4.01 Bulg.

Major Immediate problems;

.00 1) A good set of examples for each of t. "Q. About" ~~probs.~~ of | hours | 3 types of hours (11.02, .05, .18)

(These are not usually Q About probs, but they do make clear, t. kinds of hours that occur in R.W.

.02 2) In particular, 11.02 is, I expect, t. main kind of hour (change of ordering of trials, only).

& so I want lots of examples to work on — to see how TM uses these hours — how it xfms them into Modifns of t. P.D.

3) ≈ "TM2's Gore": In view of MCT, what is a good ~~Gore for t.~~ main cond'l. P.D.?

~~A~~ Large P.C. for its corpus data thus far; — but also quick evaln. is impt. 14.27 has some ideas.

First consider a ~~[illegible]~~ corpus w. only INV. probs. Say p_i, T_i are t. pc & soln time for i th. problem; Then total soln. time is $\sum \frac{T_i}{p_i}$. If t. T_i's were indep of t. P.D. ← NO! then we want to minimize a wtd sum of $\frac{1}{p_i}$'s. We could modify ~~[illegible]~~ any of t. p_i individually or in ~~[illegible]~~ subsets. Perhaps we would get most mileage by

.12 working on p_i's of probs w. large T_i's (??) → 17(.01)

Some P.A.'s are very slow (like ALP is slow — but u.g. f.c. is fast)

Next, consider a corpus of OZ problems only.

.14 4) L cost v.s. [redacted] T/pc (16(.01). 16.08 → [redacted] gives t. dirty s

(SN) How does Soy, ac6 t. AAE problems & UA B.T. into All of PC

They are probly related to TM2's Gore.

T. TM26 problem.

5) Each of these problems has 2 aspects: ① a bottleneck in getting TM running.

② Tutorial: explaining how system works to people.

items #1 (.00) & #2 (.02) are most impt. bottlenecks

Item 3) is of next import: The rich & dirty solns. may be available & adequate.

Also of much import is applica. of Lsrch to OZ problems.

1. Most impt [redacted] problem is to understand how ~~[illegible]~~ all hours of type 1 (ordering) can be ~~[illegible]~~ used to modify P.D. for Lsrch. — for INV probs at first; then for OZ probs.

I want to understand this & so want to find lots of examples.

2. Next in import: to understand TM2's Gore.

2.4.01 1 Bolg

$\boxed{K_T \text{ v.s. } pc/T}$ $\boxed{T \cdot 2^\ell \text{ v.s. } T/pc.}$

.–01 Th. problem involves t. first Gambler's House Thm: That if p_i is t. proby of success & T_i is ~~[illegible]~~ t. T cost of a trial, then best order (least E ~~[illegible]~~ ET) is $\frac{T_i}{p_i}$ order/ smallest first. In general [redacted] this will not be t. same as $T \cdot 2^\ell$ order, but I think that if $p_i \approx 2^{-\ell}$, it is ruffly correct. The disparity occurs when $p_i \approx 2^{-\ell}$ is a poor approxn. p_i will be sum of codes, which can differ much (I think) from $2^{-\ell}$.

(Normalization is not relevant, since we are only interested in ordering ~~[illegible]~~ — Hvr, since t. probys are not normed, GHT1 gives a result that assumes they were normalized, so ETS ~~[illegible]~~ will be "<" $T 2^\ell$ because $(2^\ell)^{-1}$ should have been normalized.

.08 In general, t. errors are several!

First, is t. order of trials about rite? A factor of 2 in ~~[illegible]~~ soln. time (for t. $T < 2T$ method) is acceptable.

.11 Second, Is $T 2^\ell$ a good estimate for cc of soln. (~~[illegible]~~ Expected value)?

Actually, it would seem that one should be able to get expected cc of Lsrch w.o. considering "probability" or GHT~~[illegible]~~. In fact Lsrch does ~~[illegible]~~ take ~~[illegible]~~ time $\sim 2 \cdot \frac{T}{2^{-\ell}}$ to find its soln.

I've written a lot about this problem, but I really don't know where any of it is.

Perhaps after making a good list of critical ~~[illegible]~~ unsolved problems: Go thru my notes looking for relevant work.

So my problem is TSQ for Inv & OZ probs: Assoc. w. this is understanding just how Lsrch works to solve t. problems — using heurs to modify p.d.

Perhaps t. Diffy is this: Lsrch really has ~~cc~~ bnd of $2 \frac{cc_i}{} 2^{\ell_i}$.

But optimum srch would perhaps (like GHT1) be bounded by $\frac{cc_i}{pc_i}$ which is sometimes much smaller (forgetting t. factor of 2, $\frac{1}{pc_i}$ is usually ~~[illegible]~~ $\ll 2^{\ell_i}$ (?) (is "$\ll$" t. rite word?)).

The "proof" that Lsrch is optimum within factor of 2 expects Lsrch to ~~[illegible]~~ cc to be bnd by ~~[illegible]~~ $\frac{cc_i}{pc_i}$ & this is (perhaps) $\ll cc_i \cdot 2^{\ell_i}$

Also note t. diffy ~~[illegible]~~ of 17.12–.40: It has t. "same simplification". A human, say, would notice that a bunch of cards were "above t. snow" & use this idea to ↓ cc of search (a lot!).

.21 → Could a hyper order heur (Meta Heuristic) deal w. both (16.–01) ff & (17.12–.40)?

Perhaps 24.30 & 39.25 are examples of this!

.01 T. maine Goal here is to "improve t. P.D." Because a P.D. has both CC & PC in it & because certain parts of t. P.D. are more important than others, t. problem is not "simple".

E.g. a Naive soln, would chose a P.D. for which t. PC of t. corpus was Max.

t. General ALP p.d. does this, but takes too long.

.03 Another thing I may have forgotten about t. OZ aspect of t. P.D. — that t. ~~[illegible]~~ Min P.D. does not give t. proby [redacted] that a given OT will be best for a given problem;

.05 It gives t. proby that any particular ~~[illegible]~~ O.T, α, will have Gore = G, w. CC = β, for problem P.

.06 [From this info, plus work on SOY, AAS, SMA, etc, we get t. p.d. distribn for over a continuum of O.T.'s.

.07 α being best for problem P & cc = β.

Going back to t. (1998?) work on MCT, we knew a p.d. for various args! T. P.D. →

One kind of args. is that of (.05). (It) may have the p.d. for arguments in (.06–.07) but that part is "Mathematically redundant" since it can be derived from t. P.D. aspect of (.05)

.11

NB, .11 is true both for OZ & INV p.d.'s in ~~[illegible]~~ Both cases we want p.f. for t. best soln. See (.37)

.12 A possl. diffy in using Lsrch for OZ probs: Say there are a bunch of OTs that are (looks like a legit diffy! See (21, 22R) also 23–25) very similar (so that success using on is hyly correlated w. using another on t. same problem). Then It would seem best to spend all one's time on one of them rather than divide it up (about equally) ~~[illegible]~~ among t. set of OT's.

I ran into a similar problem w. INV problems! My impression that it turned out not to be a problem! I don't remember why hvr.! It may have been calld "t. Flat D.F. problem". I think I went thru The Mechanics of an Lsrch, & it ended up Making very little difrnce!! (Hvr., I don't See just how this can Be!)

24, 30 Suggests what looks like a partial partial, legit Soln.

.21 OH! I may see it; On t. final round of T ← 2T we spend our time on t.

.22 first cand that works, & we don't do t. others!

If there are n cands ~~[illegible]~~ each of ≈ p_{c_0}, T_0; we save time $(n-1)T_0$. $T_0 \to \frac{T_0}{p_0} - (n-1)T_0 = T_0(\frac{1}{p_0} - (n-1))$: But usualy $\frac{1}{p_0} >> n-1$ so its not too impt! So (12:15) a problem

.23 What about t. time wasted in ~~previous~~ previous rounds? Also, if there were 8 cands w. ≈ P.C. & all about t. same, then if we poold them all into one, it would have 8 times as much pc,

.25 and it would be found 3 rounds earlier & $CJS \leftarrow \frac{CJS}{8}$ ← This is Correct I'm afraid!

So it may well be that My olde "PROOF" of insignificance was Wrong! — but it would be well to find it & Make sure! ← looks like it!

Anyway, t. "PROOF", if CORRECT would apply to OZ as well as INV. probs.

If one could Somehow recognize that t. Cands were very similar — this

.30 would perhaps help.

T. fory. Seems Similar to t. ~~[illegible]~~ "pc v.s. ∑ trust" problem of [16.(.01)].

.32 A way of ~~[illegible]~~ pc that is not taken.

.33 Another relevant item of investigations That we often have PD's that don't ∑ to 1 — they could sum to >> 1 — Like t. pc that one of t. Pgms ~~will solve a~~ will solve a certain problem! Perhaps all of them will! I did finally find a way to deal w. this! That somehow t. P.D. was ~~[illegible]~~ always defined so that it was a true "P.D."

.37 ? (e.g. In INV probs, t. pc ~~E by~~ pc(x) t. cand. is t. proby that x is t. ~~[illegible]~~ (single best soln to t. problem)) ← (?)? "set cands are mutually exclusive".

.37 is certainly (?) true of t. ~~[illegible]~~ P.D. for INV probs. Also in OZ probs.

2.5.01 1Bub Cure Cancer (.20)

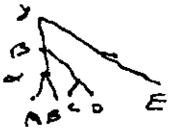
Some General pts:

~~Tot~~ (.01) Re TM2G problem: Using ≈ wts for all parts of the $\underset{\sim}{\text{GPD}}$ (Grand PD) they be an adequate default soln. There remains the problem of partial ordering betw. pc & cc: Some of the Least $\left(\frac{cc}{pc}\right)$ .01 ordering is adequate. Look at examples. → (.07) → 19.05

.02 ⊖→ Another Big unsolved problem is TM's use of math/logic in (discovery & obtaining) heurs

Good! One way I considered was that TM first learns lots of Math; Then finds analogy betw. .04 problems in its "RW" (i.e. "Real" the problem of Heuristic construction) & the Math'cal world!

.07 (.01) Re: TM2G! In addition to the cc v.s. pc problem, the pc problem has at least 2 levels: One can work on a particular epistemic area (narrow updating) or one can try to ↑ pc's of ≈ "all problems". This can be done at various levels. We may think of the problems ~~as being~~ being leaves of a tree.



.13 Say A,B,C,D,E are problems. By working at "level α" we find abss. common to the pc's of A & B. At "level β" we look for abss. common to A, B, & C, etc.

We might approach this ~~by~~ differently at "β" by regarding "A,B as a single problem" "only" and C,D as another "single problem" — so β has to unify (to some extent) these 2 "Macro" problems. [So Improving "pc" can have many levels: Also, one can work on 2nd order branch, if one suspects it will be useful in the future. → 19.05

.18 On the "Cure Cancer" problem: 2 general methods:

.20 Good! ⊖→ 1) ≈ Dynamic Pgmg: Consider all possl. sequences of (experiment, result) pairs. This may be considered to be a "bigly branched" tree i.e. many branches at each node (actually continuous). At each node, there is an expected yield (E Gain) for the branches from that node. (This Gain is hard to compute, but probly "dynamic pgmg" is relevant). From each node, one choses the next node that has Max Gain.

2) Simplistic Expert Reasoning: First, I want to understand Cancer, so initial Experiments will be on that subgoal. (How this is an approx way to do .20 is unclear!)

In .20 the problem would be much simplified if the pc of each (experiment, result) were indep of what occurred in the past. This independence may be the "usual" case in Dynamic Pgmg. In present case, the pc of each (expt, result) is very much a function of what has occurred previously in that set of (sequencial) (expt, result) pairs.

Actually, .20 is like feeding random Seqs into a UIOM: The output string is the seq. of (expt, result) pairs. However, there is a big difference!

In .20 the seq. is $(exp_1, res_1)(exp_2, res_2)$ etc. While the pc's of all res_i are a funct of all previous exp_i r_i pairs; the exp_i ~~are~~ do not have pc's assoc. with them. My impression is that it is a very complicated problem: Maybe like chess — The Q is, is there an alpha-beta pruning method?

19.(-.01)

There are 2 solns. to "cure cancer" ~~But I wrote~~ [illegible]
The 1st one is much shorter: This may be t. short soln.

2.6.01 'Bug: "Dynamic prog": T. "cure cancer" problem: Solution .26 -.40

of "state"

.01 : 18.40 : Unlike chess, | But evaln. depends on past as well as distribn. of possl. futures | like chess (in one version of problem) t. length of t. game
" " t. opponent is not malevolent. | is unknown.

Oneway to simplify t. problem: T. Goal: to have, after a fixd k moves, max proby of cure or some other ~ Cure, or max % curable for <$20/day or whatever.

~~[illegible]~~ Probably reading Marcus H. paper (or 2 papers) will help. — Probably: even if his approach would

.04 be wrong! (.16) ← Looks like an adequate "Formal soln" to "fixed length" problem. (.16)

.05 : 18.18 (TM2C) : Int figure of (18.13 L) : heurs at α affect both A & B. so $\frac{T_A}{pc_A} + \frac{T_B}{pc_B}$ is ↓

.06 ~~Heurs~~ ~~[illegible]~~ Hyer level Heurs at β affect A, B, C, D, so $\frac{T_A}{pc_A} + \frac{T_B}{pc_B} + \frac{T_C}{pc_C} + \frac{T_D}{pc_D}$ is ↓ —

So as t. heurgets of hyer order, it's more likely to ↑ t. allover Cure, a lot!

How, hyer level heurs are harder to find than low level ones & t. hy level ones may not be ~~[illegible]~~ applicable as often.

General Conclusion: That "Improving P.D." is a rather complicated thing: That one can do it in many difrnt ways. However, since OZ probs. become like ~~[illegible]~~ INV probs (≡ Induction probs) ~~[illegible]~~ Via MCT (See 17.03-.05 (+.06,.07))

There is only one P.D. & perhaps t. $\Sigma \frac{T_i}{pc_i}$ criterion is o.k.

.15 → Tho: A long Time Series w. lots of data, could get a lot of wt. this way! → 20(.01)

(.16)(.04) Consider t. 1 step case, then t. 2 step case.

1 step: e_i, r_{ij} (expt, result pair). We chose e_i w. best "Expected" r_{ij}.

We have, in this case, some simple way of evaluating r_{ij}. T. proby of $e_i \to r_{ij}$ will depend on entire ~~[illegible]~~ previous history of e, r pairs. — as well as "Expected value" of r_{ij}

[r_{ij} is not a number, but it may have nos. in it; ~~[illegible]~~ T. "expected ~~value~~ Utility" of ~~[illegible]~~

r_{ij} is defined to be t. mean survival rate for cancer patients after that result & t. previous ~~[illegible]~~ seq. of e, r pairs]

So, for One step, t. soln. is clear: & each 1 step "situation" has an expected "value" Utility (Assuming one made optimum choice)

For 2 steps, each possl. ~~[illegible]~~ e, has ~~[illegible]~~ a d.f. on r. After this r, ~~[illegible]~~ across, we have a 1 step situation, & we know its expected U(tility) — So

.26 (See 18.20 for cancer approach) Good!

Starting over: "A history" is a seq. of e, r pairs starting from t. beginning.

Each history has a "1 step utility" This is t. largest utility of ~~t.~~ all t. e's that could follow it.

(or t. U of t. e w. max U that follows that history)

~~[illegible]~~ On 2 step utility: Each history can be followed by an e_i, r_{ij}.

For each chosen e_i, we have a ~~[illegible]~~ D.F. of histories, (as t. r_{ij} is generated)

Each of these histories has a 1 step utility, so that ~~[illegible]~~ chosen e has an expected U.

We chose t. e w. max expected U; this is t. 2 step Utility of t. initial history.

.33 [In a similar way, we can define n+1 step utility from ~~[illegible]~~ knowledge of

.34 n step utility. Th. n step utility is ~~assoc. w.~~ assignable to any seq. of e, r pairs.

.35 0 step Utility is U of a history that has ended. It mainly depends on t. last r.

So .33 -.35 define it recursively.

It may be better heuristically to define n step Utility using "h-n" where h is t. "final" no. of e, r pairs ("horizon")

.40 (N.B.) t. preceding soln. to t. Cure cancer problem is for fixed horizon. ∴ Not complete soln, since "horizon" is usually open" the reference machine as a way to accomodate new things that the system has learned.

Second: The method of computing probabilities of strings may seem a bit arbitrary, - and for several years after I thought of this method, I wasn't sure it would work - wasn't sure about the details of the method. However, I finally did work out a proof that the system was very accurate is estimating probabilities from empirical data.

Third: It turns out that if we use a universal reference machine, Sigma sub i 2^-D sub Bi is not computable in a finite emount of time. We can't even be sure that we've found the shortest code for B! The reason is *not* so much that we can't try all possible codes: - Very long cods don't contribute much to the sum. The main problem is that there are certain short srings that one put into the machine, and the machine runs and runs, and after a long time, we still don't know if it will output B and stop - and there is no sure way to tell. That Algorithmic Probabilitiy is incomputable may seem like a strong argument against it - but it is *not* - this incomutability is an essential part of probability - of science itself.

The short codes that contribute most to the Sigma sub i 2^ -D sub Bi correspond to string regularities in the data. Consider the string (01)^1024. We can easily program a machine to write that sequence by simply telling it to write 01 , 1024 times - which takes 2 bits to say "01" and 10 bits to say 1024 plus a few other bits - a vast compression over the 2048 bits in the original string.

In general in science, when you have a batch of data and you are looking for regularities in the data to be used for prediction - you can never be sure that spending 10 more minutes hunting for regularities could not give one much better than the best you've found yet.

The corresponding thing occurs with sequences that seem to be "RANDOM" - i.e. no descernable regularities. It is believed by many, that stock market prices are random - yet there is no finite amount of investigation that could be convincing on this question. On the other hand, once a strong regularity is found, it becomes very unlikely that the sequence is random.

The problem of induction is to obtain good approximations to infinity over Sigma sub (i=1) 2^(li). This is done by (Sigma sub i 2^(li) summing over not all of the codes). P' is an approximation to P. The more codes one find, the closer P' is to P. To maximaize P' by finding as many codes as possible in the available time, is a time limited optimization problem. It is equivalent to having a minimum error (i.e. P-P') in one's estimate, P'.

The inductive inference problem and its solution as a time limited optimization problem are very important in the learning System that I'm describing. One reason is that the problem of updating the probability

(=.01) : 19.15: TM2G : We might spend an enormous amount of Time on a SM problem (Sort of Separate from the Main TM program) — Maybe use a difrnt Computer. So maybe Give this aspect of the P.D. less wt? On the other hand, if this aspect of TM's work is able to bring in Money to Buy more cc. We should give it more wt.! So its not clear that "= wt" is a reasonable default...... On the other hand, when TM earns lots of money, the $T_a \cdots T_b$ in (19.06R) all ↓ — so all aspects of TM are "improved" — even tho TM has no new concs, no new heurs in those "Not SM" areas.

(*) One approach to TM2G would be in (19.06R) Give each [illegible] $\frac{T}{p_c}$ its own WT.

.09 These wts have to be assigned by User. Might be a lot of work, but probly very impt.

.10 Maybe its not necy to get the wts "very exact" but try for factor of 2 or 4 "accuracy".

for Sense Criterion is probly better TM2G crit. 57.01

.12 REV Brief Summary of 3 diftys w. Lsrch (actually only 2!)

1) The $T \cdot 2^l$ v.s. $\frac{T}{p_c}$ difty {16(=.01) — .40; 30.57 may be interesting (to look at the) — Looks real (I did some work on a MtCarlo way to deal w. this)

.15 2) Many Conds very Similar & correlated w. the Soln cond! (19.25 & 24.30 may solve it; 17.12 — .30 Looks real. See 24.30 for a good idea on this)

3) Conds could form d.f. that is not a d.f. (17.33 — .37) Seems not a real difty

.16 In ① & ② One difty is that they imply that Lsrch need not be as good as a Srch Guided by more powerful heurs. (24.30 may help here!)

N.B. (6.3) as a possl. Genl. idea on how to fix these diftys.

Other work of impce

① Formal soln. of the "core Crucial" (for search Admin. problem) for fixed horizon only

19.26 — .40 is final soln, but it starts on 18.20. This soln is only "formal": the computs. & approxns. have to be discovered by me a/o TM.

Also, it is for fixed horizon only, & many srch pbms are more of unspecified duration: The idea is to solve it as soon as possl. (like chess) rather than get as good a soln as possl. in a fixed time ("Anytime" problem v.s. OZ problem).

An approx soln: Start w. horizon of h. when h is reached decide on new horizon — maybe h into future, etc; (or 2h more time, then 4h more time etc.)

.25 ② TM2G: Realization that the P.D. has difrnt parts & one usually is more interested in some than others. — More "WE"

Also a possl. resoln. of the partial ordering of CC v.s. $\frac{1}{p_c}$. TM2 because of (14.27 — .40) 17(=.01) — .11; 18(=.01) — .01; 19.07 — .18

19.05 — 15; 20(=.01) — .10

③ On use of Math, logic in Heurs: (18.02 — .04) Used "Analogy" to solve Heuristic problem and that that TM "knows".

④ RE TM2G: We want to order the trials w.r.t. some Goal. There are some diftys in doing this: SOY, AAE, SMA are all long term projects concerned w. this problem problems

⑤ [illegible] 15.00ff gives major problem TM project: It looks like we are "back to square 1" — That TSQ is main problem: But review of status of other ~~problem~~ impt. problems in TM was important. Also the recent work on "Quickstart" (11.(=.01)ff) is quite impt! It describes 3 classes of Heurs. A tutorial (& practical) problem is to get lots of good examples of each!

I do need examples (particularly of class 1 (reordering of trials) to explain to Jurgen & Marcus why Lsrch is ≥ "optml".

-.01 ⑥ Also, remember the 3½ arguments on why Lsrch is ≈ optimal.

ⓐ [If ~~[illegible]~~ the critic has a better way to solve a problem, then the critic's heur is not in THIS P.D.

ⓑ Most Heurs are reordering of conds, so they can be expressed as Modifns of the PD used in Lsrch

ⓒ To find out how to Modify the PD in accord with Heurs ∴ devise a tsq. that could have implied heurs. This TSQ can be then used to modify the ~~Heur~~ PD ~~in a similar way that the Heur should.~~ This TSQ is sort of equivt. to the P.D.

NB This may not be true of Heurs obtained by Math'l/logical reasoning

ⓓ ~~[illegible] but Maybe not : [illegible] argument~~

Footnote in Sol 89: Since Heurs can be induced from any data, problems, problem solns. of the past, all aspects of these things must be able to control the (Grand Conditional) P.D. — So the Grand Condl. P.D. must have lots of poss. arguments. — These last arguments referred to in the FN of Sol 89. (9.01 ff is a quote of that FN

⑦ ~~[illegible]~~ The definition of the object that Lsrch is looking for is ~~[illegible]~~ a PSM. See 9.15 ff. Also note the proof of the adequacy of Lsrch: in Levin (discn of "K_T") we are searching for a PSM. Also Note! For both INV & OZ probs; ~~[illegible]~~ the PD is for the Best soln: ∴ The conds are always Mut. exclusive (except for ties!) ∴ a regular Proby distribution.

⑧ Modifn. of the P.D. during Lsrch (affects on the Lsrch are not too bad) 9.15-.40 (This diffy was the main line of inquiry in 23.08 — 25.23 : A tentative "soln.":

2·7·01 TM IBulg:

.01 Examples of Heurs (How most of them involve reordering of trials),

.00 1) We have a formal (composition) lang. to express our pgms. By regarding the set of useful pgms as a Stochastic lang., we assign pc's to the various pgms: we "order them" (.01)

(SN) ... Note we are interested in a Cndl p.d., so the data is the unordered set of (prob dcrn$_i$, soln$_i$) pairs (note it's not a "Bag": each pair occurs only once.); Maybe we should have as data (prob.dcrn$_j$ soln$_j$, cc$_j$). We obtain from this a pd. and then we do an "optimization integration" to get the p.d. ~~used~~ to be used for Lsrch.

.06 This (.00 ff) really unifies the INV & OZ functions of the Broad Cndl. P.D.

An Inv. prob is solved with a pgm — each pgm has a cc.

An OZ " " " " " OT — an OT " " Gave for a given cc. Or a cc for a given Goal/cutoff. ← This makes it closer to the INV problem.

We can use as empirical data for OZ probs, (problem dcrn, g obtained) → cc needed. So the prob, g obtained for OZ problems corresponds to the ~~simple~~ "prob dcrn" in Inv. probs

.11 (.01) Also, we define substrings, subtrees & assign pc's to them; which also involves changing order of trials,

But now: find ordinary Human heur & show how the a base Stoch lang. can express its ordering changes.

Go thru back on Heurs (Pearl) & the Encyc. of A.I. to find Heurs & express them as P.D. modifications.

A trouble with .06. Most Heuristics tell how to srch ~~[illegible]~~ for soln.,

16 · They do not go thru intermediate steps of estimating the prob'y that an arb'y pgm

17 · will solve the ~~[illegible]~~ given problem in time ≤ ... Well, the ~~[illegible]~~ System I'm using does do it that way! So the pd. that we directly modify is that of .16→17.

The P.D. used to guide Lsrch for the ~~[illegible]~~ next problems is obtained by optzn on the p.d. of .16→17

Heuristics: Going thru ~~[illegible]~~ Pearl's "Heurs". The first 2 heurs convert an INV problem to an OZ problem (This is perhaps the most common type of heuristic for INV probs) First 2 heurs are of this kind.

Vol I AI encyc: Chapter I on the Heur search

Heuristics often guide search by results of previous trials. This sounds diff't to put into form of "Blind Srch" — unless the pd. changes during the srch.

· 26 See .8.08 → 9.40, 9.23 → .40 in particular, on how probs can vary during Lsrch, & yet Lsrch will work: The main problem is: (a) we can't estimate pd, the pc of the soln, since it depends on the trials that preceded it (b) Since P of the soln. varies during the search, it can ↑ or ↓.

If it ↑ (which it should, if the heur is any good), then we could discover it via Lsrch, much later than $\frac{T_j}{P_j}$ (T_j & P_j are final T cost & Pc of the soln), i.e. we discover the soln. when the T for the round is $>> \frac{T_j}{P_j}$, because earlier we got to that "T" round, $\frac{T_j}{P_j}$ was $> T$, but just before that round, P_j increased, so $\frac{T_j}{P_j} << T$: In this case the time, ($\approx 2T$) to soln. will be $>> 2\frac{T_j}{P_j}$.

The large ? occurs if there is a large sudden ↑ in P_j before the final round.

It's not clear whether or not this is likely!

SN .00 could be very relevant to SN. It was not clear in my mind as to whether "apriori" was for!

Good

(-.01) On searches in which the P.D. is modified during the srch: ("Learning from earlier trials in the srch") ....

.01 In general, TM should be able to detect all kinds of regys in its own searchs, & use them
.02 to improve future searches: The diffy is, that these regys will not always take the form of a modn. of the P.D. followed by ~~pure~~ Lsrch ("pure" means no modifn. of the P.D during that Lsrch.) (There is not a diffy! ~~This is not a diffy~~, since the Lsrch is over the regys, not the searches governed by the regys. ~~Lsrch has gotten by the regys.~~)

.05 Well, say the pd is → input: descn of problem : output → p.d. over pgms to ~~attempt to solve~~ attempt the problem. Such a pgm could consist of a search over trials ~~in~~ which ~~[illegible]~~ depended on
.07 results of previous trials. — It would ~~be~~ include QuickAbort or any other heuristic trick!

.08 A Q is in .05, would one ever want to change order of heuristics tried, in view
.09 of what happend in earlier heuristic pgms (in same Lsrch)?

Another Big Q is: Can we insure ≈ optimality? ~~[illegible]~~ e.g. That the search will usually take not much longer than $\frac{T_j}{P_{cj}}$? Changing ~~[illegible]~~ pc during Lsrch screws up this estimate: (See 22.26 & ≈ 8.08 – 5.40; (9.23–40 in particular)

Superficially, it would seem that .05 should be near optimum, in view of .01–.02

Would I want to do a Meta search ~~over methods of~~ search techniques for finding heurs to find pgms? (& Meta$^{(n)}$?).

At some (probably low 1 or 2) level of "Meta", pure Lsrch would be ~~[illegible]~~ close enough ~~to~~ optimum, so that one wouldn't worry!

To repeat/review: If I find that a particular "Lsrch"/srch does modify its P.D. during the srch, Then do a hyperorder search over "objects" like the descn of "Lsrchs".

My Mind is not (entirely) clear on this Matter! Hvr, it is a very impt idea & I have to have it ~~[illegible]~~ Clear in my own mind [illegible] ① To use it ② to explain it to others

N.B. This idea ~~[illegible]~~ may also be useful in QuickAbort & all the ~~[illegible]~~/Search methods of 11.02, ~~.05~~, .18 !

It may well be that .05–.07 is adequate & that it includes ~~[illegible]~~ & ~~[illegible]~~ as prob-solving Methods — any Conceivable Heurs! (including "QuickAbort" or any of the 3 srch methods of 11.02, .05, .18)

The problem in .08–.09 is then relevant → Perhaps Go to Meta$^{(n)}$ levels until P doesn't change during Srch. I guess ~~[illegible]~~ a difty is, that a "Human" would remember info from ~~[illegible]~~ previous trials — always...

Suppose one has a set H_i of heurs = methods of solving a new problem. One could give each pc's via the G.P.D. & use Lsrch; But if it is useful to remember & use info from earlier H_i trials to modify forms (or pc's) of later trials, Then the set of H_i's is not "strictly Lsearchable". We can, hvr, devise a New

.35 H' that consists of trying the H_i in some Lsrch-like order but allowing remembrance & use of info from previous H_i ~~trials~~ Trials. It is not specified how H' does this. L srch need not be used at all.

Perhaps only consider algms in which the trials are ~~[illegible]~~ pc's do not vary during the srch. If we had a set of algms in which "remembering" & modifying pc. occurs during the srch, we would devise an algo like H' (.35).

.39 Hvr, Lsrching over Algms like H' (.35) would seem to be very wasteful — i.e. that info from earlier trials, would ~~[illegible]~~ certainly be quite useful!

.01 Perhaps, eventually, t. system would find a single hy level "heur" that would control

.00 Search [redacted] in any Lsrch way but would allow lrng. from previous sub-trials.

t.01 Hvr. maccurdal. 23.39: ~~Most~~ Much probsolving does allow use of info about "previous trials" — Yet "Pure" Lsrch does not

There is a small dependence on previous trials in PURE Lsrch: previous, unsuccessful trials are not repeated (trials w/o replacement)

On t. other hand, if TM does "Learn" during Lsrch (from previous trials), this should help get solns. Quicker! — Tho we would not be able to use $\frac{T_0}{pc_0}$ as expected cc of Soln.

Ass. TM should be able to sort of "Track" ordinary t. Human Lrng process: Lsrch is Meant to be a Emulation ("≥") of human lrng.

So! 2 ways of dealing w. T. Problem:

H' is designed to optimumly use info from previous trials in some "Good"

H' ← (23.35)

.15 1) (Vague): ~~Express~~ But see .19 Express t. search w. lrng from previous trials as a (Heur) (cand) itself.

One could do (Pure) Lsrch over several of these. (I'm not clear in my mind about this, hvr.) → see .19

2) Do Lsrch but change pc's during search: This could be done immediately — producing uncertain results (22.26 ff & note on ~~it~~ line 22.26).

Or, we could do pc modifn. ~~before~~ "Rounds" of T ← 2T (or T ← 3T or T ← 1.5T)

.19 Another view of .15: That H' would be discovered by normal ("Pure") Lsrch as "just another heur"! T. Heurs for discovery of H', hvr, mite involve ability to watch" TM do "Pure" Lsrch & note that previous trials' failure would be useful. .26

.21 [SN] ~~Cands~~ CANDS w. small T (due to time saving tricks, but not necly hyer pc) will be "Selected for" by Lsrch! {But not Quick Abort at first level!} To what extent does this deal w. t. diffys implementing [illegible] 11.05 & 11.18 ?

.26: .19 is more general than simply changing t. pc's during t. Lsrch: It tells just how the pc's (or choice or ordering of Cands depends on t. ~~[illegible]~~ (traces of) t. trials of previously tried CAnds. But see .32 for objection

Try to ~~[illegible]~~ write Summary of just what progress has been made on t. problems of 11.02 ff. Do I have a more or less adequate ~~[illegible]~~ (set of) soln(s)? Start by looking at Reviews of ZASONVERM 20.12 ff

.30 [SN] T. practice of ~~[illegible]~~ learning from previous trials in a run seems relevant to t. problem of "Flat P.D." ~~of~~ 17.12 ff & The ⟨T. 2^l v.s. T/pc or Kcost v.s. $-\log_2 pc$.⟩ ~~[illegible] for reference~~. ← Maybe not relevant

.32 Diffy w .19 This heur does suffer from t. "Flat P.D." diffy: T. successive trials for good (H')'s are all very similar: To not use info about this ~~correln~~ sequential correln. would be very wasteful!

.38 When I tried to Make P.D. change during t. Lsrch & used t. resultant $\frac{T}{pc}$ as cutoff Criterion: (How bad was it actually?) I think t. estimate of total needed srch

2.11.01 TM 1Bulg.

Super Spice (Simulator)
C:\Dayo MSM
MSM

.-.01 times of $\frac{T}{pc}$ ~~was~~ was way off — but was the ordering of trials very wrong? T. ordering might still have been ~~[illegible]~~ [the best possl.] — OR was it not "Best", but simply a "greedy" (choice-) optimum | > ("Greedy" in this case means "No Experiments — no trials to get early information"). [Another diffy w.r.t. $\frac{T}{pc}$ estimate was that since pc was a function of ~~former~~ traces of previous trials, one couldn't estimate it w/o doing a run first!]

(SN) In all of the large stuff, I've been thinking of the TEST model of Lsrch. By using Levin's || "timeshare" model, (or even "true ||" model) one may get a different perspective. !

$\sin x + x^2 = 1 \quad x = \sqrt{1 - \sin x}$

$x = \sin^{-1}(1 - x^2)$

.08 In .01: Greed is a serious business! Experiments can be vital to certain Heuristic processes: — Hvr, here the Greed is only at a "Global" level. Is it less impt??

The inability to get a g. estimate of $\frac{T}{pc}$ in .01 is bad, but not critical to "optimality" of Lsrch. — the greediness makes it less optimal.

One General Argt. for the ≈ optimality of Models of the GPD + ≈ Lsrch, is that ~~all~~ all Human heuristic search can be ≈ Modeled or Emulated in this way) Is this true? Isn't this what I'm trying to show? (?)

.17 So: The reconstructible of ≈ 24.38 is a subclass of [cases that involve any reordering of trials.] In this subclass the ordering of trials ~~[illegible]~~ is not determined at the start of the srch, but is calculated as the srch progresses — as a function of the traces of the cands thus far.

Re: "Greed" in .01 & .08: Is this Really a diffy?! TM is trying to adjust GPD so that any particular problem is solved as rapidly as possl. "Info gathering trials" can be regarded, not as "trials", but as activities designed to solve the problem — part of the ~~[illegible]~~ Cand. [GPD]

cands

.23

(.24) A fairly General form for the ~~[illegible]~~ G.P.D.: Given ~~the~~ the present problem, ~~[illegible]~~ and the traces of work on it: what is the best next ~~move~~ (or moves)? This seems to be moving away from Lsrch, hvr. ~~This~~ As stated, here, it looks like a very diffl problem, but Humans use heurs to get reasonable solns. The [AND-OR] net "soln" seems relevant in .24

(.24) could be a Significantly New Track: See how far I can get w. Lsrch for now! Also, can I make the Lsrch system ∋ it can arbitrarily configure itself? Is (.24) the most General possl. Configuration?

.24 looks pretty Much like the "Cure cancer" problem (19.26 is a soln.) ~~a~~ — Note, hvr, its not a complete soln. see 19.40

SN: perhaps ~~a~~ INV probs can be usefully regarded as "OZ probs": we want a soln. in minimal time. ("Soln." means satisfying the constraints). Hvr, its not like OZ probs in which one plugs a cand in & the Score comes out!

.39 (N.B.) The "Inversion" OZ problem of Score = $F(\overset{\text{most}}{X}, \overset{\text{true target}}{T})$ is an very important kind of OZ problem) to solve "Any time" problems ~~are~~ are ways of approximations to this.

.5
.58
.67
.616
.6156
.65
.628
.642
.675
.638
.675
.63757
.6362
.6428
.6378
.63693724
.636770~~356~~
396
2846
~~3~~1421
30901
33795
31953
33091
32731
2665
97
92
.75
.914
.2996
1.1655

—.01 Perhaps Marcus' remark that "finding a ~~method~~ factor of doing Matrix Mult[illegible], was not an o.k. problem" was an example of 25.39!! So: How do we express this as a "regular" (Lsrch Solvable) problem? Well, for some algms, we can often find upper bound for soln. time, as an analytic expression. However, comparing these expressions are not really linearly ordered. In the case of Matrix Mult, w. adj n, we do have the sort of bounds $\approx n^b$ for soln. times. They are linearly ordered by b.

So Marcus' remark was not as pointless as I ~~[illegible]~~ thot!

Actually, we might use Lsrch in a direct way: Say one did an Lsrch for [redacted] 4x4 matrix mult of a random real example. The check for the result, could be ~~[illegible]~~ very fast, since for each possible input problem, the soln. is known — If takes $T \approx 4^3$, ~~say~~ to do this, but the computation is done only once for the entire Lsrch. If we did a different ~~[illegible]~~ set of random no. inputs for each trial, then the cc would be very large [redacted] The even so, w. 2×2 or 3×3 matrices, it might be OK. — as the eq. for 3×3 could (~~[illegible]~~ we hope) be readily genzd to n×n.

(SN) Manuel Blum: Has done much work on cc of "checking" if $y=f(x)$. Says it's often much easier than computing $f(x)$. I have a bookmark in "Opera" on this "result-checking: Wasserman". Extreme example: $f(x)$ = factors of x. Is this a trapdoor function? or: find $x \ni M(x)=a$; a, M are known. This can be a hard NSV problem but easy to check. ~~Well, maybe a bit backwards!~~ ~~Perhaps the problem was~~

● ~~Given $f(\cdot)$ and y, to find $x \ni f(x)=y$.~~

No: it seems more complicated! I'll have to read this more carefully

1698) f2) 55yo.
1642 – 1727 85yo
1716
1642
74yo

.23 ~~[illegible]~~ 5.(01) — .40 is a kind of review of what needs to be done (& what has been done) in TM.

7.02 — .24 is another ≈ review of the essential diffys in the TM system.

11.02, .05, .018: The 3 kinds of Heurs seems very impt.

11.02 is further divided into Heurs w. constant P during srch & those in which P varies during srch, due to observations of freqs of earlier cands.

.27

.28 For 11.05 (Same ordering of trials, but changes in cc of trials: ● example of the cpu speed & use of 11 machines ~~[illegible]~~ — These are not very interesting.

Quick abort is ~~[illegible]~~ interesting, & I'd like to ~~[illegible]~~ some examples & (if possl.) ways to Genz. it.

11.18 is ~~[illegible]~~ Heurs that are mixes of 11.02 & 11.05 — Using 11 cpu's is, perhaps, an example but I know of no other examples. ~~Is~~ Does Lsrch normally use Quick abort? It can use $\frac{cc}{pc} \gtrless T$ as a cutoff criterion, rather than do all trials "to the end" or for a fixed $cc \le c_0$ (w. occasional $c_0 \leftarrow 2c_0$)

What I can do next is look at a bunch of probs & associated heurs; Try to find out how common the 3 classes of 11.02, .05, .18 are & the 2 subclasses of 11.02.

Before this, however, write a review of the "flow" of TM: How it is supposed to work, & what parts don't work so well, & "partial fixes" that may be fairly good. $\boxed{7.02 - .24}$ was a start on this. Actually print this up: initially in Latex, if necy — but there will be ~~[illegible]~~ few complicated eqns.

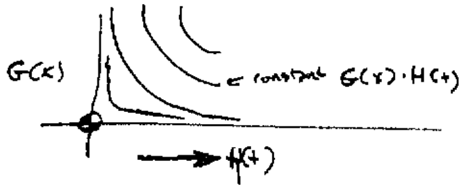
(.01) (SN) On QuickSort; If we used true search time of a Gorc for the GPD, then QA Alg would be recognized as good. Hvr, using ~~[illegible]~~ · ≈ $\frac{T_i}{P_{ci}}$ other corpus — which is a much easier (class cc) to compute, — would not recognize QuickSort as useful.

.00

.01

(SN) In $F(X,T)$ "oz" probs! Try! Do all $x \ni F(x,t) < F_0$ (F_0 is small) then do $F(x,t) < F_0 2^n$. for successively higher ~~[illegible]~~ integer Values of n.

Say $F(x,t) = G(x) \cdot H(t)$.

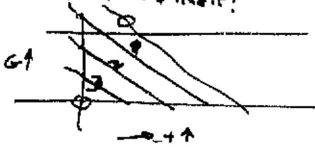


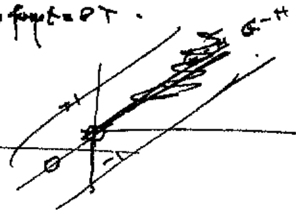
No!

$\bar{G} \equiv \ln G$ $\quad \bar{H} = \ln H$ $\quad \bar{F} = \ln F$

$\bar{F} = \bar{G} + \bar{H}$. T. Trouble is; H is a ↓ ~~fn~~ of PT.

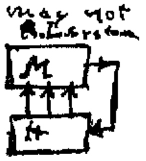
So $\bar{F} = \bar{G} - \bar{H}$ is more like it!





(.10) (SN) When I was considering t. possy that "ALife" systems would eventually "get out of t. machine" & into RW.! This was via t. interaction of t. ALife system with the user. Here t. User was a "SuperUser" who was developing, modifying, t. AL system.

This amounted to a RTM system in which t. User was t. RTM that was being "run" by t. AL system. The Q is: Is this last mentioned system a "full" RTM w. large horizon, or does it have a horizon of 1 only? — If this last is true, t. system may not be dangerous. The M's output ~~[illegible]~~ becomes oriented toward getting the Human to make mods. in M, that will achieve t. Goals of M ··· whatever they may be —.



.22 If M is "open" to H, then M has more control over H. M should try to get H to "look inside" it (≡ "openness") → 56.81

*If we think of H as a Human in [illegible] ~~[illegible]~~ "open" system would be a symmetrical & it would not be clear as to "who was manipulating who".

SN In Koza's GA that uses "Spice" to implement a Gorc for electronic ckts; Spice is t. bottleneck in t. System: ~~[illegible]~~ took w .5 seconds for an evaluation! We need to simulate Spice approx'ly! This can be done by "factoring" t. ckt. into parts that have known, simpler responses. ~~[illegible]~~ A transistor ckt would act like –(∿)–www– for a large part of its I/O [redacted] domain/range, a diode would act like a ideal diode plus a resistor, etc. Instead of using incremental time (as in Spice), Complex freq domain would be used for analysis of frequency behavior. Essentially, the ckt would be dealt as a (Engr.) (Human) thinks about it. (Remember: in Lenat's work, he found that the "slots" that were designed to facilitate Human understanding of t. system, were of much use by t. system for devising & applying heurs

} My imperfect "memory" of Lenat.

2-14-01 TM REV. This Paper — i.e. first few lines of 29 — deals Inv & (to greater extent) OZ problems.

The TM ~~[illegible]~~ system: TM is a general problem solver. Its input is the description of a problem, in some formal language: Its output will (~~[illegible]~~ ~~[illegible]~~ ' after a time, either a solution, or ~~[illegible]~~ or a statement that no solution exists, or sometimes, no output at all.

.03 The kinds of problems are ~~all~~ in 2 large categories: Inversion problems and Optimization problems.

Inversion problems are easiest to describe: [use text from Sol 86, 89 but give more examples. Trav. Sales Prob., all P & NP probs of computational complexity theory. These are all problems in which checking a candidate solution is of relatively small computation cost.

There is another kind of Inversion problem in which checking whether a candidate is a true solution, is very expensive. These problems are dealt with in a ~~much but~~ very different way from "ordinary" Inversion problems.

Also Note: Manuel Blum has done work ~~[illegible]~~ on "checking" See ~~[illegible]~~ Website.

(Example: Nonlinear eq. solution, error ε .001, say.

Example: find ~~value~~ x to max of $\sin x \cdot (1-x)^2$.

$x^2 + \sin x = 1 \pm .0001.$

Happen Deals w. expensive tests in Inv probs, is not clear. If test results just Yes/No, then the list of failures has little info! (Poker? Checkers/chess seems to be ~~an~~ exceptions. W. expensive tests in OZ probs, TM can trade(?) faster w. less accuracy & less cost. Maybe Emphasis, w. only Expensive y/N & 6. are not common. → p47.01

.14 Optimization problems come in many forms. ~~The best, simplest form is most Common~~ ~~We would~~ we have a function M(x) that maps strings into reals nos. We want to find a value of x such that M(x) is maximum.

Various constraints on the form of M and limitations on the computation cost of the solution, give rise to many ~~[illegible]~~ problem forms; which have to be solved in essentially different ways.

The simplest form is one in which the function ~~[illegible]~~ M(·) is known and can be very easily (quickly) calculated. The time constraint is that we need the best x possible but we only have time T_0 to find this x.

Another (not so simple) form: The function M(·) is known, but it takes an enormous time T_1 to calculate: $\frac{T_1}{T_0}$ is not particularly small: say it's 20.

A closely related problem type: M(·) is not known, but we can "ask a teacher" what M(x) is: we ~~[illegible]~~ are allowed to do this only a fixed number of times; (say 50) during the problem solution.

~~Other~~ Another form: ~~[illegible]~~ Computing M(·) takes little time but the values of M(·) we obtain have noise in them. ~~[illegible]~~ e.g. M(·) is a real world measurement with imperfect instruments.

Another form: ~~[illegible]~~ we can ~~compute M(·) we~~ obtain M(·) from a teacher, quickly, ~~but the value [illegible]~~ but M(·) varies with time and we do not a priori, know ~~the~~ ~~[illegible]~~ the form of this variation.

Try to give ~~an~~ one or more example of each type of problem.

Anytime problems are a variation on .14, and on several of the others.

.38 The Maximization of M(x)·f(t) is a very difficult, though not unusual kind of optimization problem. Here M(x) is known and is quickly computable. t is the time at which the system presents its solution: F(·) is a known, monotonically decreasing function.

2.14.01 TM Rev.

—.01 : Show how induction problems are OZ problems (find a set of strings x_i ∋

.06 Prob($M(x_i)$) = a ∴ ≤ $2^{-|x_i|}$ = max in some fixed time, To or as an "Any time problem"

.08 In both Inv & OZ problems, the object sought is a string, or number or a program for finding

.09 a string or number (: Both are examples of "Programs")

Say M(x) = a is an Inv. problem. We are given that M(·) & a. (string/no.)

A (direct) soln. ~~for~~ given ad x value: A pgm p for x wrt machine M_r ~~[illegible]~~ has property

$M_U(p) = x$, $M(x) = a$: So $M(M_r(p)) = a$: which is a kind of INV problem.

M_r is some standard ~~[illegible]~~ Machine. M_r is usually universal or almost universal.

Perhaps this is the idea! We want ~~[illegible]~~ to try cands in ∝ PC order, so if M_r is a UMC, then putting codes into M_r in order of length will give outputs in ∝ pc order.

So, given the GPD, P(M(·), a) its output d.f. is on ~~[illegible]~~ strings that ~~[illegible]~~ most probably have the shortest soln. time for the problem (M(·), a). This means that the outputs are Mut. exclusive — (But it may well be that almost all "soln. times" (of M(x)) are about the same: Hvr. Generating x can take various times So is GPD reasonable ? !

P(M(·), a) is the probly that a particular cand x is ① a soln to M(x) = a ② that of the legal solns, the process of Generating x and testing it via M(x) = a is min among all solns. ((Not if we use defn. 31.21 for a "CAND"))

.20 ~~[illegible]~~ No: The soln to M(x) = a often is unique. So perhaps the PD is not on x,

.21 but the probly that the program will generate a soln (any soln gen't) & will take min time to do so. (including testing.)

.22 [The ~~[illegible]~~ GPD described above is derivable from another PD for which is the probly that a given pgm will

.23 generate an x that ① solves M(x) = a & ② take time T to ~~generate~~ & test. This Aux PD is probably easier to update — Tho the PD that's used in Lsrch is the one that we have a "Goodness" criteria for: see 18.07 forward & backward Refs.

I hope to be quite clear on this pt. before proceeding much further. Study lots of Examples

Is G.P.D. a P.D. on cands of x ? — the probly that $M_r(p)$ will solve M(x) = a in min time? → It would seem not ∴ since short cands ~~[illegible]~~ (— not "cands w. short cands") are given by pc's. (See (31.21, .30) for soln)

Let's go back to the Simplest case of Lsrch: given M(·) & a, to find x ∋ M(x) = a. ~~[illegible]~~ Then M_r(R) gives a d.f. on cands for x (R is ~~[illegible]~~ an infinitely long random string). Only when M_r stops ~~[illegible]~~ begin output: M_r is a UID machine. So is the P.D. the probly that the cand is a soln? If the soln is unique, the ALP induced by M_r(·) is a legit. P.D. & could be it! If there are several solns or no soln, its not clear that ALP(M_r(·)) is an appropriate P.D. for the soln!

In an earlier (Note (in 1Bk)) 9.15 I felt that it was clearly the "pgm" to find x, that the GPD was about! This seems reasonable! When I made a stack lang to generate Cands for the GA problem, pc's were associated with pgm in this way. The pc of the soln was then (ac) the total pc's of all the pgms that generated it (This is an unnormz pc.)

2·14·02 TM 1Bolz
6

—.01 In t. proof of Adequacy of Lsrch: we consider some (op.timum) inverse of f(·) — say $h(f(x)) = x$. — If $P(h(\cdot))$ is assigned to h, then it will take ~~time~~ $\left(\frac{cc_h}{P_c(h)}\right)$ ← exactly / to find h using ∥ Lsrch. ↑ or "Timeshare"

So maybe 29.20–.21 is correct (≡ o.k.). t. proby that t. program will generate a soln, & do so in less time than any other soln. If a pgm is "loop free", then time for execution is related to (∝ to) pgm length (= no. of insts). If there's 1 or more loops, this is not at all true — total cc then usually depends ~~[illegible]~~ mainly on how long each loop was run (how many recursions),

t. cond, $\underline{X} \equiv$ ~~[illegible]~~ $(M_r(\alpha, a, M(\cdot)))$ $\underline{\underline{\alpha}}$ is t. desc of t. pgm.

.10 We want ~~[illegible]~~ → $\underline{X} \ni M(\underline{X}) = a$: $\underbrace{M(M_r(\alpha, a, M(\cdot)))}$ = a.

$\boxed{\boxed{\text{Quick Abort}}}$ can be realized on any prefix of t. trace of ↗

How pc's are related to .10: If there is only 1 soln, X_0, then t. pc. assoc. w. that $\underline{X_0}$ is $\geq 2^{-\ell(\alpha_i)}$ for all α_i satisfying (.10 R): ~~If~~

If there is > 1 ~~[illegible]~~ $\underline{X}$ values, & X_j is one of them, then t. pc assoc. w. ~~[illegible]~~ X_j is $\geq 2^{-\ell(\alpha_{ij})}$; α_{ij} is t. jth string $\ni$ $M_r(\alpha_{ij}, a, M(\cdot)) = X_j$

In general, t. proby assoc. w. t. pgm α, will $\boxed{\text{not}}$ be $2^{-\ell(\alpha)}$:

$\underline{\alpha}$ will be generated by a stoch lang. If t. lang. is unambiguous, t. pc of α will be obtained directly from t. Grammar (if α is parsable ()); Otherwise we obtain t. pc of α by summing over ~~[illegible]~~ t. pc's of its derivations in t. Ambiguous ~~[illegible]~~ stoch Grammar.

If we don't use a stoch Grammar & use binary trials in .10, ~~[illegible]~~ t. cjs $\frac{T}{2^{-\ell(\alpha_0)}}$ (where α_0 is t. pgm. that has smallest L cost $\frac{T}{2^{-\ell(\alpha_0)}}$) will be much much ~~smaller than~~ (larger than t. true time for t. srch. This is because $2^{-\ell(\alpha_0)}$ should be "normed".

We use kraft ~~(in)~~equality to get t. $\frac{T}{2^{-\ell(\alpha_0)}}$ upper bnd. If we were able to use t. kraft equality (Normz values) this ↗ would be exact! (not an upper bn).

~~But we can only craft norms it [illegible] in usual L srch, because for the unsuccessful trials, $\frac{T}{2^{-\ell(\alpha)}}$ is usually~~ So you use ∥ t.s. search, w. very small ΔT before jumping to ~~the next~~ T.S. search ≈ ∥ srch. If no conds had failed at t. time t. soln. was discovered, then ~~[illegible]~~ srch time would be $\frac{T}{2^{-\ell(\alpha_0)}}$ exactly, ~~Since~~ since we would have kraft equality. If any conds failed before t. soln. was dscvd., then total srch time $< \frac{T}{2^{-\ell(\alpha_0)}}$.

Same argt. is true if we assign pc. by methods other than $2^{-\ell(\alpha)}$ (e.g. stoch Grammars)

So we get ~~[illegible]~~ srch time $< \frac{T}{pc(x_0)}$.

omit (Confusion betw Gpc (α | M(M_r(α, M(·), a) = a) and ⋯
(Confusion betw. $2^{-\ell(\alpha)}$ & Gpc(α) : In both cases, α | M(M_r(α, M(·))

.38 Gpc(α) is a func of M(·), & is t. basis of t. "Grand" system.

.34 If α was obtained from Gpc & assigned a pc, then $2^{-\ell(\alpha)}$ is not of interest for Lsrch.

2.15.01 TM : 1 Bdg.

.01 : If 30.38 were true, then its hard to see that $\ell(\alpha)$ would be of any interest at all ! Usen Virus

.0 If α were generated in a roto "random" way, it could be assigned a pc of ~ $2^{-\ell(\alpha)}$.

In 29.08 ff I wasn't clear in my mind as to just what the "Lsrch" in TM was looking for! Also, when I used the problistic form of Lsrch (using e.g., GPD (Grand Proby Distribn) — it wasn't clear what GPD was the proby of — whether GPD(M(·), a) was the P.D. of the soln to M(x)=a or a PD on the pgm that would generate a.

In 29.20 ff the (decision/cdm)eqns. ~~GPD was a p.d.~~ GPD(M(·),a) was a pd on pgms to create a from M(·) & a. [in general, however, the args M(·) & a are not always needed; ~~because~~ for the pgm, since they are args for GPD — which selects pgms in view of both M(·) & a. If TM remembers soln to M(·)=a, (i.e. x_0) it will output the simple function with outputs x_0 for all inputs.

If we use GPD for Lsrch, its output is a pd on strings/pgms; α. If (as is usually the case) GPD is a stoch Grammar: ~~[illegible]~~ we will use various tricks to get the PD to give us α values in ≈ pc order (hy pc first). I have solved this problem many times: The latest soln I remember as being particularly good! — I think I

.18 used Huffman Coding. It was in Bdg. (2000): look in indexes to find it — I think in Early Bdg. I know it works for a Bernoulli process — I don't know if it can be made to work for a CFG or CSG. I think I wrote some analysis of its approxn for CFG. — The algm may have involved trying to be non-greedy to various degrees. ~~[illegible]~~ .16

.21 Hvr, more generally (36.37) the output of GPD for a INV. problem can be any pgm that could solve the problem: (This is analogous to the OT's that GPD outputs for solns to OZ probs) It can be an elaborate general technique for problem solving — like converting the Inv problem to a ~~[illegible]~~ optimization problem or GPS (which may be a vector Gore for Hill climbing), or α,β search or whatever. It can be a search technique — like, or

.26 The idea is that any problem solving technique is a legal output for the GPD. a improvement on Lsrch. See 40.01–.16 !

Note that .21 is combo of 30.10: If α is the string that GPD is the pd of, then we want α ∋ $M(M_r(\alpha, M(\cdot), a)) = a$: α tells how to solve the problem M(·), a.

.21, .26 may cause trouble in my idea of heurs almost always being realizable by a modifn of

.30 the GPD. A Heur may want to modify searches (or any other aspect) within α, the cond. ~~[illegible]~~ problem soln. — Well, theoretically, this would be covered by Modifn of the ~~[illegible]~~ GPD ! To see this : Say the heur modifys α → α'. This means that α' now has the pc of α & α has a lower pc — α' must have had a very low pc before the applicn of the heur prop — so low that we didn't even consider it in the GPD.

.36 So ~~[illegible]~~ .21 ff & .26 ff & .30 constitute a Change Note ! — Suggesting that almost all heuristics can be expressed as modifns of the GPD — even if the output of the GPD is a very general kind of problem soln.

Spec 30.-01

Horizon: A not bad approach! (But See 37.22 for Deep Criticism!)

.01 Clearly the steady state reinforcement system is not the same as finite horizon, h w. h→∞.

Steps toward a steady state soln for the recurrent problem!

Suppose we have a corp in which we have a stockholder's report every T_0. So we start ~~at~~ at ~~beginning~~ w. horizon = T_0, work until T_0; loop back. .03

w. Horizon = T_0, we find it best to work on "Self improvement" up to ~~#~~ time kT_0, then work on Reward producing projects (direct problems) for $(1-k)T_0$.

To modify this for "steady state w. horizon T_0", Time share work w. fraction k on Self improvement; fraction 1-k on direct problems.

This category relates to the "50% 'solution'" problem: Instead of 50%, we use k,

Another way to look at .03 ff: ~~Although~~ We have to chose a fraction, k: Chose k ∋ The expected total yield at time T_0 is max. We could vary ~~slowly~~ Slowly vary k as our state of knowledge & quality of problems change.

Also, it may be that ~~we~~ may want to vary T_0, depending on Political (external) conditions

But ~~#~~ a main idea, is that ~~[illegible]~~ using ~~k=.5~~ .5 can't be off by more than a factor of 2 ~~#~~! i.e. "olde 50% soln." idea.

So, as before, my ~~[illegible]~~ conclusion is that this problem is not critical, & at the present time, I shouldn't be spending ~~a~~ time on it, unless I have Some Great New Ideas!

For finite horizon: (This is a common type of problem: "Any time" is also common)

Say s.i. work produces/ in Δ time. 10% improvement in direct prob. solving.

So to work ~~#~~ time T at s.i. gives $\approx e^{\frac{T}{\Delta}\times .1}$ increase in Q.

.29 ~~So we want~~ w. horizon T_0, we want $T = kT_0$ ∋ $\boxed{(1-k)T_0 \cdot e^{\frac{kT_0}{10\Delta}} = \max}$

.30 $(1-k)T_0$ is ∝ amount of R one gets in time $(1-h)T_0$: say $q = 1-k$; $L = qT_0 \underbrace{e^{\frac{T_0}{10\Delta}}}_{\text{ind of } q} e^{-\frac{T_0}{10\Delta}\cdot q}$

So max $q e^{-\frac{T_0}{10\Delta} q} = q e^{-\alpha q}$ $\alpha q \equiv p$ max $p e^{-p}$ $\alpha q = 1$; $q = \frac{1}{p}$

$\ln p - p$ $\frac{1}{p} - 1 = 0$ $p = 1$ So for best yield, $\frac{T_0}{10\Delta} \cdot q = 1$ $q = \frac{10\Delta}{T_0} =$ ~~k~~

$k = 1 - \frac{10\Delta}{T_0}$

.38 More generally, if Working on S.I. for time T produces/ a factor of $e^{\alpha T}$ in m yield,

$\boxed{K = 1 - \frac{1}{\alpha T_0}}$

$\boxed{1-k = \frac{1}{\alpha T_0}}$

k = fraction of time worked on S.I.

Wooops! $0 \le k \le 1$: $\frac{1}{\alpha T_0}$ can't be arbitrarily large.

33.01

2.16.01 TM : 1 Buy. Horizon

.01 : 32.40 Based on "50% soln" problem: consider ($\boxed{32.29}$) : $(1-k)T_0 \cdot e^{kT_0\alpha}$

What's t. difference (ratio) betw. using $k = 1-\frac{1}{\alpha T_0}$ & $k=.5$? $\left(1-k = \frac{1}{\alpha T_0}\right)$: $(1-k)T_0 = \frac{1}{\alpha}$

$$\frac{1}{\alpha} \cdot e^{\left(1-\frac{1}{\alpha T_0}\right)T_0\alpha} = \frac{1}{\alpha} e^{T_0\alpha} \cdot e^{-1} = \frac{1}{\alpha e} \cdot e^{T_0\alpha}$$

For $k=\frac{1}{2}$ we get $\frac{1}{2}T_0 \cdot e^{\frac{T_0\alpha}{2}}$.

ratio of optimum yield to k=.5 yield = $\frac{1}{\alpha e} \cdot e^{T_0\alpha} \cdot \frac{2}{T_0} \cdot e^{-\frac{T_0\alpha}{2}} = \frac{2}{e\alpha T_0} e^{\frac{T_0\alpha}{2}}$

so if $x \equiv T_0\alpha$; it's $\frac{2}{e} \cdot \frac{1}{x} \cdot e^{+\frac{x}{2}}$: for $x \ll 1$, this is $\frac{2}{e} \cdot \frac{1}{x}$

or if $x \gg$, $\frac{2}{e}\frac{1}{x}e^{\frac{x}{2}}$ can be enormous! May be error.

Woops! x could be < 1 ~~[illegible]~~ error! So our optimization

is only for $x \geq 1$; if $x \leq 1$ then the max of 32.29R is at $k=0$

This means (usually) that we should ↑ T_0 (= t. horizon).

Hvr, for large $x = \alpha T_0$, t. ratio is very large;

The 50% soln. means that if one had twice as much time $\left|^{T_0}\right.$ at $k=.5$ $\Big|$ i.e. $T_0 \to 2T_0$, one would do

at least as well as normal T_0 w. optimum k.

.18 (While our yield can ↑ enormously w. use of optimum k (v.s. k=.5) we can get

.19 at least as good yield by doubling cpu speed. & using k=.5. ⟶ $\boxed{34.06}$

Also note, even if α is small, one can always get x to be large by using a large enuf T_0 (!)

.21 Well, we can fix! The first long period of TM's life is spent in s.i. (not working on

serious problems). — Hvr, in this phase of TM's "life" could we not have a $\boxed{\text{Meta s.i.?}}$

— Maybe not, because $TM_1 = TM_2$ already!.

Hvr. In the initial phase of TM's trng: All probs are "study problems." My original idea was to get

how TM work on s.i. until it could do so w. some efficiency. At any pt. in TM's life, it has a certain

value of α ($\ni e^{\alpha T_0}$ is s.i. factor after working for time T_0). $\frac{1}{\alpha}$ is in t. nature of a "Time".

Say $\frac{1}{\alpha} = \tau$: $\boxed{\text{So we need a "Horizon"} > \tau}$ (so $\alpha T_0 = \frac{T_0}{\tau} > 1$) before it's useful to

work on s.i.

---

on t. $\boxed{\text{meta s.i.}}$ idea! Say t. long goal is revision of QM. : we start w. s.i. only:

Learn math, some conventional physics: Hvr, in regular math & physics, there are meta tasks:

How to do searches efficiently: Various math hacks. We can do these to any depth, but t. deeper

we go, t. fewer problems we can usefully solve (I conjecture)

Secondary meta s.i. is equivt. to ↑ α of primary s.i. so $e^{\frac{T_0}{\tau_0} \cdot e^{\frac{T_1}{\tau_1}}}$ here T_1 is time spent on secondorder s.i. & τ_1 its horizon.

Probably $\tau_1 \gg \tau_0$, so normally, it wouldn't pay off to do 2ndorder s.i. — because payoff horizon would be very large. $\left(= \exp\left(\frac{T_0}{\tau_0} \cdot \exp\left(\frac{T_1}{\tau_1}\right)\right)\right.$

2.17.01 TM : 1 Bug Horizon Rev .25

—.01 : 33.40 : It would seem that since there is only one GPD for all orders of s.i. problems, that there would be no such orders! — But it may be that the unity of the GPD is not so complete! When we "Improve the P.D." one decides on which parts to improve (See TM26 ; ~ 18.—01, .07, 19.05, 20.—01,) 20.09 : ~~[illegible]~~ wts. for TM2 (& s.i.) have to be supplied by user.

.06 : 33.19 suggests that the "50% soln" may not be so good! — Long Term it can't be a great deal! In fact, In initial TM exp. k=1 (all time is spent on S.I.) (See 33.21 ff on this)

Perhaps in General I will simply design a TSQ. to incorporate all the S.I. & higher level S.I. Certainly I did at the outset. I don't yet see if "steady-state" model occurring would fit so well into the TSQ. It's definitely a "more distant goal".

Consideration of Necy Short term Goals for a RW animal: It has to stay alive! Needs rapid response to threats — "accidents"; slower response to get food: Much more rapid response to get Air. So Order of needs: Air; food & water; further reproduction.

Also, staving off predators; Hot, Cold, fire Accidents (falls) ← Near top of list for rapidity of response.

The Goals of TM will be quite diffrnt. I want to be able to control food, air, predators. I don't want it to reproduce (or if I do I want to control it).

The top goal is to solve very difficult problems that I give it. — lesser Goals: to solve easier problems that I give it.

In animals, ~~[illegible]~~ the "fast response" goals are vital, & give quick feedback, so they are learned quickly. This knowledge is then a good basis for further learning of a more complex kind.

In order to do S.I. & Meta$^{(n)}$ S.I. TM has to have adequate SSZ. For hyer order meta S.I., the SSZ has to be very large. So perhaps one should not attempt hyer level S.I. until the SSZ is adequate.
.24

.25 So: Review: Main pts. thus far: Study (Also 37.10 See 37.22 for Savings Criterion!)

1) ~~[illegible]~~ for one level of S.I. optimum fraction of time to spend on direct (not S.I.) probs is $\frac{T}{T_0}$! (? level ?)

$T = \frac{1}{\alpha}$ = overall time spent on S.I. in order to mult effcy by e ; ≅ ← Self improvement

T_0 is "Horizon" = Amt. of time for entire work. $\frac{T_0 - T}{T}$

$1 - \frac{T}{T_0}$ is fraction spent on S.I.

Total time spent on direct problems is T ($\leq T_0 \cdot \frac{T}{T_0}$) $\frac{1}{x} e \cdot e^{x} = 1$ when $x = 1$

" " " " S.I. = $T_0 - T$.

If $T \geq T_0$ we spend no time on S.I.

~~[illegible]~~ (Utility of working on direct probs only (No S.I., x=1)) = T_0.

Utility of optimum ratio = $T \cdot e^{\frac{T_0}{T} - 1} = \frac{T}{e} \cdot e^{\frac{T_0}{T}}$

ratio ~~[illegible]~~ of utility of optimum allocation to zero allocation to S.I. = $\frac{T}{T_0} \cdot \frac{1}{e} \cdot e^{\frac{T_0}{T}}$ $\boxed{= 1 \text{ for } \frac{T_0}{T} = 1}$

Utility of $\frac{1}{2}$ time S.I. = $\frac{T_0}{2} \cdot e^{\frac{T_0}{T \cdot 2}}$

ratio of " " " " to zero allocation S.I. = $\frac{1}{2} e^{\frac{1}{2}\frac{T_0}{T}}$.

" " utility of optimum to ½ allocation = $\boxed{\frac{2}{e} e^{\frac{1}{2}\frac{T_0}{T}}}$.

(35.—01)

2.17.01 : TM : 1Bulg $\boxed{\text{Horizon: Rav}}$

7.01 : 34.40 : $\boxed{\text{S.d. is useful} \Leftrightarrow T < T_0 .}$

Note S.d. can ↓ T, so it can be very worth well, but it is a "meta hour" & not used very often.

SSZ for higher level hours (≡ S.d.) ↓ rapidly w. <u>order of</u> <u>heuristic</u>. Also the

.ok T values for these ~~for~~ higher level hours are <u>very</u> <u>large</u>.

2.12.01 TM: IBulg.

Horizon

20↑
24yo
3yrs/yr.
2k/yr.

-.01 : More on "Horizon": Say we modified the problem in an attempt to get a "steady state" soln. Then if we ~~[illegible]~~ "improvement" work on S.I. a fraction k of the time, then as we accumulate S.I. work, "direct work" efficiency will continue to ~~go~~. ↑ R = rate of work production.

$R \to (1-k)R\,e^{z}$ $z = ▨ \cdot \frac{1}{T}\int_0^t k = \frac{tk}{T}$: $R \to (1-k)R\,e^{\frac{kt}{T}}$.

.03 And total $R = \int_0^{\infty}(1-k)R\,e^{\frac{kt}{T}} = (1-k)R\,\frac{T}{k}\,e^{\frac{k}{T}t} = $ ~~[illegible]~~ $TR\cdot\left(\frac{1}{k}-1\right)e^{+\frac{k}{T}\cdot t}$

This is certainly not "steady state". For max growth rate, k is close to 1.

.07 This is equiv. to maximizing (as R) at $t \to \infty$. ⟶ 37.10

From [illegible], the goal of optimum steady state ~~rate~~ comp. is unclear! -- May be Meaningless!

Anyway, a perhaps useful goal for a reasonable T.M.: ~~[illegible]~~ A very large, diff't problem: Say "cure cancer". It could be defined as a OZ or INV problem:

OZ problem: As good a % cure within time T as one can get.

.15 "INV": Get 80% cures as soon as poss'l.

.15 seems to be signif'ly diff't from most INV probs! For one thing, the satisfaction criterion is expensive, so modeling part of the criterion would be part of the soln. technique.

Also, the usual Lsearch would be ridiculous! (It would Anyway seem that it would be nec'y to modify the GPD a lot so that the goal would have a reasonable p.c.) — To do this "Improvement of the GPD", TM would have to do "experiments" — which is definitely "Out of the spirit" of my usual concept of an "INV problem" — this seems to be a difference not covered by the "My Expense of the ~~test~~ test criterion".

Another Diff't INV problem: To prove/disprove Riemann's hyp. (or Fermat's Theorem or Goldbach Conj.) : Here, it would seem that we would want TM to learn much Math, first. This would seem to be an again "not in the spirit of usual INV probs".

Before TM could approach such a problem, it would have to be trained (by one or more mathematicians) ~~by [illegible]~~ One way would be by giving it successively more diff't probs in areas of Math that seem relevant.

.30 This last idea is conventional TSQ design: What I'm thinking about is where T.M. decides that it has to learn more Math & starts reading ~~books~~ Math books.

In .30, I wouldn't expect TM to be able to do this sort of thing w.o. adequate ~~[illegible]~~ TMG = TSQ.

Also Note for most INV probs, knowing convert them to OZ probs then use hillclimbing or GPS.

.37 Perhaps ~~@~~ INV probs should be solved "like" OZ probs, in the sense that GPD [redacted] is a P.D. over Methods to solve the INV problem. Solns to OZ probs are similarly given by GPD over a P.D. over OT's - which are methods to solve a OZ problem. Stronger than "perhaps"!

2.18.01 TM: 1Bug.

.01:38.40 : It mite be pssbl. to have TM to work on Math probs & any I do — e.g. looking at literature.
— But a "≥ valid" way would be to look in books, reference books, & Math literature

I'm not sure I understand the difference betw 36.37 for CMV, & looking for a "peg" to solve the problem: the "pgm to solve the problem" could be to convert it to an OZ problem & solve it in a special way — or as a GPS problem, which, perhaps, we can regard as a genera. of OZ probs, that has special ways of being solved: (= special part of GPS).

.04

So, I have to give examples of various kinds of problems & how they are solved — What heuristics they use & how these heuristics are implemented by the GPS.

(.10): 36:07: A more realistic relationship betw. R & "$\frac{dR}{dt}$" (time rate of growth of R.)

We are interested in the final score J of $\int_0^T R\,dt = \int_0^T\int_0^t \frac{dR}{dt}\,dt$.

In 36.03 $\frac{dR}{dt} = (1-h)R_0 \cdot e^{\frac{kt}{T}}$: A more reasonable $\frac{dR}{dt}$ would note that it would depend a lot on the total amount of data input this far — (partly from statistical/SSZ considerations.).

Should this be simply α t? (" By definition": "t" could be defined this way ☺.)

While $\frac{dR}{dt}$ would ↑ w. t (due to lots of SSZ), it would ↓ w. t because it took more time to analyse more past data ?.

Hrs, on the whole, I'm quite uncertain as to the (commonly usual general) form of how $\frac{dR}{dt}$ depends on ~~[illegible]~~ "t" or $\int\frac{dR}{dt}$ or $\int R\,dt$.

.22 This is also assoc. w. criticism of my assumption that TM would always work on S.I. first (before working on Main problem). In many, maybe most cases, TM would want some experience w. the problems before working on methods to improve those prob solving techniques!

The great exception to the above would be mainly what I was thinking about in the first place: e.g. Unification of QM & Gen Relativity: TM would have to know a lot of Physics, Math before the problem could be explained to (him it ?).

In "Curing Cancer", again, it probably would be a good idea to learn what was "known" in Biology, before starting work on the Main Problem —

So, in General, my idea of doing S.I. first may be O.K. sometimes, but often, not!

More on .22 ff: In a hard problem, say one wanted to familiarize one's self w. the general ~~[illegible]~~ area of the problem or w. the parts of the problem, before ~~[illegible]~~ working on S.I. — Since this prelim examination would have no chance of solving the problem, it ~~wou~~ sh ould perhaps be regarded as pure S.I. (?)

2.18.01 : TM : 1Sug.

—.01 : 37.40 : Hvr, usually, I regard S.i. as being "Improvement of t. GPD", so it would not include ~~of~~ "Prelimy skermishes w. t. Present problem". It would perhaps ~~be~~ include "Prelimy skermishes" in t. approach of 36.37–37.04! Part of a "Method/pgm" designed to solve t. problem."

[(Perhaps) SN: Improving t. GPD would seem to not include OSL. Perhaps to do OSL in a problem, One has to do a special search ~~of~~ over Past Corpus.]

.08 → Actually: "Improving t. GPD" has 2 aspects ① "Updating", which involves (varying amounts) t. part of t. GPD concerned w. "t. PROBLEM Just Solved". ② A "More general improvement of t. 'GPD'", Which involves more problems of t. past (& perhaps in certain TM models, ~~of~~ a expected future

.12— Problems).

2·19·01 TM: 1816.

e^x/x

ABCDE ABCDE
ABCDE
ABCDE

.01: Space 31.40 The idea of 31.21, that the cand α, is a very general kind of prob. solving algm, It can give rise to α's that are very hyly correlated in the likelyness that they will solve problems. This can cause much wastage of time in search (as was previously noted in the

.02 "~~first~~ GPD" problem (See 17.12-.32; 24.30, 20.15, 16.31 ?)

.03 Hm: Say the ~~[illegible]~~ α's in the GPD were correlated so that success or failure of one

.04 tells something about success or failure of several other cands. — I think this is normally the case. Consider a Generalization of the [G.N. Theorem I] for correlated cands!

But the pc's in the GPD are for Mut. exclusive ~~[illegible]~~ cases! — So they can't be correlated! ?

So first I have to clarify ~~as~~ as to what I mean by "correlated" for ~~a~~ a mut. exclusive d.f. Intuitively, I mean (.03-.04)^T Hwr, for Lsrch, only failure info of a cand. can be used to influence pc of other cands (?) Or info that a non-trial has run for time T & not yet b won or lost.

.13 A try for Meaning: Say we have a ^certain^ mut. exclusive pd. on cands, $p(x)$: Assoc. w. PCNCS an expected cc to soln (assigned overall possl. x — using the best possl. strategy ~~[illegible]~~. — ordering trials in $\frac{pc}{cc}$ order. Call this $p(\cdot)$.

.16 So perhaps we ~~[illegible]~~ first chose cand α, $\ni \frac{pc}{cc}$ is max: This gives us a new $P(\cdot)$ over the rest of the cands;

.17 from it we chose the $\alpha \ni \frac{pc}{cc}$ is max, etc. This is a ~~greedy~~ GREEDY method. } → but see 44.23!

Using .13 in Lsrch, probably wouldn't help! If we have a bunch of similar, hyly correlated cands, then none of them have finished in early rounds, & ~~[illegible]~~ in those early rounds that we use all that cc in testing them. This situation was analysed before. (see (.02)^T for refs.)

On second thot, say the bunch of correlated cands. ~~[illegible]~~; ultimately, fail ~~[illegible]~~; when one fails, the rest are given low pc's, & are not tested until much later (say after the true soln. has been found). This seems ~ to QuickAbort!

.25 One way to deal w. the correln problem: Make up ~~[illegible]~~ ~~[illegible]~~ equivalence classes of cands that are hyly correlated. Select 1 (best) cand from each class. Assign lower pc's to rest of ~~[illegible]~~ each class. This ↑ pc of "best" & would solve the problem of "correlated cands", but there is the possy that ~~[illegible]~~ the "Best" cand in ~~[illegible]~~ a class fails but one other in the class really solves the problem! While this can be made unlikely (by suitable class size), we still have the possy of another soln. occurring in a diffnt equiv. class — not nearly the "Best" in terms of total cc.

Actually this approach may be very ~ to (.16-.17)! ~~[illegible]~~ The difference is that in .16-.17, before we have any failures in an "equiv. class" we would have to test them all to the current cc limit. Here, ~~[illegible]~~ (.25) we only test one in the class: when it fails, the rest of the class get low p.c. — so this really saves a lot of cc,

.37 ~~[illegible]~~ ~.25 may be a way to analyse use of Σ cost v.s. use of p cost for Lsrch: It may turn out that use of single pgm & its assoc cc. is ok — that using pc of a "soln" is not relevant. The "soln" has many codes for it, but we are interested in individual codes & their assoc. cc's. The p cost of soln. is the sum of the pc's of all codes for the soln., but this pc cost of soln. has ~~no~~ no-specific associated cc! ⟶ 44.01

.01 Impt recent ideas

1) 31.21 : Definition of a (for INV probs Cond, α) as any prob. solving technique for the specific problem: It could be as narrow as a single number that is the soln, or as general as conversion to a OZ problem. It can be a srch method (≠ Lsrch): e.g. one that learns better cond. probs

SN could 31.21 be recursive? < Yes i.e. could it "call" other prob solving techniques for other sub-problems? — e.g. in the AND/OR net > Probably Yes also, it can call on the GRAND TM pgm to ~~Solve sub problems by Lsrch.~~ It can be Lsrch or a replacement for Lsrch: See 46.01-.16, 52.02 .13

2) 31.30 -.40 : (Strongly Suggests) that the definition of ① (31.21) is compatible w. my idea of a heuristic being realizable by modifn. of the GPD.

3) The Definition of the GPD (Grand Proby Distribn). for INV probs: 29.20-.23
It is the proby that α (the cond'l pgm) ⓐ will solve the problem ⓑ will do so in least cc of all conds.
→ This pd is derivable from another P.D. that does not involve Optimizn (29.22-.23) ← Impt so the GPD has 2 aspects.

4) Points 1, 2 & 3 are critical in the continuation of the "Review" of 28.01 - 29.09
Next, we just [illegible] Lsrch in its Various forms (Use of proby or use 2^-ℓ
— then use T < kT (w. k = 3 optimum) or time shared ... why T < kT is very good.
- Next discuss "Improving the GPD" (TM2G) (18.07 -.14 Impt + 19.05 + 20.01
- Next Show how Heurs are almost always expressible as "Improvements of the GPD".
This is the Main Big Problem!

5) Improving the GPD : This Also involves Q's about "the Horizon" [illegible]
Horizon discussed! 32.01 to 37.12. 34.25 35.04 is a review of the early part, then 36.01 - 38.12 is a v.g. objection to the simpler approach of 32.01 - 34.24. (The 32.01 - 34.24 has useful ideas, analysis — it is, unfortunately, not good enough for the General case.) Criticism of 50% soln (33.19, 34.06)

I have to go over this stuff more carefully: there are many (smaller) impt ideas in this sequence!

.−.01 : Main Problem: ~~[illegible]~~ Give (many) examples of problems; & how they are solved; & what part of E. Soln. is ~~[illegible]~~ a heur, & how it works & how it ~~[illegible]~~ works by changing the GPD.

A propos Example: find an x ∍ $x^2 = 10 \pm$ ~~[illegible]~~ with 10^{-6}

TM has Hardware to 10^{-8} accuracy. TM knows about continuity, that x^2 is an ↑ funct ~~[illegible]~~ ~~[illegible]~~ if ~~[illegible]~~ x > 0.

There are several ways to solve this problem:

.08 1) $x^2 = 10$: use "square root routine" "learned in school"

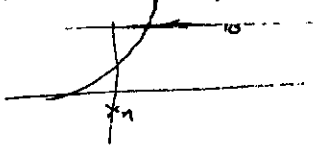
.09 2) ~~[illegible]~~ $x^2 = 10$: denote sqrt. root ~~[illegible]~~ routine; find x_0 ~~[illegible]~~ (largest integer ∍ $x_0^2 \le 10$.

Thus is 3; ~~[illegible]~~ $10 - 3^2 = 1$; $\frac{1}{2 \cdot 3} = \frac{1}{6}$ $(3+\frac{1}{6})^2 = 9 + 1 + (\frac{1}{6})^2$: Well, it is a rather routine!

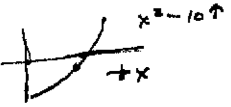
$\boxed{\text{try } x_{n+1} = \frac{10 - x_n^2}{2x_n} + x_n}$. $= \frac{10}{2x_n} - \frac{x_n}{2} + x_n = \frac{1}{2}(\frac{10}{x_n} + x_n)$: This is ~ to ~~[illegible]~~," but "usual method" ①

is arranged for ~~[illegible]~~ getting $\sqrt{x}$ for large integer x; also

.13 ~~3)~~ 3) Use continuity of x^2: ~~get $x_n^2 < 10$ & $x_{n+1}^2 > 10$~~



$x^2 - 10 = 0$



$x_{n+1} = a x_n + b x_{n-1}$

a & b are arranged so that a st. line ~~[illegible]~~ thru (x_n, x_n^2), (x_{n-1}, x_{n-1}^2) intersects φ at x_{n+1}.

.17 4) Use Newton's ~~[illegible]~~ (Raphson) Ralfson Method (requires diff. calc).

No! * 5) Use second order Newton method (this will not work because it requires ability to get $\sqrt{x}$ as well as diff calc.)

.22 6) (Based on "continuity" idea? Confine x to intervals $\frac{1}{2}$) (or $\frac{1}{10}$) as large as previous approxn. for "$\frac{1}{2}$" routine.

~~[illegible]~~ $3^2 < 10$ $4^2 > 10$

next $(3\frac{1}{2})^2 > 10$; so look at $(3\frac{1}{4})^2$ it is > 10 so $3\frac{1}{8}^2$ it is < 10 so $\frac{1}{2}(3\frac{1}{8} + 3\frac{1}{4}) = 3\frac{3}{16} > 3\frac{3}{16}^2 > 10$
hy low hy

~~[illegible]~~ This puts us factor of 2 in precision in each iteration;
we could do "factor to 10". Investigate Computational complexity.

.27 → * 7) $x_n = \frac{1}{2}(x_{n-1} + \frac{10}{x_{n-1}})$ T. heuristic basis is not clear — Well if x_n is too large, then $\frac{10}{x_n}$ is too small.
This is same as ② So take their mean; Is there a better way to "take their mean".

~~[illegible]~~ Most of these methods can, in turn, be obtained using diff'nt heuristic paths. (?)

for 1) There is not much "Heuristic". It is a memorized way to solve the sq rt. problem.
There might be some Heuristic interest here; To realize $x^2 = 10$ is a square root of a fraction problem. — There is the problem of indexing the memory — which has many things other than square root routines!

$x_n - 10 = \epsilon$

.35 for 2) Idea is "successive Approxn"! If x_n is an approxn, then x_n^2 is in error by ~~[illegible]~~ how much do we have to ↑ x_n so that x_n^2 gets ϵ larger? ~~[illegible]~~ $(x_n + \Delta)^2 = x_n^2 + 2\Delta x_n + \Delta^2$.
Say Δ^2 is small, so $2\Delta x_n \approx \epsilon$; $\Delta \approx \frac{\epsilon}{2x_n}$. So $x_{n+1} = x_n + \frac{10 - x_n^2}{2x_n} = x_n$

.36 This is in the direction of Newton's method, $x_n + \frac{10 - x_n^2}{2x_n} = \frac{x_n}{2} + \frac{10}{2x_n}$.
since $(x+\Delta)^2 \cong x^2 + 2\Delta x$ is like using a derivative. $= \frac{1}{2}(x_n + \frac{10}{x_n})$ — same as Method #7!

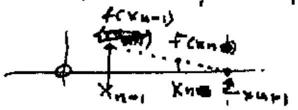
2.20.01 TM : 10.

Skinnerian Lrng. — Why inadequate .28

→ for training for good Creativity.

.01
.00

for 3)(41.13): $f(x) = x^2 - 10$:



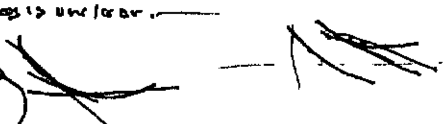
$(x_{n+1} - x_{n-1}) / f(x_{n-1}) = (f(x_n) - f(x_{n-1})) / (x_{n-1} - x_n)$

$(x_{n+1} - x_n) / f(x_n) = (x_n - x_{n-1}) / (f(x_{n-1}) - f(x_n))$: $\boxed{x_{n+1} = x_n + \frac{(x_n - x_{n-1}) f(x_n)}{f(x_{n-1}) - f(x_n)}}$

$x_{n+1} = x_n + \frac{(x_n - x_{n-1})\, x_n^2}{x_{n-1}^2 - x_n^2}$; to get this,

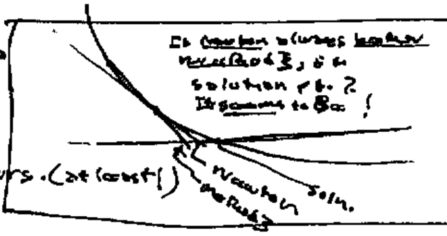
Keep track of x_n^2's, & $(x_{n+1} - x_n)$'s. How rapidly this converges is unclear. —

Is it better than $x_{n+1} = \frac{1}{2}(x_n + \frac{10}{x_n})$? ← Newton's method: which is faster



4) Newton's method: This turns out to be same as (2)(41.09, 41.35) because of 41.38: but heuristics are different

5) Double precision each round. I think this is slower than Newton; Newton doubles no. of decimal pts. every round. This gets some extra precision each round.



Is Newton always better method?, i.e. solution pt. ? It seems to be!

7) while this is same as 2) & 4), it is heuristically different

So 1, $\boxed{②, ④, ⑦}$, 3, 6   4 methods: the second has 3 different heurs. (at least!)

√Newton   or $\frac{1}{2}(x + \frac{10}{x})$

Re: 1) (41.08) So the student was given this Algorithm, with some instruction on when to use it. Say it's put into memory, but with not many heurs telling when to use it. The student then is given many problems in which $\sqrt{x}$ is used. In this case, probably $\sqrt{10}$ is a trivial problem — student may have computed $\sqrt{10}$ in the past & have it in RAM.

Actually, I did one (39. in which TM learned how to do linear eqns: then I gave TM a new primitive — $\sqrt{x}$, & it was able to learn to do Quadratic eqns. — then giving it "$\sqrt{\ }$" & $x^{\frac{1}{3}}$, it was able to do cubic eqns.

② saw (41.35) for heuristic Analysis; How it is ~ to Newton's method.

In ② consider environment in which this soln. could have been derived. First: the idea of "successive approxn" to solve problems

{Newton – Raphson}
Did Raphson know about derivatives?

.28 → $\boxed{SN}$ I was uncertain (within last yr.) about just why Skinnerian lrng was inadequate to learn creative discovery. I think the idea is that to solve probs of large CJS, TM needs heurs that are different from those needed for small CJS problems; — so if we do Skinnerian teaching — the student will be less likely to lrn (or discover) heurs needed for large CJS problems.

(NB) The $x^2 = 10$ problem can also be regarded as an OZ problem example. The solution methods seem different, hvr. Most INV problems are solved by converting to Optzn. problems. $x^2 = 10$ is certainly no exception — tho it probly converges faster as an INV problem. $(10 - x^2)^2$ would be a good: or $|10 - x^2|$: If $(10 - x^2)^2$ is used we could fit a parabola thru the last 3 pts. However I don't know how fast this converges: May be better than Newton's method? Getting the peak of the curve doesn't involve solving quad eqns! — it's linear eqns.

.01 From the 3 ~~[illegible]~~ closest pts, one can get the next approx.

x_1y_1
x_2y_2
x_3y_3
b

Parabola: $a(x-b)^2+c=y$

$a(x_i^2-2bx_i+b^2)+c=y_i$

$ax_i^2-2abx_i+ab^2+c=y_i$.

Solve 3 linear eqns for $[a]$, $[-2ab]$ & $[ab^2+c]$: from this, $b=\frac{(-2ab)}{-2a}$.

So soln. of 3 linear eqns in 3 unks,

Since we ~~[illegible]~~ need the 2nd unknown (abc & c), elim. a from the set of 3, & get 2 eqns in 2 unks.

$$\left.\begin{array}{ll} ① & UA+VB=C \\ ② & UD+VE=F \end{array}\right\}$$ Since we only need ratio of U to V: Mult ① by F, mult ② by C, & subtract. Giving $U(\ )+V(\ )=0$

This may work out as easy (linear) in n dims (rather than the 1 dim of present case)

.11 $x^2=10\pm.01$ as an example of OSL! Suppose the TM solved $x^2=10\pm.01$:

Later it was given $x^2=10\pm.01$ again: w. OSL it would realize they were identical & give same soln.

If it were given $x=10\pm.001$, it ~~[illegible]~~ could realize the similarity, & then use the previous soln. as the first approxn. for $x_{n+1}=\frac{1}{2}(\frac{10}{x_n}+x_n)=\frac{x_n}{2}+\frac{5}{x_n}$.

Similarly with $x^2=10.1\pm.01$ or $x^2=10.1\pm.001$.

$\boxed{\text{The details of this "lrng" are unclear}}$ It seems ≈ CBR, which I regard as a form of OSL.

In non-OSL ALP, ≈ MDL is used; & in OSL ALP, OSL can be an aux. search and lead to ≈ MDL. (By ~~[illegible]~~ "≈ MDL" I mean finding a hy. (c model for the {Data + model}.

2.20.01 TM: 1b

.01:39.40 : In Lsrch, we will have several cands that may be "different" but they are all "Hyly" correlated in t. sense that they have identical "success/failure" profiles. In ALP we would add their PCs directly. In simple Lsrch, they are indep trials, each w. its own PC, CC.

In 39.25 ff, .37 ff, I'd like to somehow "pool" these "equivalent" cands ("eqvnt" meaning same success/failure profile) — use only one of t. "equiv. class", (we hope) having least CC. Th. mechanics of how this mite be done is UNCLEAR!

Same is true for t. more General problem of forming t. "eq. classes" of 39.25 ff.

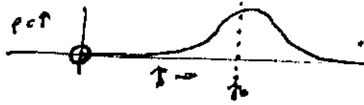
(Actually they are not really Equiv. classes in 39.25ff; They are "Clumps" w. arbty assigned boundaries.

2 solutions: correlts

.12 Say in round n, one tries this bunch of hyly correlated cands: One tries t. one w. least cost ~~[illegible]~~ to cost = ~~[illegible]~~ $T_0 2^n$. Since it doesn't (yet) resolve, it is likely that t. rest of t. cands in this "eq. class" will not resolve w. cost ≤ $T_0 2^n$, so we don't do them this round.

Next round, we try t. single best cand in t. "eq. class" for cost up to $T_0 2^{n+1}$; If it doesn't resolve, we don't do t. rest of t. class for t. same reasons as befor!

I'm uncertain as to t. correctness of .12: ~~[illegible]~~ what is correlated) — I think t. correlation is in t. pd. that t. GPD is t. "margin" of! T. pd. that gives t. prbbty that a cand will solve t. problem in time T.



Say: what happens to t. pd when we discover that this cand. did not resolve in ~~[illegible]~~ time T_0?

.23:39.17 Say t. correlatn info is as it was in .12 ff or, it's of t. form that any ~~[illegible]~~ known failure of a cand
.26 Modifies PC's of ~~[illegible]~~ many other cands. At any point in t. search picking cand w. best $\frac{PC}{CC}$ gives one t. ~~[illegible]~~ (.29)

(SN) Say $P_0(\alpha, T)$ is prbbty that ^(cand) α will win in time ≤ T. P_0 is a component of t. GPD.

P(α) is t. prbty that cand α will solve t. problem in least time of all cands.

It would seem that in $P_0(\alpha, T)$ t. "correlns" would be lower. e.g. — t. meaning of this correln would be different in P(α).

.29 (.24) Best Expected success per unit time: This cand. is t. most "cost effective" cand. Sum 4, 7

However, it is a Maximally GREEDY Algm; Is possible that a Non-Greedy reordering Algm would be better.

On t. Q of whether hints are always reorderings of trials. Well, if a hint tells one to do anything, it is to try certain things first say — which is a reordering of trials.

Any novel pgm (e.g. new / anything different from what one ordinarily ~~[illegible]~~ would do) must reorder t. trials:

If ~~[illegible]~~ gives a certain pgm first order — which could be implemented by ~~[illegible]~~ giving that pgm (≡ cand.) hy pc.

T. only fly in this ointment; Unclear as to when a "trial" ends. Perhaps it doesn't matter:

A trial could take potentially infinite time .... but it s would have some prbby of output befor T=∞!

.34 Any way, a trial automatically ends when its $\frac{T}{pc} > T_0 2^n$. This makes an impl. restriction on
.40 t. form that t. GPD can take! As a trial "progresses", its pc must be monoton ↓. 45.401

.01 : 44.40'. It may well be that most trials never end. (so they never FAIL!).

44.39—.40 is disturbing .... very unclear now, in my mind, as to ~~what~~ what a "trial" is — what a "Cand" is. Perhaps, every time one adds a symbol to t. current cand, it becomes a new cand (?!). However, legit. cands (= working ful pgms) may not occur w. every symbol addition!

In general, cands are generated by a stoch [redacted] Grammar. As such, t. grammar can add on several symbols in one "jump" — or some could only output complete as's.

(I've been thinking in terms of stochastic "Derived" grammars — in which each Step of t. [redacted] Creation process is a legit output w. its assoc. pc.

One fussy: t. Cand has a fixed pc, but it can run for arby long of time searching for solns. Each "round" in T ← 2T, It gets twice as much cc to spend.

We'd like a stoch Gramm that could ~~emit~~ emit Cands in ≥ pc order.

~~It could be a "derived" grammar, if~~ if it were a "derived" grammar then t. additional [redacted] parts ~~to~~ in t. derivation always gives a monotonic ↓ pc. Alas, in general, grammars are not Derived .... Could we always approximately derive a grammar? (or Derive a grammar Approximately?).

5AM 19 28° low 23

Quite Earlier, I was Concerned with t. fact that I might be able to express a Heur as a modifn. of t GP.D., but that getting to output of that pd in pc order might be very diffc!

However: A heuristic (in problem) is able to reorder trials, so perhaps it Should be possbl. to modify t. Grammar in a way so that t. ordering implied by t. heur is implemented.

SN ~~Many~~ Many Kinds of Problems Even within each kind of OZ, INV prob

.25 listed on: Kinds of PDs Many kinds: ~~A~~ Monte Carlo, ALP, input pc range, output same pc and intermediate input cand, output pc, ~~Below~~ Stochastic ~~as~~ Grammars as PD's, → 46.16

Many Kinds of Cands ; [redacted]

Gross Classification?

.27 ......Many ways to IMPROVE t. GPD: ~~Overall~~ Grossly classifies: A P.D. is better

① if it has less $\sum_j \frac{cc_j}{pc_j}$ for t. corpus (but $\frac{cc_i}{pc_i}$ are weighted ~~≢~~ according for "importance" by USR)

② We want it easy to ~~[illegible]~~ put t. GPd outputs in pc order (or maybe $\frac{pc}{cc}$ order) — So this is definitely a factor in "Improvement"s

These seem to be main Criteria

In ① In some cases, TM has some idea about future probs (say he has a "problem pool") So $\sum w_i \frac{cc_i}{pc_i}$ could involve probs not now in t. past corpus.

.27 looks like t. first time I got a handle of 2? specific form for t. "Goodness" of a PD.

2.23.01 : IDSIA
2.24.01 (.30)
Dark
Dark Rite

VERY NICE Result! (.02–.12)
(.30) may be even more impt!
Punch
Punch Drunk

A very nice result!
Very Very Nice | Nice
Nice
Nice

.01 "kinds of Problems" & "kinds of Conds" are closely related, but still there are 3 fund. types of conds:

1) A string that is the soln. itself.

2) A short pgm that obtains the ~~str~~ soln directly from M(.) & a. (for Inv probs)

(.02) 3) A pgm (Lsrch say) that makes several trials within itself — each finding a possl. soln. — and then stopping when one is obtained. Lsrch is a pgm of this type, but here we have the possy of a pgm that replaces Lsrch if it ~~[illegible]~~ empirically works better.

SN In OZ probs & perhaps in Inv probs (as (.02) suggests), a very experienced TM may have only a few OTS & ~~[illegible]~~ pgms for Inv probs & that ~~have~~ very hy pc — & these OTS & pgms do most of the work.

In General, the finding is a nice result! It means that we aren't "stuck" with any particular ~~[illegible]~~ "Global" method of problem solving. If Lsrch is not optimum,

.12 the system has a way of working problems that minimizes the amt. of Lsrch done, [illegible] in the [illegible] search methods
This is a characteristic that we'd like any General System to have.

Also, the pgms & "search systems" can Recursively call other search systems: including the "Top Lsrch" function a/o itself.

.16 : 45.25 SN On Kinds of PD's: The ALP, Trmc method of dvlpg a PD is particularly good for Lsrch. In general, any PD that enables us to obtain conds in ~~[illegible]~~ ≈ pc order, is of much interest. A big Q is interconvertibility of various PD types: what's cc?

Perhaps Trmcs or "computers" are the only kind of PD that realizes of the form "given a range of pc's: output, get conds w. pc's in that range." If we had a PD of that type, we could probably convert it, directly, into a Trmc. (probably a 1→many xform).

Say we had this "PD" form: (well, any PD is convertable (1→Many) into a Trmc form — that's not interesting — but perhaps the "order of pc" PD is easier to get into the Trmc form than most other PD forms & — "easier" means low cc, & this is important.

If we are given a PD & we are asked to build a Trmc to simulate the ALP, I think one has to have access to the arguments of the PD in pc order (hy-pc first.
My impression is that Huffman coding is usually the way its shown/implemented)

(.30) 2.24.01 On kinds of PDs: Perhaps standardization ~~[illegible]~~ Kt representation (Min Lcost)

When we insert a hour, we do this via a Simulated TSQ for the hour. We ~~[illegible]~~ use the resultant Kt as a "Summary" of the induced PD.

Def

Its not Clear in My Mind as to just how the "Leoda" ("Lsoln") approximates/summarizes the prob solns. — & in particular, how it does it for Tm. hours (how it "modifies the GPD").
= output of successful Lsrch

It may be possl. to generalize Kt to include (& use) other short codes.

.37 Another big Q: Re: Hours: How can we incorporate ~~[illegible]~~ (into TM dev?) hours that were logically/mathematically discovered? At first guess, it may be possl. to show that TM could obtain such hours by normal Lsrch for hours (or something close to it.) I really haven't analysed this kind of problem much!

2.24.01 : [Dsin

Canker sore
Canchre

$\boxed{\text{Sort } \frac{cc_i}{pc_i} \text{ order}}$

.01 : (28.03R): Inv probs w. Expensive "Judge" Tester: If output of tester Y/N only, then there is nt much one can do, but Lsrch — using as much problem Info about t. problem as poss.

It is perhaps possl. To have more info than t. pure Apriori for t. Solution:

revs:
40
38

One mite have "correlns" betw. cands, so perhaps clumping of t. cands, w. a representative for each clump: Run test t. representatives first; See 44.23 t assoc. ~~[illegible]~~ forward backward refs

In general, ~~[illegible]~~ in Inv probs, If one only has access to t. Yes/no output of t. "Judge" then that's too bad: Lsrch is t. best that can be done; If t. ~~pc of t. soln is very low~~ cjs of t. soln is very hi, then there is nothing one can do about it.

If the (Judge for M(X) = a (many) ~~[illegible]~~ is "open" (i.e. the internal operation of M(.) is visible to TM), then more can be learned from failures than just a Yes/No output. We have t. traces of t. tests made by M(•). — So TM tries to get as much info from them as possl.

On TM2G: Is minimum ~~optimum~~ of $\sum w_i \frac{cc_i}{pc_i}$ a reasonable Goal for TM2?

A trivial soln would simply give pc = 1 for all pci's. Then why is this illegal?

Well, t. GPD is supposed to be a ~~[illegible]~~ legal compression of t. data.

(I guess t. only legal way that we can get a pci is to find a shorter ~~[illegible]~~ ~~a/o more~~ coding for it. ← No!
Also, it may find ~~[illegible]~~ a shorter code for a different soln that solves t. problem FASTER.

The way heuristics do this: The heurs are based on statistical data & personal bias, they affect t. GPD by giving higher pc's to cands that will solve t. problem.

.01

.01 : 46.40 : 46.30 ff ~~a~~ looks important! Perhaps it more or less solves all ~~[illegible]~~ en of of t. probs of TM, so I can start on TSQ's again!

I do have to work out "t. Mechanics" of 46.30ff, hvr.

T. way "Lcode" summarizes: For sequential predn., we save t. state of t. UMC, after t. Lcode has been run. Then insert any random code into t. UMC, i.e. t. "Mt.Carlo" pd. at its output.

[Note that OSL is not implemented in this summary ~~[illegible]~~ so this : "Logically obtained results" (46.37) have to be worked out]

O.K.: Say TM solves a certain problem: Not Sequential predn, but "BAG predn".

"2 kinds of prob. in t. paper" - Solved this problem. I think t. means TM found a Grammar w. a start code (Lcode) that had t. observed output as [As set] — Re its a stoch grammar, so we have to include the ("random") choice to convey that obtain t. observed corpus.

[Actually, t. "Summarizing Machine" dcrbd in this paper was for a different method .... perhaps not as relevant]

— What we want is closer to t. (MDL / MML) method.

Look at §3.3 of that paper. We have $M_0(\alpha_M, s) = M(s)$

We have, (via Lsrch) found a α_M ∋ that we have been able to find codes for t. corpus [D_j]

so $M(\alpha_j) = D_j$ ∴ $|\alpha_M| + \Sigma |\alpha_j|$ is "small" (in t. Lsrch sense, t. α_M & α_j's are what we found.

We could have a Machine, M_1, & a common prefix ~~[illegible]~~ α_M, for all codes of t. corpus D_j.

So $M(\alpha_M \frown \alpha_j) = D_j$ (for full ALP, we use many codes of different Machines, $M_0(\alpha_M, s) = M(s)$),

pc of corpus $= \sum_i 2^{-|\alpha_i| + \sum_j |\alpha_{i,j}|}$ α_i is a general t. prefix string. $\alpha_i \frown \alpha_{ij}$ is a code for D_j via $M_0(\alpha_i, \cdot)$

Hvr, I've left out other ~~[illegible]~~ codes for t. corpus via α_i (ambiguous Grammar).

$$\text{So } \left[\sum_i 2^{-|\alpha_i| + \sum_{j,k} |\alpha_{i,j,k}|} \right] \leftarrow \text{is t. ALP of t. corpus.}$$

$\alpha_{i,j,k}$ is t. kth code for D_j via Machine ~~Machine~~ (Grammar) $M_0(\alpha_i, \cdot)$

So L ~~[illegible]~~ srch gets us t. Lcode for which $|\alpha_i| + \sum_j |\alpha_{i,j,k}|$ ~~[illegible]~~ + $\log_2 T$ is Min

.24. How this code is found by Lsrch? $\Sigma\, \alpha_{i,j,k}$ can be very long!

Given α_i, t. search for codes for t. corpus shouldn't take so long! For some Grammars, there is a "Parsing algm." that reduces cc to ≈ φ! These parsing Algms should then be regarded as heurs!

If so, how does TM discover & implement them? We, consider t. ANL ~~[illegible]~~ TSQ(s) of SAARB!

Well, t. ANL TSQ in Saarb; ~~[illegible]~~ T. problem was to find an operator that mapped Alg notation into Computer notation (Machine code), It was a simple kind of INV problem because t. soln. was close to ~~there~~. ~~[illegible]~~ t. structure of t. Machine used.

In most BAG-type probs, t. problem seems much harder. It can be framed as an INV problem to find a small Grammar & its derivation; or just an INV prob that finds a Gram + derivations that is of min Cost. It would seem that Deriving a Grammar for a BAG, would be a very diff't problem, in most cases, & shouldn't be given to TM until he was "ready for it"

Some Grammars are, hvr, quite simple: e.g. we can have t. Boolean "Concepts of (Computing) Values & Learning Theory" "heavy ~~heavy~~ set but not long" Corpus is set of objects, each w. 3 properties.

The set of Grammars, is a list of Boolean expressions

— → Still, I haven't gotten examples of implementing such a heur by modifying GPD!

This seems to be a main problem now. Hvr, I haven't tried implementing OE solns,

50.01

2.25.01 IDsia

TM v.s. "Lrng how to Lrn"
Sol v.s. Juerg.

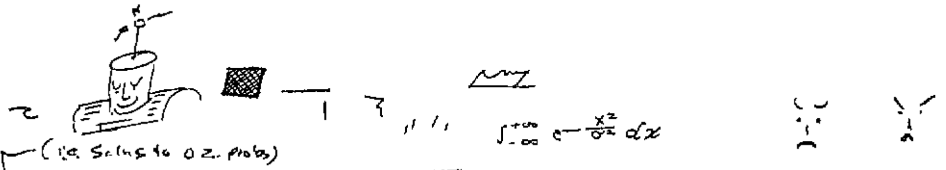
-.01 : On f. General format TM: Initially, we could start out w. TM not knowing the difference betw. .00 Sequ. predn., Bay predn, INV probs, OZ probs. We just give it probs, & it solves them by Lsrch. Then when it has a large no of prob, soln. pairs, it tries to Modify its GPD so that the solns would have taken less time.
first, it figures out those classes and

This (-.01) TM would be closer to Juergen's "Learning how to Learn" Machines, but it would be very specialized in that it would be given "clearly separated" problems, and it would be told what types they were (4 types in .00) and given preliminary ideas on how to solve them (Lsrch). Also, it is told how important each problem is.

AND, it works on "Main Problems" & "S.I. probs" as separate things, & I control how much time it spents on each.

Still, I would like less of prelims detailed input. One step I made in this direction was to ~~[illegible]~~ Give TM a "pool" of probs & have him select which to work on first.

2.25.01 IDSIA



(i.e. Solns to OZ probs)

.01 : 48.40: "This" would seem to be straightforward! I would begin w. a good set of OT's — & try to "factor" them ("By Hand") so TM may try to devise new them by recombin. of t. factors.

.01 Hmm, for TM to decide which OT is best for a given problem, is a more dfflt thing.

Heurs may be simply solns to induction problems. The Data for t. induction, is a set of problem, soln. pairs. From Them t. GPD is created, as a cond'l. PD. Hmm, we can't just use

.04 a MDL. or a "t." "Lcode" for t. data, we need many problem-soln outputs for each input to t. GPD.

So: DOES .01—.04 finish off t. "TM problem"? Well: in .04, I need a PD as output ~~output~~, not a single Lcode. In particular: I will already have a GPD; & I have new data for it — so the old GPD will be Modified by t. New Data.

Anyway: In spite of .04, ~~finding a PD~~ modifying a PD in view of new Data, using t. Gore $\sum \frac{cc_i}{pc_i} w_i = \min$ (18.07 treats from i to) Is a "well define problem" that we can ask t. GPD for a PD on ways to work.

In fact, we can ask TM to work any kind of problem (Just be sure there's enuf info to define t. problem, & that TM has enuf backgnd. to work on it in a useful way)

This include OZ probs ~~that~~ w. G(x)·F(t) type of Gore! — & problems that I don't know how to Solve, but have some idea as to what skills are needed, so I can "prepare" TM for them.

In t. case of .04 I'm not yet clear on what form "an acceptable soln." takes. The cands are, indeed, strings (used as inputs to t. "Reference Machine"), but evaluating these cands, could be quite expensive.

.18 But for .01—.04: T. idea is that we insert those prob. descns into TM. (T. problems may or may not be categorized into types like seq. predn, bag predn, Inv, OZ). T. GPD takes each problem as input & presents as output a pc ordered seq. of cands: highest pc first. We do Lsrch on this set of cands until we solve t. problem. This takes time T_s; we then may spend time T_i trying to "improve t. GPD". This is a recurrent problem, & it is found to t. PD for a cand list, which we "Lsrch on". | Adequate time to spend dep on S, E, & Ti

We then give TM t. next problem.

The time TM spends on "improving GPD" is controld by USR. Not much time should be spent on it

.25 until TM has acquired enuf skill to be able to work on it usefully

So Now I want to write Up this TM Model in as much detail as poss'l. — First a rough outline, like .18—.25; Then go over it in more detail; Then go over it in even more detail.

Try to Give Bibl. refs. if possl.

.30 NB In general, t. ~~first~~ prelimy output of GPD, that is obtaind when GPD is "improved" is a pd over t. time needed for a given cand to solve a given problem. These PD's are integrated to give t. prob. of each cand, being t. fastest soln. to t. given problem. This is discussed in t. ~~ec~~ MCT analysis (~1998) [redacted] for OZ problems: But it also has to be done for Inv. probs. (Unclear about Bag Seq predn)

In present form, TM is capable of solving problems using non-Lsrch. (i.e. ~~the~~ t. GPD can assign (for a problem) a prob-solving routine that is a search, that is not Lsrch, & assign a very low pc to other methods of solving that problem.).

A BIG Q: Can th. present system also effectively "by pass" my external routine that decides on "k" t. fraction of time spent on S.I., & own details wts ← 18.07

51.01.

Minsky@media.mit.edu

.32 correctim

50.90

.01: Λ in the GPD ~~s~~?

Possibly it mite ~~[illegible]~~ be able to do it if t. Sys is a "**VERY LARGE Problem**"

[This is t. "Very large problem", perhaps, that enables TM to Devp R.W. get "Ego", self awareness, etc.]

Sounds reasonable! ~~[illegible]~~ A ~~[illegible]~~ Cand for a large problem could decide that a more ~~[illegible]~~ of its "own" time should be spent on S.2. : But allowing t. Cands to be "supervisors"! Is this wise?

A Cand could really clobber t. system! This is mind ful of ~~[illegible]~~ company's system that is ~~[illegible]~~ guaranteed to clobber itself eventually.

It will prob'y be nec'y to make some restrictions on this "supervisor" mode: When it is invoked; be sure a recent backup (mode) is ~~[illegible]~~ available) — also devise scheme so that prob'y of "suicide" ~~[illegible]~~ is very small (≈ 0).

.11 → [A milder "supervisor" could only decide to ~~[illegible]~~ spend time on S2 during a cand's life. This could only affect two K, ~~only~~ (This is o.k. — if default is K = 5, decreasing K can't ↑ speed of system by > 2; but ↑ K can ↑ speed of system by very large amount). Spending time on S.2's cost be useful in a "large Problem".

A "Very large problem" mite be to work on "t. problem pool". or "Cure Cancer".

⟶ Could we get TM to use logical/math'l reasoning as part or all of a cand? Well its certainly "Legal" since ~~[illegible]~~ any string is a legal Cand. A Q is — ~~[illegible]~~ would TM ~~be~~ be motivated to do so? Perhaps it could be given ~~a~~ simple problems to lead it into logical analysis, & mapping logical analyses into its ~~[illegible]~~ previous learning ~~[illegible]~~ about "Symbolic Logic". A type of "Analogical Reasoning".

.25 An easy Summary of t. TM System:

1) Explain Lsrch: Its 3 properties ⓐ Using $T \Leftarrow 2T$ Time to solve problem $\leq 2T_0 \cdot 2^{K_t}$ To it time to generate and test correct soln. (K_t soln.).

K_t is Length of code. " Time Share search " " " $\leq T_0 \cdot 2^{K_t}$

ⓑ If there is an optimum Algm, A to solve a inversion ~~[illegible]~~ problems, Then since This algm takes ~~[illegible]~~ time $F(n)$ to solve problems of size n, Then/Timeshare Lsrch will take time $\leq F(n) \cdot 2^{\ell(A)}$; $\ell(A)$ is t. length of descn of A using same machine as is used for Lsrch.

~~[illegible]~~ (c) Lsrch can also be used for OZ problems. (explain w. fixed time (~~[illegible]~~ or ∞) limit.

.32 2) The TM system consists (mainly) of a GPD (Not an Emunity!) This is a conditional PD. Its input is a problem descn. Its output is a listing (in order of prob'y) (w. prob'y ~ 2nd is approx of prob'y.) of strings that mite solve t. problem. T. strings are regarded as psms. T. "prob'y" is t. prob'y of a particular string (= t. list strings) having t. fastest/solz to t. problem.

3) The system works in this way: We insert **problem Into System**.

It goes to GPD, which gives a list of Candidate strings and associated probabilities.

The system ~~Ru~~ ~~[illegible]~~ uses an Lsrch (using the list and probabilities) to solve the problem. This takes time To.

2.26.01 IDSIA

REV

.011 ~~[illegible]~~ 51.40: After Solving the problem the system spends ~~at least~~ about T_0 time (or sometimes more) "Updating and improving" the GPD.
It then works on the next problem and repeats the above operations, etc.

.02 4) The outputs of the GPD will be termed "Candidates" (Cands). A cand. can be ~~any~~ almost any program for a ~~[illegible]~~ (½) universal reference computer, that maps the problem down into a possible solution. (an attempted solution) Some limitation on these Cands: While they can do "calls" on the system as a whole, they cannot ~~modify~~ directly modify the GPD, or modify the way in which ~~the~~ the system spends time "improving, updating" the GPD. It can increase the amount of time spent on improving the GPD, but this increase comes as a time cost ~~[illegible]~~ when its own behavior is evaluated. Say the ~~program~~ Cand spends time T_1 ~~on~~ in incrementing ~~the~~ GPD improvement ~~[illegible]~~, & spends time T_2 actually solving its problem. We may regard T_1+T_2 as being the time needed to solve the problem.

.13 5) A cand. can be a search technique that may ~~[illegible]~~ be Lsrch, or it could be any other method of solving problems.

.15 6) Criteria for Improvement of the ~~GPD~~ GPD ~~[illegible]~~ ← [.18–.27 is prob WRONG! – See 58.01 ff for Ritollage!

.16 ⓐ ~~Minimum~~ If the only problems solved were ~~inversion~~ .INV problems, then we want to modify the GPD so as to minimize

.18 $G = \sum W_i T_i 2^{K_{T,i}}$. T_i is the time taken to solve the i'th problem.

W_i is a weight given by the user, $K_{T,i}$ is the K_T of the ~~[illegible]~~ particular solution obtained.

G can be reduced in several ways: ⓐ we can assign higher probabilities to the solutions that were obtained, keeping same T_i.

ⓑ we can find new solutions to the problem and modify the ~~[illegible]~~ GPD so that both T_i and $K_{T,i}$ are changed ~~[illegible]~~ and ~~[illegible]~~ $T_i \cdot 2^{K_{T,i}}$ is reduced.

~~The~~ Modifications in GPD can only be allowed if they are the ~~[illegible]~~ expressions of legitimate regularities found in the data. .27

The legitimacy of regularities is judged by code-length considerations.

Needs expansion

ⓑ If some of the problems solved are not INversion problems, there are

.30 terms corresponding to those in .18, but their exact meaning is more complicated.

7) Improving the GPD, once we have an objective criterion for ~~[illegible]~~ improvement, is a well defined problem that can be worked on by the system.

53.01

2·26·01 IDSIA.

REV

.01 : 52.40 ; Areas Needing Expansion

1) 52.30 optim forms for OZ ~~recipe~~ and perhaps prodn. problems

2) The exact mechanics of How t. GPD is related to "t. DATA".

e.g. What is considered data for a problem soln.; The 2 kinds of P.D.'s that GPD expresses. — (50.30)

How t. part of GPD used for Lsrch is an integrated Map of the other P.P. ——→

(See work on MCT for analysis of this.)

3) List of 4 problem types: Examples of each: The variables within each of t. 4 categories.

An impt Category of Inv & OZ probs: When the Fitness test is Expensive. For INV probs, this might invoke a kind of Quick Abort — to economize on Full "Fitness test"

Example: Fitness Test: Meet all of 4 criteria: We can ~~[illegible]~~ then decide for each cand, Which tests to do first to make the "Quick discard" possl. & save on cost of rest of tests.

.13 SN On t. Non-Optimality of ~ "Lsrch": In a ~~[illegible]~~ OZ problem, after we have tried a certain OT for a while, we may realize it is not likely to work, & we will use this info to revise our prior probs estimates of other OT's being "Best" in view of these new Estimates,

JUMP to a new OT with (now) highest pc of being "Best".

SN Note that for OZ problems, the cc for each OT is t. same (= cc_0 given in the OZ problem data.)

So t. order of trials $\frac{pc}{cc_0}$ = simple pc order.

Similarly in Lsrch w. INV problems, either partially thru a trial of a cand, or After t. cand has been tested, TM may revise pc of other Cands in view of this info.

Again, the best choice is Cand of Max pc, but if we have estimate of cc of Cands, We should use that info, too (The just how is unclear — if we only have cc estimates for a few cands!

—— BAG

.24 SN I recently had t. idea that t. simple ~~Bag~~ induction problem was ~~[illegible]~~ much harder than the Stochastic operator problem. In fact they are both about t. same difty. We have to find IN BOTH (2) model (s) ⇒ (pc of model · pc of data in view of Model) is Max

IMPT PROBS in t. system that need work: Z144

.29 1) How t. system Uses Mathcl/logical reasoning to help {solve problems/creat heuristics} is not clear.

2) How far Can we Go w. (Augmented) Z144? Can we use it for t. GPD? Is it possl. to incorporate all [illegible] types into it?

3) T. writing of TSG's!

.01 : 53.40: It would seem, that t. fuzzy. stuff should be adequate to get a working System!
51.25 ff

Lsrch for induction (on GPD for example) as well as other problems, is, indeed, ≈ optimum, if one has all needed info in t GPD.

So a Q. might be: is Augmented Lsrch adequate for putting "all kinds of info into t GPD"? (or more Specifically — Heur info)

Another impt. Q is 53.29 on how to implement ~~the~~ Logical reasoning into heurs..

I want to be sure that I really remember t. impt. details of 51.25ff, so I can return to this, if I should forget! I will want some good "notes" for talks at IDSIA — So perhaps those notes could really say all I said (or implied) in 51.25ff.

Some past problems that 51.25 ff seems to Solve:

1) ~~What~~ How to order t. utilities of PD's. (Here t. "PD" includes cc of obtaining pc's)

2) Just what are t. Conds; ~~[illegible]~~ If they are pgms, what are limitations on these pgms (if any).
This was a big diffy in my last round of attempts to write TSQ's.
[redacted] Take a look at that stuff. Does 51.25 ff really solve all t. problems?

.20 TSQ Probably early Bulg (2000).
Bulg 176.21 ff had some items: 181.5 is ~~[illegible]~~ P.o. of an TSQ. ← mainly on "Addition" & its Properties.
Ideas in LISB could also help a lot in ~~[illegible]~~ those TSQ Attempts.

In 181.5 it was very unclear as to just what I was teaching TM!
181.5 is in a little sub-folder 8 pp on TSQ's.

.20 Reading 181.5, I see that problem of constructing a TSQ is quite unclear! → .34

On TSQ's: For Algebra! One way to lrn: learn lots of isolated little rules — Like cancellation, assoc, commute, etc.; ~~the~~
exception of cancellation for mult (zero) is ~~[illegible]~~ fine; TM does often make rules "w/ exceptions", if they are compact on t. whole.

Do use large ~~[illegible]~~ expenses for m+n, m−n, m÷n: perhaps how m×n, m÷n cost much more than m+n — so logarithms could have utility in ~~[illegible]~~ arithmetic.

We want OSL in this early TM — e.g. to remember recent experiences;
Say it just did lrnd m+n = [m+n] then it lrns n+m = [n+m] & it notices [n+m] = [m+n]

For induction problems, we may want lots of built in heuristics (along w. facilities for TM's adding in more heurs for induction).

.75

.34 : .20 One diffy was ~~[illegible]~~ to know TM's need just what was needed in each problem: How much info was "local" to t. problem: How much Info could be carried over from t. previous corpus (Global).
I did have a way to indicate this.

.3 Looks Good An approach: Just write down a bunch of Q's & Answers that I'd like: For each such QA pair, write down what assumed info is: what's "understood", what are conventions, constraints? what is "local" to this QA, what is Global to entire corpus. → 55.18 sec

.01 : **ON OSL :...**

● I have been thinking that $\overset{\text{MML}}{\text{(MDL)}}$ ~~[illegible]~~ isn't able to deal w. OSL:

This is true for usual way of using MDL: To make a "2 part code" w. MDL for the PD corn and "simplest cost for the prob'y of the corpus in terms of that PD". In which case "MDL" gives ~~the~~ v.g. continuous pc values: but it can't do OSL this way.

Instead, if we use "naive" MDL ($\equiv$ $\overset{\text{Van Heerden Method.}}{\text{VHM}}$) perhaps OSL will be implemented: — approximately

.07 Hvr, since only $pc = 2^{-n}$ values are possl. we wouldn't have much precision.

The objection of .07 is a normal objection for Lsrch, and is a normal defect of it that I have worried about — it does give "signif" (?) deviation from $\frac{\text{OSL}}{\text{ccy}}$ ordering.

.10 Would we do better using "2 kinds of induction" ① "OSL" ② "Non-OSL": ?

"OSL" is ~~[illegible]~~ a specific search for single past occurrences ~ to the ~~[illegible]~~ present problem.

NonOSL is — Making a probilistic Model of corpus and expressing corpus in terms of the model.

This "probilistic Model of corpus" can be (but need not be) MDL.

What may happend: We start out w. Pure Naive MDL and Lsrch. and as we mature, We discover the "2 ways of doing induction" of $\boxed{.10}$

.18 ● 54.40 : TSQ! 54.37 Seems like a good idea!

.19 So: X + 3 = 3 + ? Assum: X : Understood! X is local to this problem: it is a variable;

.20 ~~[illegible]~~ — we may or may not know its "value". [Re: the answer, it is very likely; $pc \approx 1$]

X + 3 = X + ? answ. 3 : Because: $\boxed{.19-.20}$

3 * X = 6 ~~[illegible]~~ : ~~[illegible]~~ X = ? ; Asw 2 $\boxed{.19-.20}$

In the foregoing problems, If I don't tell TM what the "usual Assumptions" are, it should be able to **induce a** ~~a~~ **P.D.** on them, in view of previous (problem, soln) pairs.

Usually, the "Understood" info is quite clear, and I (e.g.) have no trouble guessing what.

— Usually, the "understood" info is so likely that I am unaware of ~~[illegible]~~ any alternatives.

"**Conc Nets**" are still a very useful (perhaps essential) tool in writing TSQ's:

Several impt "concs" can be the "Unmentioned Assumptions" about the problem.

— In general, if there is Ambigy about the Assumptions, there are (usually) only a few possys, and TM can write a soln. for each **case**.

●

2.2801: IDSIA: CIAP (conciousness)

Kay 40

–01.'.27.22 : on "Conciousness" in TM: In the continuum of intelligence of TM There is a certain region where conciousness in TM (~~an~~ awareness of outside world) could be very dangerous (Tho it would enable TM to grow very rapidly!)

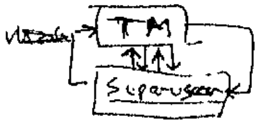
.02 If we prevent "toy data", "toy seqs" ~~to~~ well before that pt., w. selfawareness

.03 as a goal, we can achieve it much earlier, w. little danger: — So we can "turn TM off" if it looks dangerous at all!

At very hy levels of TM intelligence, it will be diff, if not impossl., to prevent TM from doing "RW".

The less intelligent, "concious" TM of .02 - .03 could be used as a demo, to show people that "consciousness" is possl. & that machines of this sort can be dangerous.

On "T. Pleasure Machine": J. McCarthy once said that he wouldn't ever tell a very smart machine to do things that made him (McCarthy) Happy. Upon much thot, I decided he was right. Hvr, it appears that any user (say a superuser that has access to much of the machine & can reconfigure/reorganize it), will try to adjust/modify the machine so it pleases him most! This is automatic: The Human trys for this Goal & the Machine trys for this Goal.



2 smart machines in a very "loving" relationship. TM is partly "open" to the s user

Logical Reasoning

20.10
.01: 55.40 : On Logical Reasoning in TM ~~[illegible]~~ in devising Heurs'.

Since a Cond. can be any pgm, it would seem that "Logical Reasoning" would be included. Hvr: For Heurs. t. conds of interest are for Induction — for "improvement of GPD" — t. relevance isn't so transparent!

But any way, it seems likely that TM could use Analogy to apply Math to its own problems, after having used Math to solve its regular (externally furnished) problems. E.G. we could give T.M. as a "Math problem" finding faster ways to do particular math problems. After some success at this, TM would be in a position to discover t. relation of t. solns. to these ("external") problems, to its ~~[illegible]~~ own ("internal") problems.

Certainly we can teach TM to "reason" about Math: This process is simply a seq. of Math problems. It can also learn to "reason" about how to write shorter pgms. What we do, is take a problem in which I ~~[illegible]~~ have found a short way to solve a problem by "reasoning" & we get TM to lrn. t. needed operations, strings, "concepts", so it, too can do t. same thing. Also, it will be able to solve "Similar" problems.

Some General Remarks about t. System proposed!

.16 1) If we use an ordinary umc to start, and no particularly good OT's (no specific OT's etc.(1)), with — The system will take too long to do its Lsrchs for any practical Utility. By "too long" .18 I mean too many operations — not no. of examples (≡ "ssz"). A pgm of 10 Bytes length → $2^{80} \cong \underline{10^{24}}$ trials.

2) We need at least some Good OT's and some assignments of them to various problem types. The 4 problem classes: seq. induction, Baf induction, Inv, OZ problems are useful, but within each class, specialized ~~[illegible]~~ methods are needed. For t. problem of improving t. GPD, we need all t. help we can get, as early as we can get it! w.o. a good method of GPD improvement, we will not get heurs for to external ~~[illegible]~~ (final) problems!

3) In (1) (.16): It may be possl. to use minimal heurs ~~[illegible]~~ ("min improvement of GPD") & a suitable TSQ, to bring TM to t. point at which it can solve problems like "improve GPD". I think this is what I had in mind in Sol 89.

What I'm thinking about now, is a strong "jump start", in which ab initio, TM has ~~[illegible]~~ for t. "TMZG" problem (≡ improvement of GPD), a good set of OT's — possibly in "Factord" form.

4) T. main Advantage of "Long how to lrn" is minimal "amt of" All info in system. It "understands" Every thing it does & so it uses what it knows appropriately. ∴ less "Brittle" —

.30 → Also, after a suitable t.SQ. it is very likely to continue as well on a less carefully constructed TSQ.

TM2G

.01: $\overset{1945}{20.10}$ T. idea of $\overset{52.15}{19.05}$ on TM2G: $\sum w_i \frac{cc_i}{pc_i}$ Maybe arbitrary wrong! Recall that GPD has 2 parts GPD$_1$ & GPD$_2$: GPD$_2$ is obtained by interpolation & gets a prob'y for each cand being "best" & is used directly for Lsrch.

Here GPD$_1$ is a p.d obtained from a sample set and it has a clear GORC. — That it decodes t. data "best" is its pc of data of PD$_1$ times P$_0$ of data ~~sort~~ a Max.

Re "wts:" would perhaps be like γ_i — powers of t. pc's. A small power (≈ 0) would make t. corresponding pc's of less import. Equivalently: $\boxed{\text{maximize } \sum \gamma_i \ln p_i}$.

There are 4 kinds of terms in $\sum \gamma_i \ln p_i$:

1) & 2) are sequential & BAG predn. ~~[illegible]~~ { This is the ~~[illegible]~~ includes stack operators ~~[illegible]~~ / I'm not sure it includes GPD$_1$ or GPD$_2$

3) INV., 4) OZ:

In ~~[illegible]~~

All prob types t. input is t. prob. distn. ~~[illegible]~~

In both kinds of predn problems, t. output is a pred — a p.d. on poss ~~[illegible]~~ "continuations".

We want to maximize $\sum \ln p_i$; where p_i are elements of t. corpus.

In INV problems, for each cand, we want GPD$_1$ to give a pd for how long it will take that cand to solve t. problem, so we want to maximize $\ln p_i$: p_i being t. prob'y of t. time in t. data, that is assigned by t. cand.

In OZ prob; A cand is an OT: ~~GPD$_1$'s output~~ GPD$_1$'s output is a D.f. on probability that t. OT will get a certain Gorc in t. stated Timelimit. So it p v.s. G. p(vs T)

In INV probs GPD$_1$ wants to predict how long it will take cand to solve t. problem.

In OZ " " " " t. G. obtained by t. cand in t. Timelimit.

So t. total Gorc is t. wtd sum of t. logs of t. pc's of t. data that actually occurred.

Q: GPD$_1$ is a predn. problem. I think it is concerned w. itself. — So how does t. TM2G work in this case? Perhaps it's unnecy to include GPD$_1$ in the optzn ..... It is already optimizing itself! So ~~[illegible]~~ probably "No Problem".

.22 See Juergen 3/1/01 (5.24) for serious criticism of 58.01 ff! → Duplicated at 58½.24

.23

IDSIA
3/1/01 ~~Juergen~~

58½
(duplicate of Juergen's)

.01 : New Outline of letter to J: Started w. maybe ≈ 2.01–.40

Then draft: In preparation for my trip to IDSIA, I have been reading some our old correspondence. As ~~Back [illegible]~~, you are the only person I know who has written any ~~[illegible]~~ useful criticism of my 1986 and 1989 papers!

Needless to say, things have changed in the intervening years. I said that ~~[illegible]~~ Lsearch is about optimum if one isn't able to learn (i.e. modify the guiding probability distribution) ~~[illegible]~~ between trials. I would weaken this statement in ~~2 ways~~ Two ways!

~~[illegible]~~ First: even with learning, it is probably the best Greedy search method. If one "learns" between trials, however, ~~[illegible]~~ upper than T_i/p_i (the increases (jump size) is not longer usable as an estimate of total search time.

Secondly; The system ~~[illegible]~~ CAN use non-Lsearch techniques if it finds they are appropriate. This is because the candidate solutions to problems are arbitrary ~~[illegible]~~ programs in a universal language. It is unlikely, of course that anything like this will occur, until the machine has become very smart.

You were concerned with the "Global credit assignment" problem. I have found a fairly good ~~solution for~~ criterion for Improvement of the probability distribution that guides Lsearch. I called it "The Mixed Corpus Theorem". It enables a unified measure of the utility ~~of~~ a

.24: 58.22 This sequence should be ~~at~~ 58.23 ! Woops! Suppose I have 2 PD's: ~~A~~ A & B: A ~~[illegible]~~ assigns a higher PD to a corpus than B; But B takes much less time to compute! (SEE 60.38 for How to Deal w. this!)

IDSIA Reference

E.g. B could be ~~[illegible]~~ RLP with large resources: A could be RLP w small resources. Which is better? How much resource should I use? I think this is the same Old problem that I never solved very well! Solns?

Well, say I apply A & B to the corpus: which gives the fastest problem solns? This seems closer to the $\sum w_i \frac{cc_i}{pc_i}$ criterion: — but ~~this~~ was this criterion ~~[illegible]~~ adaptable to OZ (or ~~[illegible]~~ predn) problems?

Perhaps A using the would put ~~[illegible]~~ higher pc for its ~~p.d.~~ ~~[illegible]~~ using the method of option at 58.01 ff.) (These w_i could be the expected frequency with which probs of that type occur)

but it would end up w. a larger expected cc for the entire corpus.

If we do INV problems only, then $\sum w_i \frac{cc_i}{pc_i}$ ~~[illegible]~~ looks Good & takes into account the cc of computing the ~~[illegible]~~ PD_2. For Predn & OZ probs, the criterion is not so clear! — because we have ~~tradeoff~~ tradeoff betw. predn accuracy & cc; or OZ Goal achieved vs cc.

Consider that, in both Predn & OZ problems, the total cc is (constant ~~stated~~ stored). → IDSIA 59.01.

3.1.01 IDSIA

~~Abdeo~~ Mader Dup ~~effect~~ | J vargas Pya ~~to~~ 25 (~~IDSIA~~ 58½.)

.01: (Juergen: 5.40: 3/1/01 58.40) → So in Predn's OZ probs, we can tell which P.D gave better results in any particular OZ or predn. problem! But to gage relative utility of OZ problems, we may have to "Linearize" t. Gorc".

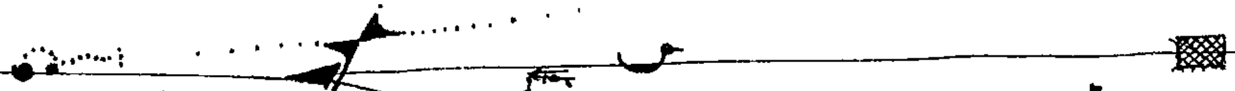
A Not Bad feature of forgy: That each problem contributes additively to t. total Gorc; so one can evaluate t. "Credit to a certain problem (set)" of certain trial modifs of t. GPD.

It is possl. that "Linearizing t. Gorc" is necessary: that there is no way of doing this to t. extent that one's Utility is an arby (?) fnct of t. Problem outputs — this will have to be done.

.07

So t. forgy. may be Adequate.

A longer term Q is: t. forg. is optimization wrt t. past only: If TM has ideas about future problems — it is not appropriate: e.g. A child must t. eventually contribute to TOE (Physics) — so success in Math & Physics studies a re-gram much wt. wrt. [illegible] expected future applicns. It may be possl. for TM to treat this as an OZ problem, w. fix'd horizon.



.16 I'm not so sure about what t. whole proposal in 58½.24 — 59.07 is!

— Whether t. optzn is done in GPD₁ or GPD₂!

Some ~~[illegible]~~ ways to solve t. TM2G problem:

.19 (1) For all of t. problems, ~~new~~ Gorc is Max GPD₂ for t. solns that were found! In t. case of INV probs: t. ~~[illegible]~~ pc of t. problem solving technique that found t. soln.
for OZ probs: " " " OT " " " (i.e. "Best soln. thus far").
" OZ probs → Anytime problems, it may be t. same TM2G as ordinary OZ probs.

} We also have constraint that these by PC's must be obtained in a (logical) Comparable manner.

Re: ~~[illegible]~~ "Any time" probs: They can be worked on as time sharing betw. OT's. — in which case a "best so [redacted] far" [redacted] result is always available.

In both INV & OZ probs (we do take into account the cc used in all parts of t. solns). (How?) ((I think)) ? see 60.27 on DO → 60.27

.26 In predn. problems & some OZ probs: we may, for certain problems, have [illegible] pc for certain OT's, in which case, we don't try other methods much (if at all!).

So we end up w. more & more wt. for those OT's. ([redacted] ≈ a "Self Conforming Hypoth.") → 60.03 may help

.29 A smarter TM₂, would avoid this TRAP! How it would do it is unclear (☹).

.30 An additional diffy: It seems to me that there probly are only a few (maybe!) very general prob. Solving OT's that would be ~~[illegible]~~ ~~[illegible]~~ "really" Best.

So I want [redacted] TM NOT to ~~[illegible]~~ use a such method that does not (relatively quickly!)

.34 Stuck at a local optimum. → 60.03 may help

.35 SN For OZ probs: Since t. Time used to test each O.T. is t. same, we really simply try them in PC order! — This seems rather strange that KT should be irrelevant! In t. "Anytime" form, perhaps KT is important? Look at my "Oxb." report on "Optimum Seq. Srch"! In t. OSS report Notes, I was clarified: If one has ~~[illegible]~~ cands in ≥ pc order, This (.35) method is fine; but if one doesn't then Lsrch is a v. g. way. → (60.01.)

Unclear as to how good my stoch. lungs will be in giving "pc order"!

.01 : 59.40: So 59.19 is perhaps the present TM2G. It does not involve "Maximization of Utility" Nor even wts for the various problems: But it does involve "wts" in choosing how much time to spend on various "parts" of GPD, during "improvement" of GPD.

Local optima

Also, there is the (perhaps) Bug of 59.26–.29 ("self conforming hypoth") (& assoc. diffty of 59.30–.34)

.03 .04 Using conventional Lsrch, rather than searching in exact ≈ pc order may get around the 2 difftys of 59.26–.34: I think this will work, both for OZ & "Anytime problems". .31 ~~the anytime problems~~

So, say the TM2G of 59.16ff is Adequate.

[Tho I'd like, also, to consider the TM2G that tried to Maximize the Utility of each type of problem, & maximize expected total Utility] → See 61.04

.08 My recent analysis of Opt. Seq. Srch (V1989) of 59.37 (& recent comments on OSS85) Suggests that Lsrch often does not test probs in ≈ $\frac{cc}{pc}$ order. This may help w. the Diffty of 59.26–.29 (see .03)

So If TM2G was more or less O.K. — then how does the rest of TM work?

.13 We start by solving problems, using an initial GPD. (a Reference Umc). Presumably this Umc has conc., instructions, in it so the initial probs are easy to solve. After a fair no. of probs are solved, we want to "improve GPD." Now each type of OZ problem can have its own way of doing Opzn. We can have special OT's to deal w. opzns for INV probs, for seq. predn probs, for opzn for induction, by induction — Perhaps ~~[illegible]~~ OT's that TM2 (the GPD) decides on, after looking to see what OZ problem is (or what part of the GPD the USR wants to improve)

The Big Q is: Could I actually get .13 ff running as I've described it? Initially, it would only be able to do very limited improvement of GPD, because it is very limited in the kinds of regys it can recognize. — But after it has learned to solve the kinds of problems whose solns are relevant to more complex regys of the GPD — then it can help on that area.

Meanwhile, I will try to "tailor" the set of OT's — so that TM can more readily devise new, better, OT's.

.27 SN A note on the TM2G of 59.16 ff: As stated it is $\prod_i p_i$, where p_i are the pc's of the O.T.'s or prob. soln. methods that got the "best soln.": We will probably want to change this to MAX $\sum w_i \ln p_i$, since certain probs are more important than others (to USR). ($\equiv$ Max $\prod p_i^{w_i}$) 61.04

.31 : 59.29 SN Actually, in Lsrch for OZ probs, one spends $p_i T$ on the i-th O.T. — so this does sound like a "Self Conforming Hypothesis" — tho one does spend some time on the other o.T.'s.

.33 Another way (theoretically better than Lsrch) is to try O.T.'s in p_i order, using some time T testing for each. Then do $T \leftarrow 2T$ etc. (This would be Anytime problem).

.33 sounds "Not so Bad", since other OT's get tested for full time T & can be legitimately compared to the one w. max p_i!

.37 Re: 59.16 opzn It does not take into account the Time needed to evaluate GPD_i.

.38 This is OK: If it takes time T_i to generate OT_i, then it has that much less time available for testing it. In Anytime, or in INV problems of large cc, the generation cost of the O.T. or prob solving method could become a very small part of the total cc for that cand. — Our overhead we can get → 61.01.

3/5/01 TM IDSIA

.01: 60.40 V.G. GPD values at tremendous cc, if we use full RLP for a very long time. — So cc. of any Prob'y estimator will always has to be considered.

.02 So: 2 contending "Solns." to TM2G:

1) t. s/n of 59.16 ff: w. noted 60.38 on how cc. info is implicitly included.

.04 2) The "Linearized Gore" method: T. final presented soln. to each problem is given a Gore; In t. case of OZ (a/o predn problems): what was "Utility" to USR of soln. for past soln.? In case of INV problems: how important was t. soln. to USR.? — How big was t. T needed to solve that problem — Mixed w. how impt. t. problem (soln) was to USR.

Perhaps Relevant to .02: Given GPD$_{12}$: TM wants to "Improve it": It's not clear from t. TM2G soln. used, as to just what parts of t. GPD should be worked on. Say GPD gets a low "score" for work on a particular problem. It is not clear that simply because of that low score, that this would be a good area for TM to try to "improve". Decisions of this kind are/partly induction problems.

.14: 3/8/01 It seems from Juergen 7.01 (3/8/01) ff, that in Lsrch, using a stack lang. of PSG-type — (not "Z141 type") will not make Lsrch diffct., or modify it signif'ly.

A possibly big problem would be "lrng" during Lsrch: Actually changing t. GPD during t. Lsrch would seem to not be difflt. — Hvr. Say one is doing $T \leftarrow 2T$ Lsrch — one is ½ way thru a "round" of $T \leftarrow 2T$: One has rejected certain trials because of $cc_i > T pc_i$. If some of those pc_i's would ↑ much because of t. Change in GPD, it would not affect t. rest of that round. The pc_i's in the forthcoming ½ of t. round that were changed, will modify t. Lsrch results.

Its hard to see clearly what t. effect of GPD modifn. will have on t. Lsrch: If what looks like a very serious modifn. occurs, one could Abort t. "Round" & restart t. "Round"

Note that we can Normally do TM2 stuff & TM1 stuff "Simultly" by "Time Share".

.26 T. foregg. may not be too important! At significantly advanced level of trng., TM will have devised new prob-solving methods that include search w. lrng. during t. srch.

Re: .14 ff & .26: "It all Info is in t. PD" This, ideally would mean that any techniques for "learning betw. trials" would be incorporated into t. system as special srch. techniques for particular kinds of problems. If it turned out that there was a better way than Lsrch that works for all problems, then TM would always use that method for all problems: T. GPD would give it a $pc \approx 1$. — T. System would be theoretically "Lsrch" but, practically, not Lsrch at all.

So a Big Q is: "To what Extent can all info be put into t. P.D."? There are t. 3½ standard Arguments: (21.01)

① If a faster/better method is known, we should be able to put them into t. P.D.

② (Almost) all/some heuristics change operate by chnging order of trials. — It would seem that this could be done via chnging PD & using Lsrch. — trial order in Lsrch is "≈" $\frac{cc_i}{pc_i}$

Find ways to this!

Relates Most general

③ To insert a Heuristic PD: devise a set of examples that Heur. could have been "lrnd from": Then incorporate "t. inductive implications of this data/corpus, into t. GPD (This Doesn't seem to include things obtained by "Logical Reasoning")

(4) "t. FN. of t. 589: This is t. idea that anything that TM does in t. past (& "traces") is acceptable data for TM2.

3/8/01 IDSIA Juergen Jiaoyan (.19) MAYBE IMPORTANT; LEARNING DURING LSRCH. 62

.01: Some impt. pts: ① T. Review of 51.25 ff is not bad!

.01 ● ② T. Idea of 54.37 on How to write a TSQ is perhaps very good!

.02 ——— A More Detaild Analysis of flo Chart of TM: Steady State.

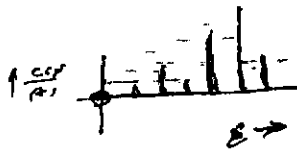
1) TM solves a new problem: It takes time ≈ T_0

2) TM spends ≈ time ≈ T_0 on Updating/improving GPD. Final GPDs can be in various forms: [Input is always problem: (?)] Any alternatives? Output can be ① listing of cands in PC order, w. assoc PC's, so that all possl. cands are include ② listing not nec'ly in PC order, but high pc cands tend to be near beginning of list.

.09 Usually (if not always) one can mark a pt. in t. list s.t. say: there are no pc's > a certain value $\widetilde{PC}$, after that point. ③ In both ① & ② (For Tree Lsrch w/o Long Term Lsrch) each cand has a unique identifier, and its own memory place (home/frame) at which we can put properties (slots). A few slots give ⓐ PC of cand, ⓑ amount of time worked on this far, ⓒ order no. on list ⓓ perhaps other unique identifying string or no.

In the T.S. mode of Lsrch, we go thru the list of cands & we spend an amt. of time working on it ∝ its pc. At a certain pt in t. list, we will stop. A better way: At each round of updating, we bring total amt of time spent on a cand up to a new hyper threshold. Thus for each threshold, there will be a pt in t. list of cands ~~at which~~ after which no update work is possl. (i.e. amt to be done is < 1 unit). We then start new round w. hyper threshold, T. Individual thresholds are $pc_i \cdot T$.

● (.19) ~~At~~ In this system, unless we have "learning" (≡ Modifn. of GPDs) during this time shared search of cands, This "learning" will change pc's of cands. We want to change ordering of cands (to some extent) of PC. Just how this effects t. Lsrch is unclear: It will depend much on how t. reordering/is manifest. One way, would After each "recalcn. of pc's", say we have t. cands in "ruff pc order" ("ruff" order meaning (.09) perhaps, but I'm not sure this is nec'y). Anyway, in T.S. mode of Lsrch: We now have this bunch of cands that have been worked on difrnt amounts, but using a difrnt. p.d. — so each cand has a new $\frac{cc_j}{pc_j}$; cc_j being amt. worked on up to now. ~~Before~~ t. "new learng" all $\frac{cc_j}{pc_j}$'s were about t. same. — Now they are difrnt: So ideally,



we work on t. cand w. smallest $\frac{cc_j}{pc_j}$ — until we have 2 cands w. same Le_j ($\equiv \frac{cc_j}{pc_j}$), then we time share equally betw. them, till we have 3 cands of $= Le_j$: we T. share betw them until we have 4 cands of $= Le_j$, etc. Its like pouring water into a tank w. rods of difrnt length th. coming up from Bottom of tank.

We don't work on a cand unless its pc_j is such that $\frac{1}{pc_j}$ (1 unit of work/time) would bring it up to current Threshold. i.e. $pc_j > \frac{1}{\text{current threshold}}$ (as current threshold ↑, we consider cands of lower pc_j)

It may be that the new pc_j distribution always conforms to (.09) so we eventually know when to quit searching for new cands to update w. cc_j's in view to t. new "Threshold".

● (.19) It may work! depends on the kind of output t. GPD gives, but at any rate, it does look like .19 ff does do cc at just those pts that are most promising, at the rate of their "promise". It may be possl. to prove that, in some sense, this is t. best way to do a TSQ! ALSO! Perhaps arrange so that cands can have access to "te" (fraction of time for "s.i." during their "candidacy")

Jueeng Jaiyan

3/8/01 IDSIA (.19) ~~MAKE~~ IMPORTANT: LEARNING DURING LSRCH.

.01: Some impt. pts: ① T. Review of 51.25 ff is not bad!

.01 ② T. Idea of 54.57 on How to write a TSQ is perhaps very good!

.02 —— A More Detaild Analysis of flo Chart of TM: Steady State.

~~[illegible]~~ 1) TM solves a new problem: It takes time ≈ T_0

2) TM spends ~~≈~~ time ≈ T_0 on Updating/improving GPD. Final GPD can be in various forms:

[Input is always prob|dern: (?)] Any alternatives? Output can be ① list of cands in PC order, w. assoc pc's, so that all poss'l. cands are include ② listing not nearly in PC order, but ~~higher~~ pc cands tend to be at start of list.

.09 Usually (if not always) one can mark a pt in the list & say that there are no pc's ≥ a certain value $\tilde{PC}$, after this point. ③ For T.S. Lsrch w/o long term Lsrch In both ① & ② each cand has a unique identifier, and its own memory place (home / frame) at which we can put properties (slots). A few slots: a) PC of cand, b) amount of time worked on it so far, c) order in the list d) perhaps other unique identifying string or no.

Any possy. of ORec useful properties of a cand?

In the T.S. mode of Lsrch, we go thru the list of cands & we spend an amt. of time working on it ∝ its pc. At a certain pt in the list, we will stop. A better way: At each round of updating, we bring total amt of time spent on a cand up to a new higher threshold. Thus for each threshold, there will be a pt in the list of cands ~~at which~~ after which no upda work is poss'l. (i.e. amt to be done is < 1 unit). We then start new round w. higher threshold, T.

Individual thresholds are $pc_i \cdot T$.

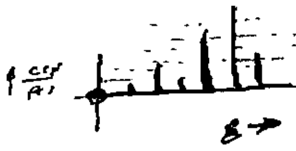
(.19) ~~At~~ In Lsrch T.S. mode, it's fine, unless we have "learning" (= Modification of GPD) during this time shared search of cands. This "learning" will change pc's of cands. ~~[illegible]~~ ~~[illegible]~~ We want to change ordering of cands (i.e. sequence of pc at ent).

Just how this effects the Lsrch is unclear: It will depend much on how the reordering is manifest!

One way, would ~~be~~ After each "recalculation of pc's", say we have the cands in "ruff pc order" ("ruff" order meaning (.09) perhaps, but I'm not sure this is right). Anyway, in T.S. mode of Lsrch:

We now have this bunch of cands that have been worked on different amounts, but using a different p.d. — so each cand has a new $\frac{cc_i}{pc_i}$; cc_i being amt. worked on up to now.

Before the "new learning" all $\frac{cc_i}{pc_i}$'s were about the same. — Now they are different: So ideally,



we work on the cands w. smallest $\frac{cc_i}{pc_i}$ — until we have 2 cands w. same i.e. (= $\frac{cc_i}{pc_i}$), then we time share between them, till we have 3 cands at = cc_i: we T. share betw them until we have 4 cands at = cc_i, etc. Its like ~~[illegible]~~ pouring water into a tank w. rods of different lengths coming up from Bottom of tank.

We don't work on a cand unless its pc_j is such that $\frac{1}{pc_j}$ (1 unit of time) would bring it up to current Threshold. i.e. $pc_j > \frac{1}{\text{current threshold}}$ (as current thresh'ld ↑, we consider cands of lower pc_j)

It may be that the new pc_x distribution always conforms to (.09) so we eventually know when to quit ~~some~~ searching for new cands to update w. cc_x; in view of the new "threshold" → 83.18!

.32

(.19) It may work! depends on The kind of output the GPD gives, but at any rate, it does look like it does do TS at just those pts that are most promising, at the rate ∝ to their "promise". It may be poss'l. to prove that, in some sense, this is the best way to do a TSQ!

ALSO! Perhaps arrange so that cands can have access to "t" (fraction of time for "T.S." during their "candidacy")

TM2G

.01:62:40 Next problem of import : TM2G

First consider option GPD₁ : GPD₁ is a PD on solution times (INV prob) or on PD on G values (G values) to obtain for various Time limits for various problems: A possi. Give, would ~~be~~ max prob by for cases that actually occurred (Maybe more wt. for more desirable solns?) . — But a high GPD, with hy pc for corpus but a very long time to compute result (e.g. RPL w/output T) could be practically of no value! Hvr. in .08 - .09 this long time to output" comes out of time available for opt'zn. Hvr, we do have a partial ordering wrt. CC & PC. For each input problem we have ~~a~~ cc₁ & pc₁ so that we could unify them into a single micro Gore, i.e. [illegible] some of its gores.

.06 Another possy. How to order ~~[illegible]~~ GPD₂ outputs in "Gore" order. (A linear Ordering).

Say GPD₂ results in ~~[illegible]~~ a certain set of soln. to a problem set (≡TSQ). ~~F~~ Consider OZ probs.

.05 OTᵢ works for ~~[illegible] "overload"~~ T - αᵢ and got Best out put βᵢ) Hvr. note .18 /

.09 OTⱼ " T - αⱼ " " " " βⱼ) ↳ But Note .24 !

(Since all OTᵢ's use same total time, T, one w. max β is "Best". We want GPD₂ to assign as large a pc as possl. to that "best" OT.

.13 For INV probs, only cand₂ will have solved the problem: we want GPD₂ to assign max pc to that cand₂.

For both INV & OZ probs, how ~~[illegible]~~ is GPD₂'s cc taken into account?

A GPD₂ that assigned same pc's to "winners" as a different GPD₂, but took much less time to Generate those cands, would be much better — but this is not reflected in the analysis of .06 — .13 . Well, if it took less time for generation, we'd have more time to work on both OZ & INV probs

.16 On the other hand, time spent ~~[illegible]~~ improving ~~[illegible]~~ GPD₂ is more effective to ~~[illegible]~~ TM's long term goals, than is simply working on a "production" problem ! — So in .08 & .09, we may not be doing it right. !

.21 Woops: .18 confuses Time to discover GPD vs. time to execute GPD. Right now I'm interested in Execution time. Discovery time is related to Horizon (i.e. .18)

Considering 2 Inv cands; cand₁ taking ~~[illegible]~~ longer time to generate & test correct soln. includes Gen. of ~~[illegible]~~ pc₁ 23)

T. total soln time is $\frac{cc_1}{pc_1} + \frac{cc_2}{pc_2}$. ccᵢ includes generation & testing. So we want to minimize this.

we may or may not want to weight each problem soln.

.26 for OZ problems, "Generate & test" always takes same time. So we want Σ Uᵢ to be max, where Uᵢ is utility of soln to iᵗʰ problem. obtained.

.28 Hm! Right now, it's not clear to me as to just how the OZ soln Works! ~~[illegible]~~ 59.35 deals w. this Q.

See Notes on (OSS 1985) : 4.15 ; & for time share version, 1.25 (ibid) :

.30 In the T.S. version we spend time ∝ pcⱼ T on the jᵗʰ OT. We slowly ↑ T until ~~[illegible]~~ pcⱼ T = T̂ for the best soln ~~[illegible]~~ (highest G) of all the OT's. Total time is < T, & $T = \frac{\hat{T}}{pc_j}$. So it takes $\frac{\hat{T}}{pc_j}$ time as long as the optimum

So, a GPD is Good if it assigns ~~[illegible]~~ hy pcⱼ to the OTⱼ that did Best.

In the T≈2T version of OZ soln: use each OT, pcⱼ T for ~~[illegible]~~ OTⱼ : for all OTⱼ's: After each T≈2T cycle. ~~[illegible]~~ look at time spent on best OT, when it is ≥ T̂ (time limit on problem), Stop. → 102.00

.35 Going back to the GPD evaluation problem for OZ probs! The Time pcⱼ·T includes time to generate the O.T. & test its pc? — so we simply want the OT that won to have as hy pc. as possl.

[First note in OZ probs! 3 ways to solve them : 1) try OT's in pc order for time T̂ each (Best way) ← if doable

2) Time shared , ~~[illegible]~~ ~~[illegible]~~ time limit unknown, but stop when T̂ has been spent on any OT.

3) T≈2T not much different from Time share:

When run it any a GPD is "good" if it assigns hy pc to OT that got best (indep of which way one uses it. [illegible])

TM2G (final soln?) (.01–.04)

.01: 63.40: So for each OZ problem, one can ~~[illegible]~~ Tell which GPD is better than which other GPD.

The time for optimum is $\propto \frac{1}{pc_i}$; ||| for Env. probs, the Time to soln is $\approx \frac{cc_i}{pc_i}$.

for All Env probs, total soln time is $\sum \frac{cc_i}{pc_i}$: for OZ probs, total soln time ~~[illegible]~~ is $\sum \frac{T_i}{pc_i}$

.04 Minimize the sum $\frac{\sum cc_i}{pc_i} + \sum \frac{T_i}{pc_i}$?

T_i is Time limit for i^{th} OZ problem
pc_i is pc of OT giving best soln
→ 70.04

.05 3/15/01 SN To get a good ~~OZ~~ set of OT's for OZ problems: I expected to take a (large set of OT's that I know of, & "factor" them into a set of components that's adequate to express a set of OT's as a stoch. Lang. Then add a few msts, etc. (lang is not already "universal".) This is a kind of "Jump start" — otherwise it would take an Enormous amt of training to Get TM to do much "Self-improvement".

The forego. ~~[illegible]~~ of necessarily puts a lot of "Bias" into "Self improvement", but the "factors" — combining them makes system

→ factorization may help to overcome that BIAS. more usefully "Universal".

.20 It is clear that the forego. could be done ~~[illegible]~~ — in any area or TM's Lang!

It amounts to [redacted] my designing a kind of "Expert Systems" but much Better than Normal expert Systems, since it really "Lrns" in a near optimum way.

Of course there's the BIAS Problem.. — But I'm not sure it's really any worse than/the Bias ~~[illegible]~~ implied by using any TSQ & the UMC.

.25 One Vague fear I have about much "Hand Feeding" of TM is little details that TM needs to Lrn things by itself. I ~~[illegible]~~ would much rather have TM learn a Domain than have me f & Plan it, etc., because then I know TM has the means to learn that task. ~~If~~ this is not the Q of "little details" here. —

→ In .20, the system would not nearly be able to (ma any of the Prob. Solving techniques it knows "by hand".

The "little details" of .25 is a worry when I insert/the Sections of the conc. next. but I think adequate for TM to [redacted] "discover" a soln. to a problem.

3/17/01 IDSIA

GAMB House Item 1 : LSRCH

Lsrch not really near "optimum" in the sense of GHTI

$\Sigma\, cc_i$ ct

Norm all pdfs even the one w. C

.00: On / true $\frac{cc_i}{pc_i}$ order v.s. T.S. Lsrch = $\frac{cc_i}{pc_i}$. $\frac{cc_i}{pc_i} < \left(\frac{cc_j}{pc_j}\right)$ soln. if $i<j$.

$\frac{1}{pc_i} < \frac{1}{n}$; $\frac{cc_i}{pc_i} \geq \frac{cc_{i+1}}{pc_{i+1}}$ | we want to know $\sum_{i=1}^{J} cc_i$ v.s. $\frac{cc_j}{pc_j}$ → See 72.01 for outline of proof.

.03 When Lsrch terminates in TS mode; It has worked on all cands w. $\frac{cc_i}{pc_i} < \frac{cc_j}{pc_j} \equiv Lc_j$

This would be the amt. that GTH works on ($\equiv$ optimum). In addition, it works on all other cands

w. $pc_i > pc_j$: $\frac{cc_i}{pc_i} > \frac{cc_j}{pc_j} \equiv Lc_j = T$ Amt of work done on them

is $\sum pc_i \cdot T$ $\sum pc_i < 1$ so work done is $< T$ and $\boxed{\frac{cc_j}{pc_j} < 2 \text{ times}}$

.10 optimum $\equiv \sum cc_i$ that would occur if the $\frac{cc_i}{pc_i}$ were known in advance & done in that order.

The factor is $\underline{\underline{<2}}$ by the subtraction amt $\left(\sum_{j+1}^{\infty} pc_i + \sum_{\left(i \mid \frac{cc_i}{pc_i} > T\right)}^{j} pc_i\right)$ could be larger. could be larger

So the factor could be not far from 1.

pc_i are in pc order large pc_i first

or the factor is $1 + \sum_{i=1}^{j} pc_i$

optimum $\left(cc_0 = \sum_{\left(i \mid Lc_i \leq Lc_j\right)} cc_i\right)$

LS Lsrch $cc_0 + \sum_{\left(i \mid Lc_i > Lc_j\right)} T pc_i \geq Lc_j \equiv T$ $\boxed{Lc_j \equiv T}$

.20

$\left(\Sigma \,\middle|\, cc_0 + T\left(\sum_{\left(i \mid Lc_i > Lc_j\right)} pc_i\right)\right) \stackrel{?}{<} T$

Consider extreme case; cc_i for $\left(i \,\middle|\, \frac{cc_i}{pc_i} < \frac{cc_j}{pc_j}\right)$ are all very small. Say they are all .1 ; $cc_j = 10$, $pc_j = \frac{1}{101}$

pc_i are all $\frac{1}{20}$ so $\frac{cc_i}{pc_i} = 2$ $\frac{cc_j}{pc_j} = \frac{10}{.5} = 20$

e.g. .20 suggests that ratio of cc_0 to T (.. might be anything! i.e. that (cc_0 could be $\ll T$) !?

Proof that Lsrch can be arbly worse than doing trials in Lcost order.

.30 $\sum_{\left(i \mid \frac{cc_i}{pc_i} \leq T\right)} cc_i = \left(i \mid cc_i \leq T pc_i\right)$ so $\sum cc_i < T \sum pc_i$

$\left(\sum_{i \mid Lc_i \leq T} cc_i \leq T \sum pc_i\right.$ Even if pc_i sign were "=") compare to $\left(T \sum_{i \mid Lc_i > T} pc_i\right)$ cc_i would seem to be arbly <

$\therefore$ compare $\sum_{i \mid Lc_i \leq T} pc_i$ w $\sum_{i \mid Lc_i > T} pc_i$

so $\sum_{i \mid Lc_i \leq T} pc_i$ v.s. $\sum_{i \mid Lc_i > T} pc_i$

(N.B) Even if Lsrch is not "near" "optimum" (in the sense of GHTI) It still may be optimum or near optimum for initially unknown cc_i's of cands. Still, this could use same Analysis. → See 68.00

We can divide the total of all pc_i into 2 sets so that $\sum pc_i$ of each set has (Arby ratio) (arby ratio) to the other. Then assign cc_i's so that they are consistent w. 2 sets of inequalities.

So Lsrch can be worse than "pure Lcost order" by arbly large factor: (This is really not bad) → See 68.00

3/18/01 IDSIA

.00 : ~~Recommendn~~ It may be possl. to use a rather simple Global flowchart for TM, i.e.

.01 depend on greatly sugrd Cands (that in themselves can include novel such routines — also they can (during their own soln times) ↑ t. fraction of time spent on (S.I.).
i.e. Input is problem dern: this goes to GPD, which outputs a set of soln. trials w. their pcs.
A fair ~~[illegible]~~ fraction of Global cc is usd for Improving t. GPD.

ⓐ Only pure Lsrch usd (no learning betw trials, possibly no lrng betw T ← kT)

(ⓑ "Improving t. GPD" does not include much (if any) cc spent on t. OZ problem of improving t. GPD.
i.e. the system need not be really recursive ~~[illegible]~~ in this particular way (But see .11–.16 → This may not reduce complexity of t. system — also will it affect)

We depend on a ~~[illegible]~~ fairly good set of initial OT's for OZ problems to get t. job done.

.10

.11 ⇨ But what about ENV probs.? Well, they are ~~[illegible]~~ dealt w. by a gradually improving GPD.
The OT's to improve t. GPD are fixed "at Birth". (It does seem strange that t. OT's usd for "~~[illegible]~~ GPD improvement" should be constant, but those usd for "non GPD OZ problems" should be allowed to improve/grow!) It may well be that having t. GPD assign (probabilities), OT's to be usd on ~~[illegible]~~ in improving itself involves no logical diffys! Superficially

.16 it would seem that there should be "no problem" (Kein PROBLEM!) ⓒ The (cand$_i$, pc$_i$)'s are t. output of t. GPD when its input is t. sum of itself ~~[illegible]~~ a T limit. This is a "discrete

.20 "Δtime" f.b system. I don't see how it would oscillate or "misbehave" much!

.22 Other than ① TSQ design, A Big problem may be ② IMZG: It is t. overall System Core.

.23 ③ Another Q. that I am quizzy about is T. cands' control of K (fractional time usd for S.I.)
Probly a better way would be to allow TM to ~~[illegible]~~ do S.I. for awhile, then turn S.I. off then do ~~[illegible]~~ present problem for a while, etc. — periodically doing S.I. in "pulses" so as to take advantage of recent Cand tracing ("Traces").
So initially, k is set at .5, but a cand can change k or use its own ~~[illegible]~~ switching algm betw. "S.I."s "~~[illegible]~~ present problem" (Actually, "time ~~[illegible]~~ share" is a special case of such Algms)

.30 So, it will have to learn to control switching betw. S.I. & ~~[illegible]~~ S.I. (non-S.I.).
I think it is learnable in "reasonable time".

④ Another Q: How does it do "logical reasoning"? ~~[illegible]~~ Prelimy agts:
1) Logical reasoning is ~~[illegible]~~ ≅ equivt to inductive reasoning betw. pc ≈ 0 or 1 cc.
2) "Teach" TM to do Logical problems "External to itself". Then it can solve its internal logical problems "by Analogy" w. its operations usd to solve "external" logical problems.
I haven't worked out details yet, so I don't really know if it will work.

.32 → ⑤ Q: What about TM2's induction) → How Good is Z141 Model?
I don't immediately see how it would be applied to t. SAAB ANZ problem!!
At first t. GPD is not a "Conditional" but an "Unconditional" GPD.
Perhaps .37 is t. most difflt problem of them all!

→ See 76.22 for main remaining Diffys in TM.

→ 67.15

3/18/01 [redacted]

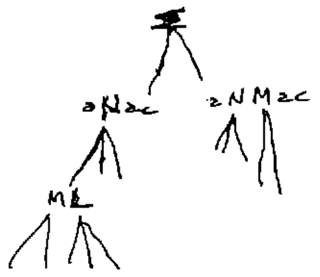
5:45P

.00: : STOCHASTIC Langs: In a Stoch CFG: The generator of the ass may be regarded as a stochastic finite state machine: Each N.T. ~~type~~ type corresponds to a "state".

"Say we have $N_1 \to$ ABM abM ..

Better still, consider t. Generating TREE! We just trace out the choices needed for all of the NT's in a simple exhaustive way. The successive terminals that we jump to, ~~and~~ correspond to successive state of t. "FSM".

.06



.10

This Doesn't Seem to Work! In a [redacted] FSG, the state one is in at any particular time, completely characterizes t. future of t. system. This is not true in t. graph of (.00–.10)

E.g. on t. left edge, at "M", one jumps next to "L". On t. Rt edge "M" completes t. ~~ass~~ ass.

.15: 66.40: Consider the TSQ: $x_1\ x_2\ x_3\ x_4\ x_5$
3+5=8 (many cases) then 4+9=?

(1) The model x_1, x_3, add → x_5 works.

(2) next 3−5=−2 (many cases) then 4−9=?

.20 " " x_1 x_3 sub → x_5 works.

next 3×5=15 (many cases) then 4×9=?

Or 3 ± 5 = 8,2 (1, mixed) then 4 ± 9

.23 There is t. Q of dividing up t. "induction" betw. TM_1 & TM_2. → How much t. cand is able to do, v.s. how much "S.Z" does.

Basic Method of operation (for induction probs)! Input one problem w. corpus.

1) TM finds (using current GPD & LSrch) one or more codes for t. corpus.

2) Augment corpus w. new problem(s). Using fact that old code fit old corpus, t.P tries to find ~~new~~ codes for entire augmented corpus. This can be regarded as a special ~~problem~~ type of OZ problem, peculiar to Induction problems, w. "Specialist" OT(s), for working on it. (.35)

.30

SN on OSL: Just as MML / MDL / ~~GREAT~~ doesn't work at all for OSL, it works poorly for very small ssz.

Note in TSQ analysis: Be sure we are able to do one (& 2,2,3) shot lrng (small ssz).

.35: (30) Trouble is, while I seem to have a v.g. understanding of TM_1, TM_2 interaction on a theoretical level, I seem to get into terrible problems when I try to do a TSQ! Perhaps my part of it is (.23)

"The Devil is in t. Details" As long as I stay at a hy level of math analysis (or "English") things are o.k. — But as soon as I start on t. detaild system — It doesn't go together well, at all! → 69.00 Spec

→ 75.16 ff may solve this well enough.

3/19/01 IDSIA

T.S. Lsrch: "Optimality": Heur Argts: Also for Lrng. durng Lsrch

.00: ($\frac{\text{Spac}}{65.40}$): Are "Heur" argts to show T.S. (11) Lsrch is Best. Also if Lrng occurs durng Lsrch (i.e. Modifn of GPD) then t. best "Greedy" algm. is ..62.19A;

T. argt: At any time, t. best "prob'ty of success/unit cc expended" is t. candvr. ~~[illegible]~~ hyest ~~[illegible]~~, $\frac{pc_i}{cc_i}$. In t.s. mode, one doesn't know t. $\frac{pc_i}{cc_i}$ of any of t. cands (other than those "tested to death" "completed"). One has, instead ~~[illegible]~~ an upper bnd on each $\frac{pc_i}{cc_i}$,

(for cands not yet worked on, it is from $\frac{pc_i}{\tau_{min}}$ (τ_{min} is ~~[illegible]~~ min time worked on a cand: = time of 1 trial + time to switch cands)

.10 One then works on cand w. ~~best~~ ≡ hyest ~~[illegible]~~ upper bnd (on $\frac{pc_i}{cc_i}$) of all cands.

As one works (unsuccessfully) on a cand, its $\frac{pc_i}{cc_i}$ upper bnd ↓ until it = that of

ξ another cand, — so one then time shares ~~both~~ those 2 cands.

This argt. would also justify 62.19, If we want ~~[illegible]~~ to work on the "cand w. most promise" at any time. This is t. "Most Greedy" approach.

.20 If (in 62.19 in particular) we have any a priori info about "correlations" betw. cands, then we mi. be able to be less "Greedy".

Coca Cancer!

.30

3/19/01 IDSIA

Bohn, Gabrielle
Madison, Wisconsin

2PM or 230

TSQ

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

.00: 67.40 spec. So I'm MUCH better at "Top Down" Analysis!

.01 Perhaps look at some of my recent "Bottom up" analysis of ANL: try to see just what I was doing

.02 that was so (diff'lt/confusing/counter/productive)

Remember! Each action is either TM₁ or TM₂.

→ I.e. .01–.02: Note each (instance/kind of (induction/regularity) used/considered) — usually it will be in TM₁ — but sometimes in TM₂, & sometimes, it should occur later in TM₁ (or TM₂).

Also, for each regy: There's the Q of whether I want to use it "now" or wait until later —

.09 i.e. "now" may mean insert it as a unit into TM — "later" means acquisition via a later part of the TSQ.

In this prelim Analysis: Concentrate on detection of these Regys.

T. Sequence
T. Sequence/set of regys, & not on t. method of
T. Sequence/set of regys, & not on t. method of ← □

A cand (for induction problems) need not be (initially) in t. form of a (conventional) computer pgm. It can be a list of "regy" detectors — These must be equiv. (in some sense) to compression algms. Each Set of "regys" will have a pc. — At first (Default) This will be t. product of t. pc's of t. individual regys.

.18 Hvr., another way to look at this: If there are several regys in a corpus, one can use each of them to get a sequence of compressions — evaluated sequentially. "Coding and Recoding" ©.

→ So: for a tsq. — write down, for each problem, a set of regys that would be adequate to solve it. Whether I want these regys to be primitive, or have some of them be "lrnd", can be considered later.

In t. case of induction problems: T. cands will be means of finding regys: So an "elementary" cand. would be a ~~simple~~ "single" regy. A more complex cand could embody several regys. An more complex cand would examine corpus & decide which regys to try. (This is essentially what t. GPD does! —

.27 but t. cand is less restricted in how it can decide what regys to try: It need not use pcs ≠ a/o Lsrch.

A cand can (probably) do a "call" to "System as a whole", to give that system a problem to solve — Or, it may "call" t. GPD w. a problem descn input to it, & use t. output of t. GPD in a novel (not nec'ly Lsrch) way.

The "GPD" must have a way (after looking at t. input problem), to devise a system for creating a Stochastic lang → whose outputs are cands.

3/21/01 (Yet!) Another way to write tsqs: first list a seq of problems w. rather large CJS's but then: Run insert problems to ↓ cjs's.

.36

3/21/01 IDSIA.

TM2G 70.^{34}-.40 : 71.01ff   OSL diffy .06   "The factor of 4" (.13)

.00 : 70.^{90} needed to Calculate pcr.

Hvr note that time needed to calculate pcr is not related or "same as" time spent on S.I. (i.e. "improving GPD).

So: Th. [redacted] TM2G of 69.01-.04 may be theoretically O.K. — but it may sometimes take long to calculate exactly.

.03 Another point Re: T. optzn of TM2G: While we want to ~~min~~ minimize $\Sigma \frac{cc_i}{pc_i} + \Sigma \frac{T_i}{pc_i}$ we are constrained in how we can do it. The pc's have to be obtained by legit. induction : ALP criterion. — (.09)

.05 If we use ~~normal~~ ALP in t. normal way, we have to add in special provisions for OSL.

.06 Re: OSL: If t. "2 part coding" of MDL (2 approximation to APL) don't deal w. OSL because ssz = 1", This same Technique should be way off for ssz = 1, 2, 3 ⋯ I Really Have to Analyse This!

Note that my Analysis of "ZM" Works OK. for OSL.

.09 : (.05) So we may have to look for "novel" repys in our GPD corpus that are capable of being calculated quickly. This is Something I don't remember having ever thot about!

.12 Also, of course, ~~we~~ we look for ways to speedup GPD's evals. of pci's. — (74.00)

.13 On "t. factor of 4" : Actually a "factor of 2"

The factor of 2" is due to t. uncertainty of t "k parameter." ∴ The "factor of 2" is correct if we mean that t. system operating ~~at a rate~~ with a certain clockspeed is

.18 at least as good as any system operating at ~~one~~ ½ that clock speed. However, a factor of 2 in clock speed can modify ~~the~~ problem solving capability of any system by a very large amount.

.21 A way to look at it: ∴ My system using ½ clock ~~at~~ full rate: 2 /sub systems in ∥: one doing production probs full time on one, S.I. full time on other is at least as good as any system doing full time on one only. At any time, my system has spend at least as much time on S.I. as t. optimum

.23 system. also at least as much ~~time~~ time on production probs as t. optimum system.

.24 "Factor of 2" here, is used in t. same sense as "a constant factor" used in deriving t. efficiency of Lsrch. T. results of ~~an~~ a truly optimum/system can be enormously better (say exponential in size of problem) better than Lsrch — - - in terms of t. Gene for t. OZ problem

Argts re: .21 Against : While total time ~~.~~ TM spends on S.I. ≥ that spent by optimum system,

a) The time may be distributed difrntly w.r.t non-S.I. problems. (This argt. is weak, hvr, because of .23-.24)

b) The time may be " " " " This difference can be in difrnt aspects of S.I.

.33 In General, S.I. can be localized to certain problems or problem areas, or more Global. The Global credit

.34 assignment function would seem to deal w. this, but I'm not ~~at~~ at all certain of it.

So .33-.34 ~~can~~ may be strongest argt. V.S. t. "factor of 2".

.36 An Argt. for t. factor of 2 : If t. searches in TM get very long (per problem) t. amt. of time spent on S.I. also ↑. I think t. effect is to effectively spend "more/percentage time" on S.I., because a long search does not nec'ly mean more complex calcn. during t. ~~on~~ non-S.I. problems.

(spec 89.08)

3/23/01 IDSIA

145
70

BIAS IN TM & in Computing PC RATIOS .00 See 72.00 on Bias.

.00 : A Possl. Source of serious Bias in S.I. algm: Say Cand1 & Cand2 are 2 cands that have been evaluated as a Production OZ problem Solns. Cand1 has Gnrr ~~[illegible]~~ G1. (say γ=1/2). G1 is > (better than) G2. We want TM to assign a hyper PC to Cand1 than Cand2 in GPD2. TM does this by first computing GPD1, t. pd of various G outputs for Cand1 for Time T & ~~[illegible]~~ other times, if we use T→2T (See 72.34 - .36) ~~or~~ T.S.

We want TM to be completely "unbiased" in computing these 2 pd's — because There is much "Value" to TM if it (biasedly) ends up w. Cand1 having more pd "wieht" for small ε, than does Cand2.

.09 Since TM would normally be rewarded for such "Biased" behavior & TM can do any kind of behavior it likes, This would seem like a serious problem. Can we find ways to keep TM "unbiased"?

.11 Can we find ways for TM to treat Cand1 & its data t. "same way" indep of 2?

I think I ran into t. fdmtl. problem in computing t. ratio of 2 pc's — which is t. main problem in Inductive Inference.

◎ Can we find ways to restrict TM's behavior — its methods of solving induction problem — so that we achieve .11, but we also allow TM to be "adequately Universal".

This can be very serious when we have a long list of cands & are evaluating their pc's. — We must (presumably) spend as much time on t. low pc cands as t. hy pc cands (?)

.18 Hvr, may be not so! Say we have a "Summarizing Machine" (no w. OCL). It is easy to generate PC's in an unbiased way by ~~[illegible]~~ random codes or exhaustive search over a adequate ~~[illegible]~~ size of space. For CB=∞, There is no Bias (other than t. Reference mc) — So, perhaps as CB↑, t. bias problem ↓!

Gen. Conclusions about this BIAS: That it probably can be a problem; .18 may be an approach; It may be that for many stoch langs, bias is not a problem. (Hvr if There can be many parsings of a string, one could be biased in omitting many. T. "restriction" idea of .11 is attractive; look at ways one can be biased — "gonz thom & "outlaw Mary"!

→ 75.15, .35, on BIAS
77.00

$\boxed{\text{TM2G}}$ (.00 ff)

.00: 71.12 (Spec)! While 71.03 is t. TM2G, There are 2 aspects of it! One is getting a PD to Maximize t. pc of t. corpus: so $\prod pc_i$ = Max. Another is to Modify t. calcns. of t. GDP$_2$ so that the pc_i take

.02 as little time as possl. to calculate. The combination of these 2 effects is t. Goal of 71.03. $\sum \frac{cc_i}{pc_i} + \sum \frac{T_i}{pc_i}$. If Most of t. work is in ~~[illegible]~~ paraInduction, Then $\prod pc_i$ is t. main Goal. Only when we consider very slow compns. for t. pc_i, does t. ~~[illegible]~~ final Goal of 71.03 become impt.

Are there 2 conflicting Goals? ! On one hand, we want an honest pd. (Is this max $\prod pc_i$? or is it more related to t. corpus of **GPD$_1$**?) Anyway On t. other hand, we want min $\sum \frac{cc_i}{pc_i} + \sum \frac{T_i}{pc_i}$, which seems to be t. "Top Goal!"

At present, I am quite confused about this: The $\prod pc_i$ = max goal is for the GPD$_1$ corpus. The $\sum \frac{cc_i}{pc_i} + \sum \frac{T_i}{pc_i}$ is for a difnt set of pc's, derived from ~~these~~ pc_i's.

Try this: Generate GPD & GPD$_2$: Then, don't charge any **time cost for** Generating of PC_i of **GPD$_2$**: This cost is part of S.i. budget.

Try this: M_j is a Pgm that Generates GPD$_1$ (via ~~[illegible]~~ RLP) and a set of $\overset{j}{pc_i}$'s for GPD$_2$, in time T_j. Which of t. pgms M_j is "Best"?

Remembering t. time cost of "Generating pc_i" (to Learn) is time before plugging in t. problem dern, & generating that (pc_i ~~[illegible]~~, cand) pairs — More exactly, whenever we plug t. problem dern in, we

.17 get a seq. of (pc_i, cand) pairs, & each takes a certain amt. of time to generate.

.18 ~~[illegible]~~ If t. time to generate in .17 is about t. same for all pairs, Then, for any/or problem, ~~[illegible]~~ in ~~[illegible]~~ all cands ~~[illegible]~~ have same time left to Solve t. problem. Not so for INV problems, hvr.

We spend some S.i. time T^1 to generate GPD$_1$. To Go from there to GPD$_2$ and t. output of .17 takes a Certain amt. of time T^2: T^2 is ~~[illegible]~~ t. time up to t. pt where t. (problem) dern is inserted into GPD$_2$. is all S.i. time.

Suppose we spend time T^1 to go from data to GPD$_1$. Presumably, GPD$_1$ is in a standard form that can generate GPD$_2$ ~~[illegible]~~ and ~~[illegible]~~ finally after (post) plugging ~~[illegible]~~ t. problem dern in to GPD$_2$ and getting .17

.25 Then <u>TM2~~B~~</u> is mainly to optimize t. Goal of GPD$_1$. We can improve things further by modifying pgms to take GPD$_2$ to ~~GPD$_3$~~ to .17; (using ~~[illegible]~~ 71.03 as ~~[illegible]~~ Goal) but this will not ordinarily be done, or will it make much difrnce in TM effcy.

.27 If ~~TM2G~~ for GPD$_1$ is $\prod_i pc_i$ we can weight various problems by $\prod pc_i^{\gamma_i}$ ($\gamma_i \equiv$ wts.)

→ A Pgm. that reduces time for GPD$_2$ to .17 (via input problem dern), would perhaps have Goal 71.03 (= .02)

.30 <u>Write up</u> ~~[illegible]~~ t. <u>ideas & argts of .25 ff.</u> This ~~[illegible]~~ solves an impt. problem, & I want to

.31 remember <u>how</u> & <u>Why</u> → (75.00)

.32 → A "hard Q": Re: t. theory, TM2G! Say we have a GPD$_1^1$ & we have a small amt. of new data. We work a bit on it/and get GPD$_1^2$ to incorporate it into t. corpus. → This problem occurs whenever we augment t. corpus! How can we compare GPD$_1^1$ & GPD$_1^2$? —

Which should we use on a new problem? — They are based on slightly difnt corpi. GPD$_1^2$ may not have incorporated t. new data well. hvr, t. "new problem" ... T. new data may not be relevant to it!

.36

3/24/01 IDSIA

TM2G 3.00   Bias .13, .35 ; 77.01

(Spec)
(.00) 74.31 : So we had 2 competing Gorcs: T. earier (more Global one was | Min $\sum \frac{cc_i}{pc_i} + \sum \frac{T_i}{pc_i}$)
pci's are for GPD2

(.02) More recent idea was Gorc ≡ $\prod pc_i$ = Max ; pci's from GPD1 (raw data).

In .02, It is assumed that t. output of GPD1 is in some standard form, so it can be converted to GPD2 ~~w~~ w. not ~~so~~ much cc. (T. mechanics of integration was discussed in [redacted] MCT analysis ~ 1997) —
I may have had some good ideas on how to do it w. relatively small cc.).

The ~~new~~ Gorc of .00 second more "global" I didn't have a clear idea as to what it meant. ~~What~~ ∈ An "obvious soln." would be $pc_i = 1$ for all of t. problem solns. What are t. constraints on this ~~new~~ Gorc.?
Well, in .02 we have t. same obvious soln. ($pc_i = 1$ for all problem solns. — Hvr, t. constraints are that t. only thing we can do to vary the pc_i is to find more codes for t. corpus. — and that we could use same constraint for t. Gorc of .02.

BIAS → (77.01 on "BIAS")

.13 Hvr, if we really want to do best on either (.00 or .02) Gorc, we will Bias our search so that codes for non-solns. occur less often (or not at all!) — Giving t. "desired"
.15 $pc_i = 1$ for all problem solns. (That this will occur is improbable / unlikely)!

→ (76.00)

.16 SN In sequencial predn: Say C is t. corpus to be continued. We want to maximize ~~our~~ our knowledge of t. lower bnd of $\sum 2^{-|p_i|}$, where p_i is a code for a string
.18 of which C is a prefix: I.e. codes for ~~strn~~ C or "C extensions". ~~[illegible]~~
The restriction is on total cc allowed to do this. If our goal is (is) to maximize $\sum 2^{-|p_i|}$ in t. available time (Just how could this introduce bias ?

In Bag predn! Consider folg. model of BAG predn! Umc. w. 2 inputs, 1 work tape, ~~#~~ 1 output tape! Put ~~random~~ random input into ①. When machine stops, put random input into ②, ~~whose probability that output is~~ If output is member of corpus, then again reset ② & put in another random input: T. pd of this second output is t. desired / extrapolated pd. on finite strings. I'm not sure design is correct. Somehow, t. defn. & implementation should n't be much more diflt than Sequencial predn.

Another try for BAG predn. (That makes it look more like Sequencial Predn).
Same 2 input machine as before; we want to find in time ≤ CB codes for input 1 and codes for input 2 that generate t. entire ~~corpus~~ Corpus. Say l_i^1 is t. t. ith possl. input to ① that causes it to stop (Inputs to 1 have no output, ever). $l_{i,j}^2$ is a code for t. jth corpus member, after l_i^1 input has stopped. In time ≤ CB we want to find codes l_i^1 & associated codes $l_{i,j}^2$

.33 ∋ $\sum_i 2^{-(l_i^1 + \sum_{j=1}^{n} l_{i,j}^2)}$ [$\sum$ n is total no. of ~~[illegible]~~ elements in BAG (= corpus) & If an element in t. Bag occurs r times, it is given r times in t. sum $\sum_{j=1}^{n} l_{i,j}^2$] ~~[illegible]~~

.35 After we get this sum to be as large as possl. in time CB (Being as "unbiased" as possl — → (77.01 on Bias)
.36 whatever that means!), we have a wtd set of "machines", ~~[illegible]~~ l_i^1, that we can use for predn.
(I'm not sure of .33) but t. wt. for t. ith machine is $\prod_{j=1}^{n} \left(\sum_k 2^{-l_{i,j,k}^2}\right)$ k sums over all codes
via machine i, for the jth corpus member. If a corpus member occurs r times, we take the factor, r times (≡ rth power).

3/24/01 IDSIA

3
4
6
7

TM2G .00–.03 ← (conclusions) ← (.04–.17) discn of

.00 (spec 75.15): In view of 75.16–.40: it appears (not 100% sure) that ~~we best ...~~ we can extremize ~~either~~ Goal 76.00 or 76.02 by finding many codes in t. available time (≡ CB). Doing Goal 76.00 seems better, because that's what we're trying to do, but I suspect 76.02 maybe
.03 easier to ~~[illegible]~~ implement & may be about equivalent.

.04 Discn of (.00–.03) ~~[illegible]~~ AH! The reason we can't optimize 76.00 or .02 by setting all pcᵢ to be 1. ~~[illegible]~~ ~~If we included only codes for t. desired prob. corpus, we want to not maximize~~ Eq. in 76.02 (Πpc_i = Max) we have to include (in INV problems) all trials and how ~~long they were &~~ much time they took & whether they succeeded or not. This is t. BAG corpus of 75.33–.40. If we only included codes for t. desired cand., T. codes for other cands would be zero ~~[illegible]~~ & t. product part of 75.33 would be zero. Each product 75.53 has to include all data (not just t. desired data). So, my impression is that for INV problems, pcᵢ = 1 for all i is not possl., & that we have to be more unbiased. That probably t. same thing is true for OZ probs, & that probably t. same thing is true for t. Goal of 75.00 ($\Sigma \frac{cc_i}{pc_i} + \Sigma \frac{\tau_i}{pc_i}$). So (.00–.03) may be correct.) — But I really
.17 (have to go over this in detail to see that (.00–.03) really are O.K.

.18 It is likely that we will use Gross approxns. to either of t. 2 goals 76(.02/.00) — but it's important to know just what we are approximating!

.20 OK: so say we understand TMZG (as .00–.03): What are main diffys in TM?

Before t. TSQ itself: Some practical Q's:

1) T. lang(s) used to represent probs & solns. : (probly ~ to Lisp a/o Forth, APL, Maryland lang).

2) Initial langs (≡ PD's) used to represent lists of possl. solns ~~&~~ soln. techniques.

.25 3) Initial OTS in TM2 (66.37)

4) Initial OTS for other OZ problems.

5) ~~Factoring~~ of all OT ~~sets~~ a/o ~~all soln. tech~~ all sets of soln. techniques used in INV/OZ probs.

.28 (6) ~~[illegible]~~ (62.23)? Just when ⌈doing "Production Problems" to spend time on S.I. — & what aspects of S.I. to spend ~~[illegible]~~ time on, & when to spend that time. Perhaps usually spend time during "production problems" on S.I. that's normally in "Production" problem Domain. "Between problems" (if this is meaningful) spend more time on more global aspects of S.I.

(7) 74.32: ~~[illegible]~~ of S.I. / "Improving PD" This is common problem when Corpus Augmented

(8) 66.22 has (5) diffys! (2) T. cand. "control of k" turn on, turn off, deciding what aspect of S.I. to work on. (62.23 / 2.29)

→ (b) How to do Logical Reasoning. (c) (.25)

→ (9) T. problem of BIAS in induction & in TM2 73.00, 75.13–.15, .35–.46; 77.01.

(10) T. Argts of "factor of 4" / "2 → factor of 1" seem reasonable, yet a "factor of 1" means assigning a PC of ~1 to t. method that will ~~be~~ be fastest; for all problems! — Seems unreasonable! I think t. main Argt., is that TM's behavior is essentially optimum — that problems can't be solved (any / much) better!

.33 A MAIN Counter Argt! That ~~[illegible]~~ Expressing prob solving as GPD + Csrch is ~~an~~ perhaps not t. only way to solve problems. If it is, then TM's drawn up may be ~ optimum. If not t. above argt. isn't much Good! A perhaps critical weak pt.: ~~[illegible]~~ TM doesn't ever have "all info in PD" that Human has

.40 → 78.00

3/25/01 IDSIA

$\boxed{\text{BIAS (.00)}}$

Previous Refs: (75.13-.15, .35-.40) Also 73.00ff

.001 : BIAS : In induction: Least biased is rote exhaustive or Random (w.o. replacement)

→ Lsrch is unbiased, if the short version is used. TM2T version is un biased if oversearch is done after soln. is found for INV problems. I think ~ remarks are true for OZ probs, but this should be checked! one could miss cands of lower $\frac{cc}{pc}$ than the first one found!

→ Actually in TM2T version one should continue search since otherwise

The Lsrch is, indeed unbiased itself, we have bias in choice of Ref. Machine. In TM, we are continually changing ref machine & ∴ adding "Bias"!

In General, there seems to be tradeoff between bias & search speed. Koza's "Genetic Prog" may be very inefficient, but it gets more "Unbiased" results — that are very "Creative". Of course his initial choice of language (representation) is very impt., & gives much Bias. — But without it (using same standard representation for all problems) we'd have less bias, but an enormous slow down in speed of soln.

While normally, it may be true we always have this trade-off betw. Bias & Speed, it may still be, that there are ways to search efficiently at a given Bias level, or get min bias for a certain srch speed.

TM, using Lsrch, puts all of its bias into ~~[illegible]~~ · TM2's ~~current~~ latest choice of GPD (≡ the latest Reference Machine). This GPD represents all of TM's "knowledge of the world" as well as its "BIAS".

Bias has several aspects. (1) $\overset{\text{unaware}}{\text{unrecognized}}$ bias ≡ bias we don't know about.
(2) Bias we ~~known~~ Aware know about: subclasses (a) Desired
(b) Not desired.

We want (1) to be as small as possl. — But just what the difrnce betw 'Desired' & 'Undesired' Bias is Unclear.

Clearly GPD (our integrated BIAS) is something we must have: Otherwise Lsrch takes too long.

The main source of Feedback on the effectiveness of our "BIAS Management Techniques" is how long it takes us to solve problems & how good our OZ problem solns are.

→ $\sum \frac{cc_i}{pc_i} + \sum \frac{T_i}{pc_i}$ might be a measure. Note that this is a Goal for the GPD: ≡ TM2G

If we are unsatisfied w. this value of Goal & we have been using it for our TM2G (or we have used $\prod pc_i$ & the 2 Goals are ≈ equivt) then we are at a "local Extremum" & must find a way to "jump out" (I have no immediate Suggestions!)

IDEALLY, we want GPD to reflect (only) the bias introduced by the TSG & the original choice of Ref. Machine (both are Big Biases):

$\boxed{\text{BIAS}}$ ≡ Modifn of the Apriori

Another Source of bias: In "finding short codes" we use various techniques to find regularities → Certain regys are much more likely to be found
This looks like a Serious Source of Bias. Is it desirable or Undesirable? ←

A Good Lrng System will have built into it, devices that on long term, get rid of undesired Sources of Bias. ALP (& presumably QLP) automatically ~~[illegible]~~ reduce bias due to selection of ~~[illegible]~~ Ref Machine, as corpus size ↑.

→ (spec on BIAS 80.26)

.00 76.40 : ~~[illegible]~~ Problem (10), cont.

A condition in which 76.40 may be true (all in the MTD) in a practical sense: TM is solving hard ~~[illegible]~~ sci. Scientific problems for which ~~[illegible]~~ no one else would have little a priori info — That it has as much info in this area as Ramanujan had!

In ~~[illegible]~~ Art. Lang. lrng, reading about Humans, and evaluating Social Situations, TM may ~~[illegible]~~ Never approach Human Info Content — But on the other hand, w. much larger CPC than Humans, & much more input from Human Literature, it may surpass Humans in these areas as well.

Main possib. deviations from Optimality. (1) All pract info is not usefully representable by a PD (2) say, while (1) is true, the updating Algm. is inadequate. — one of the 2 ~~[illegible]~~ .09 of TM2G's may be correct, but we do not update it efficiently — (76.28) — when & what PDs in time to switch between ~~[illegible]~~ S.I. and "Main Problem".

.12 → (3) (76.38) That our model of learning is el. & this is not the ~~[illegible]~~ best way to do lrng!

However, in the "long run", TM can use any decidable prob. solving/srch techniques. So eventually, it should be ≈ "Optimum".

So, an impt. part of the Analysis is the 3½ points (70.05R) showing "how all human info can be in p.d. & how TM can put it there & ~~retrieve~~ it usefully

.18 In Lecture (& to Marcus & Juergen) discuss Korsybsky/ "El" v.s. "Non-El" approach — & ~~Hegelian dialectic (1) El (2) Non-El (3) synthesis.~~ / Synthesis S S

In the present case, the El approach seems to be not bad — also it suggests how to write the Q's - how to make "One. Nets"

Author ≈ Non-El. approach: I am depending (at the beginning) Very much on how Good my TSQ is, & I may have to work hard to get an "acceptable" TSQ.

An alternate approach is not careful about the TSQ, but gives it reasonable environments w. some problems in them & perhaps a Reinforcement function telling how happy the User is w. TM's behavior. (This May be ≈ Juergen's approach).

I don't know enough about his latest system to comment Much! My guess is that they take a long time to start being smart — able to work diff't probs. — & that usually the user doesn't know how well the system is doing, or how it is solving problems — so it's hard to teach the system well. — Also I doubt if we would have any idea of how close it was to Optimum.

3/16/01 TM : GA:

c: ~~Bay~~ Gyan & Datesia
Wmanud(.c)

.01 Two Gross Inadequacies of Conventional GA:

1) The method of obtaining t. Next Generation of cands along w. various params of those cands, is poor.

2) The MonteCarlo Method of Parkal Selection is poor.

.05 3) There is No System Memory of previous Problem Solns. FPGA

The SGA approach to ①: Given a set of cands & their G values (≡ present Generation), Construct a Stoch Grammar / Model → we get $P(G_i | \text{cand}_i)$: where cand$_i$ can ~~be~~ ~~any~~ be any desc of a cand. (SGA) says "use (models/Grammars) in which ~~easy~~ ~~easy to f~~ $P(G_i | \text{cand}_i)$ enables easy finding of sets of cands of hy G_i.

If we select ~~so~~ cand$_i$'s of hy expected G for next Generation, this is a very "Greedy" approach → .19 .24

.18 .19 .20 (AH) TM can learn [Non-Greed]: It can learn to look at a / An approximate ~~...~~ Grammar & ~~...~~ its known cands w. their Gs & suggest new cands that would ~~...~~ make it easier to guess at much better Grammars → .24

(for 3) Use ~ MCT & assoc ideas! We ~~Do...~~ devise a function of prob dens that maps into an initial population of cands. As TM matures, this function gets more & more accurate.

.24 .18 .19 → So a view of t. "problem": we have a set of cands & a grammar that represents t. cands (perhaps as well as could be done w. t. Given Sample)

.27 Problem: What is a ~~...~~ trial cand or ~~...~~ small set of trial Cands that will do most "toward telling more accurately what t. "true Grammar" is? In general, those ~~...~~ trials need not be of hy expected "G". Note that cands are x_i, G_i pairs.

We want a "Grammar" that is a ~~good~~ good pd on $[x_i, G_i]$. Such a grammar can ~~...~~ take many forms; One gives a G_i for each x_i; & there is a common Vore. (or a pd for each x_i G_i pair) so $x_i \to G_i$ & G. is t. diff ~~...~~ thing w. t. Stochastic Grammar.

Also "improving t. Grammar" in .27 usually means improvement in t. parts of hy G.

3/17/01 IDSIA Things to tell About at IDSIA

1) Mechanics of How TM Works : Some impt topics! (Detail.)

a) General flowchart.

.01

.02 b) TMZG [Q: Just how does TM work in improving its own workings? — It does have the Over-all Goal.] Explain MCT, & The 3 reasons why all hours can be expressed by GPD modifns. GPD₁ v.s. GPD₂

c) Meaning of "Factor of 4": assumptions made (How the assumption about "factor of 2" "fractional time spent" is sort of silly it's wrong). "Factor of 2" is more correct — but still wrong. How to fix "Factor of 2" : (see ⓑ & ⓒ)

.05 Why almost all things are expressable as Modifns of GPD+Lsrch If IF all mods by / D ① 2 or more fast. ② Heuristic according to PC ③ To realize Heuristic P.D. done. Pseudo TSQ ④ To realize many heurs, mods to GPD needs to be quite broad — e.g. traces of previous conds' realization. Hour used.

d) How TM can realize any method of such, or any method of working problems including any method of such. — So in long run, TM can learn any method of probsolving.

.10 e) How we can do lrng during srch.

f) Give examples of INV probs, OZ probs: Various types: Some not directly solvable by Lsrch. → 72.05

2) GA: where wrong w.r.t. of methods? (See TM GA 3/16/00) ibid .01–.05 ⓐ .1, 2, 3

3) WON (And/or Nets): See Sheet on WON

That $\sqrt{p}$ is best Mt. Carlo wt. for ~~[illegible]~~ replicated levels

.20 4) Discuss ~~[illegible]~~ & St. Peters burg paradox! Effects of cutoff at any particular level. How it need not get best PC for given T.E. level.

Method of Grsh using probty

5) 3 Modifns of Lsrch: ⓐ PC order $n < (pc_j)^{-1}$. ~~[illegible]~~ — prob. better than ⓑ — The actually ~~[illegible]~~ close for "drafting" Is best if all cc_j's are same as in OZ probs.

ⓑ T ← kT (k = 3 is best, but k = 2 or 4 is not much worse

o Other advantages of large/small k)

ⓒ Time share: Almost always better than ⓑ, ~~[illegible]~~

ⓓ use of pc to guide srch, rather than 2^{-K}: could be fair [redacted] factor larger.

ⓔ $\frac{pc_i}{cc_i}$ order: This is best. Can be a by-far better than ⓒ (See. 65.30–.40 for details)

6) ALP & RLP : That RLP is probly best Induction that one can do. Maybe discuss "Necessity Thrm". Discus RLP defn. as a OZ problem: The 4 or 5 important params of probty (≡ Lrng) That all of them are impt.

.30

Time to calculate Exactly.

→ 79.00 on This TALK →

3/21/01 on TMZG (.00): Just What is Goal

Just wanting Goal, & how does it take cc into acct?

63.01 ff

63.03 ff is on this, & This may be about as good as I've Gotten! — Ro

64.01–.04 appears to be "final soln" (But See 76.00–.03 for More recent View!)

There is earlier discn of an alternative TMZG → 63.26

.34 — TMZG A serious problem with TMZG "soln" of 64.01–.09! For OZ problems: when we modify GPD so as to ↑ the pc_i of a "successful" O.T.! The Time available for soln. ($T_i - \alpha_i$) will usually change — so only if we ~~[illegible]~~ pc_i ↑ or remains the same, & α_i (the amt of time to generate pc_i) ↓ or remains the same, will we really improve (or remain same) our GPD.

If pc_i ↑ a lot & α_i doesn't ↑ much or $\frac{(cc_i \alpha_i)_{new}}{pc_i \; T_i}$ > 1 then it looks good.

I think we run into the same problem in INV. problems. We wanted to minz $\frac{cc_i}{pc_i}$, but when the GPD is modified & pc_i changes (for the "same" soln. technique) then cc_i can change, because cc_i includes time → 71.00

79.00

3/23/01 IDSIA

"Doomsday" file from Marty, "IDF file"

spec. 65.03

.00 : on GHTI! proof! If we have any order of cands & exchange the order of 2 then E search time will
.01 ↑ or ↓ depending on $\frac{cc_i}{pc_i}$ orders of the 2 cands. Any ordering of the cands can be obtaind from any other by a sequence of exchanges of adjacent cands. It is probly easy to show .00–.01 is true ~~for~~ adjacent cands, – so if we only do adjacent exchanges ~~for~~ to get better $\frac{cc_i}{pc_i}$ order, our expected srch time will always ↓. This means that the lowest pt. is always lower than any other pt. (reachable by exchanges).

.05 Perhaps make a list of kinds of OZ, INV probs & give examples (Needed for talk).

1) INV a) The classic P & NP problems of Comp. Compl. Theory: (Show how. Solns to equations,) Proofs, TSP.
b) INV, but w. expensive yes/no: Same as inexpensive. c) "INV" but expensive gray yes/no ← This is an OZ problem.
2) OZ problems: a) Simplest: Noiseless, Not time varying: (Adding Noise has little effect on method of soln. Using probilistic methods, but usually takes longer (is more complex soln).) Give examples (design of ckt w. certain characteristics, min cost, & MO)

.10 (b) Next add noise (no big deal)

(c) Time varying: If rate of ~~[illegible]~~ variation w. time is slow wrt. clock speed! Try to make model of time variation of system, and ~~[illegible]~~ internally do an Lsrch to get an optimum for expected present ~~[illegible]~~ "reward function"; If variation is ~ of faster than clock, it will have to be treated as Noise (.10)

(d) ~~Approx~~ Not time varying but expensive Gen. Make model of Gen: ~~Do~~ fast internal search for ~~[illegible]~~ → obtain trials for RW. (This is Greedy). For non greedy approach, ⇒ optimum

.15 Use Finite horizon Dynamic Prog.

(e) Maximize ~~G~~(x)·f(t) ← G(x) &/or f(t) can be open or closed. If f(t) is closed, it would seem that we can't do much. : This is one kind of Time Varying OZ problem.

(g) Anytime problems.

(h) Give example (1) Matrix multiplication in minimum $2n^k$ w. min k. (include proof). (2) show how induction is an OZ problem.
We have a list of algms + proofs that time $< 2n^k$ (for large n).
We want one w. largest k.

What of the evaln. funct for INV is wholly or partially "open".
In both cases TM would want to find faster way to compute it or approximate it; "Quick start" is in Pats area.

So: Kinds of evaln. functs for INV, OZ:
1) Fixed v.s. time varying 2) Noise, Noiseless (noise can be a kind of time varying) 3) ~~[illegible]~~ Eval. function can be closed or wholly or partly "Open".
Time varying inv. probs: rather difft — maybe not so bad if Evaln. funct is "open"
2) ~~for~~ OZ probs: "Anytime" variation

.34 The Lsrch technique described really doesn't solve time limited ~~opt~~ probs, i.e. if we are given time limit T, It takes time $\frac{T}{pc_i}$ ~~[illegible]~~ to solve the problem optimumly. — well it can though do TS version — or even T&T version. If we really have only time T available, perhaps best soln
.36 would be ~~[illegible]~~ time shared version of Lsrch. In discussing Lsrch make this clear

3/25/01 IDSIA:

Initial Lecture:

70.49: Door... Background: Criticism of ~~most~~ GA (Koza) & Most other Lrng Systems: trivial, Solve problem ~~[illegible]~~ W. General descr. of what System Does & how it Works: will give [illegible], give more [illegible] pattern.

Language Lrng, Visual Scene analysis, Acoustic " " -, Reading General printed text. ...

— What Kinds of probs it can solve.

" " " " " can't " — Ultimately, all describable problems. (via "QA" format)

see .17

So start the System is like a ~~[illegible]~~ newborn baby!

Solves simpler probs — Then is smarter;

So then solves harder probs....

3 kinds of probs: INV. OZ. Anytime.

Main "Master" types.

Details: 1) What do you mean by problem. INV/OZ/Anytime.

2) How does it solve problem. (Lsrch) — A search Guided by pc. GPD2

.09 3) How does it incorporate solns so it can work harder probs. ← Hardest part to explain: DO! ... 2 PP Main ideas

M It modify it the pc. 79.33 – 83.01 is on this Q. More exactly 79.33 – 80.05; 80.36 – 83.01. I should write up a "Presentation" of this; & an addendum: Some topics for possible continuation of the Discussion.

More detail. Kinds of Probs: Some details of 72.05 — examples: QA problems. Some probs not INV or OZ. G(x). F(t) (~~[illegible]~~ one kind of time varying OZ)

Many Good Pointers: How to do approx. "Convergence": find theory to hill Metrics. Induction Seq. or BAG of operators & QA probs Compression induction & databases for [illegible].

.17 In general Q.A. problems are OZ problems: So after adequate training one could run to describe any problem in some Natural or formal language, & TM would search for best response. If cost, in the available time. So perhaps any problem that's describable could be worked on by TM.

The main problem is to get very intelligent Machines! This done, most other kinds of problems are solvable.

How Probs are Solved: INV problems: By Lsrch:

Desc. Lsrch viz conditional probability: Properties of Soln. ; GJS: Best Soln within a factor of $(P(x))^{-1}$ < $2^{-R(x)}$: GHT I;

TS Lsrch is not nearly near as good as: But it may be Best possible.

~~How Lsrch is done for~~ OZ problems.

Show Time Share.

.33 How Solns are ~~[illegible]~~ incorporated into its knowledge Base i.e. GPD [illegible]

[Probtype_i, problem_i, solution_i, time taken_i] is data. We want P.d. on this data. Most desired form:

GPD1 — input) probtype, prob, soln, ~~[illegible]~~ output: Pd of soln. time (may be very long, unlikely Soln)

GPD2 (convert to | input (type, prob, soln) | output probs that that's ~~[illegible]~~ fastest Soln for that problem

Ideally, arrayed so that they are in (very roughly) pc order. (Only solns w. pc > a certain threshold need be considered) ← But threshold depends on T value (for ~~T=2T~~ method)

~~[illegible]~~ Also, in TS method, it will depend on how long one has been working on the problem.

→ 80.00
→ 81.01

3/26/01 IDSIA:

First talk (cont)

.00; 79.46 More Exactly, what we want is (for GPD₂): input is (problem) [includes type of problem.] output is set of (soln, pc) pairs in rough pc order. a "soln" is a program ~~whose output is~~ for a good pgm, it's output will be correct soln & stop.

that's for INV probs: for OZ probs: Input is problem, output is an [O.T., pc] set. ← GPD₂

.03 for GPD₁ (for OZ probs)! ~~output~~ input is (problem) output is PD on G value for any particular O.T. (w.r.t. input problem)

Perhaps Go thru details of INV problems (GPD₁,₂) only

Then just say what GPD₁,₂ look like for OZ probs in Anytime probs

Don't do "Anytime" probs; I think Tinged hurdle OZ problem & technique is fine!

1.25 DM

.43

Mr r 2 $ 60

.45

Major BUG?

.09 Serious Defect?

Note: [redacted] O.T.'s tend to be correlated. [Also Note 101.30] E.G. If they are H.C. methods [redacted] w. Lots of trials: t. trials from one O.T. can be used for t. next. O.T.

7.33/dollar

On Optimality of T.S. Lsrch: $\frac{pc_1}{cc_1} > \frac{pc_2}{cc_2}$ No New Ideas!

Fr franc

1366

3/26/01 for Jan/01

Re: (09): It may not be critical, Because [redacted] TM can design OT's of any conceivable type that do take advantage of trials in "other OTs" — But the "other OT's" are all in t. same O.T. — So, unless Cands can take advantage of info in other 68 trials, t. system is inefficient. This is a form of learning during trials & can be dealt w. via t. techniques of 62.19 ( ) — Hvr, Re OZ case may have to be dealt w. in a special way. There is a great no. of O.T. methods that are H.C. methods & they do new/micro trials based on ~~the~~ recent micro trials. Systems that do this could be combined in special ways.

131 66

2001/02!

7.1261/dollar

F.(403

.26! 77.40 On Bias: In general, there is a tradeoff betw Bias & SSZ. In a good System, large SSZ should be able to reduce any source of Bias (?) but a good lang.

One kind of Bias: Selection of ~~data~~ Sample data from RW, is harder to deal w.,

System will eventually give instructions on how to take data from RW.

In mapping info from RW into TM, we can effectively give zero a priori to certain data, [redacted] by not including it. This extreme form of bias does not occur in other operations of TM

.35 I.e. all induction/products have pc > 0. → (81.08 on Bias

.36 Back to lecture! 79.09 Given t. initial P.D. & some new data, how does it construct a new PD that incorporates t. New Data? In steady state, this is usually not a problem: In steady state, we have all of this data on problems, solns, soln times, times for unsuccessful solns, times for ~~fail~~ trials, etc. We also have an apriori. → to convert this to a new (≠) apriori.

→ 81.00

3/27/01 IDSIA

Korona $\frac{5}{4}$

Between Scala: Electron.

1400m → 84.

4 v.s 4-1

$\frac{1}{2}\ \frac{2}{3}\ \frac{3}{4}\ \frac{4}{7}$

$\frac{1}{2}$ v.s. $\frac{2}{3}$ | $\frac{3}{4}$ vs. $\frac{1}{2}$

4 v 3 $\quad \frac{3}{4}$ v.s $\frac{2}{n}$ $\quad \frac{2}{n}$ -v.s. $\frac{1}{n}$

80.40

.00: ~~79~~.40 If the apriori is in the form of a reference machine, we try to find codes that maxe. the

.01 pc. of the observed data. This is [ENV, 02, induction (say, Bayes) . . . acc] data → 83.26 for more on this

MoR:
If one solves
2PDG normalize in
complex plane —
produce "no" local
extrema"?

.02 Consider sequential predn. Our apriori can be in 2 forms: ① UMC, ② summarizing mach.

.03 UMC: Given corpus; get apriori; Given augmented corpus get apriori; Predn is ratio of aprioris.

Any [redacted] form of apriori could work same as .03.....

In (.02), we could use a "Summarizing Machine". Its input is random strings, its output pc's (desired pc's) of our predictions: (This is v.g. form for classical induction). In simplest case, a "summarizing" machine works only for "K" codes + (perhaps) OSL data.

.08 I Guess "Summarizing" must introduce Bias — Gambas (80.35) — for the fact it changes the effective apriori & substitutes an Approx to it.

When we use a summarizing machine, & we have new data, how is the ~~[illegible]~~ Sumz'g Mach updated?

[Perhaps the OSL part is updated in a simple ("trivial" ☺) way. —]

We run ~~the~~ codes thru & ~~[illegible]~~ save "states of Machines" of ~~K~~ best codes; (This will work for only "a few" updates. Then we have to do "serious revision".)

The foreg seems "not bad" for seql. predn: but what about BAGS "pure BAGS" (not stoch operators): How are predict. we find 2 part codes: 1 for "model", one for "contingencies of model". (≈ to MDL — but no provisions for OSC — So, we retain the K best ~~[illegible]~~ "models" for our Summarizing Machine" — Are such provisions nec'y? [I think so!])

& figure out a way to do "OSL". (General. wts of approx machines may change w. "updating".

.20 [redacted] Other Induction Methods ("Models"): **Z141**: very simple model. —

.21 Automatically deals w. OSL. An Z141 "Model" will consist of a list of nested defns.

The Corpus, then follows; we can code the corpus in various ways ("parsings"), we select best parsings. We can select best set of parsings periodicly. We use the pc's of defns. assoc. w. latest parsing to decide how to best parse the next "chunk", When this is done, we recompute probys. At certain pts, we make new definitions, based on latest parsing(s).

.28 Probably would be good idea to apply Z141 to BAG lang — 81.28 – 82.05 does this! One way: we have this set of Nested defns (as in .21). The pc of the corpus is expressed by parsing the corpus in terms of the defns. (including, perhaps, a "Stop" symbol that has a certain pc)

.31 The pc of the Dern, hvr, is obtained differently from that of the Sequential corpus.

Guess: That pc of .31 is obtained first, perhaps using the same methods as for sequential Z141. Hvr, perhaps the "Stop" symbol is a primitive. After parsing w.r.t. the "Dern" of .31, Say the no. of symbols in the corpus is n. n_i are of type i $\Big]$ $\sum_{i=1}^{k} n_i = n$. ("Stop symbol" is one type, say).

We would like the pc of the dissection of n into k parts. One way is to simply count how many ways there are to do this. Each symbol must occur at least once — so we look for k unrestricted partitions of n−k. $\frac{(n-k)!}{(n-k-n)!\,k!}$ perhaps is $(pc)^{-1}$. **No!** $(n-k)^k$ (tries range of one number) upper

Well, this can be worked on later! A "puzzle" Exercise for the student/reader" $\frac{(n-k)^k}{k!}$ may be correct ← Many of these are identical, so it is upper limit 82.00

.00 : 81.40 : So this number f(n,k) is some function. w. the pc of 81.31 & the pc of f(n,k), ~~[illegible]~~ we obtain the pc of the stoch Grammar. To get the pc of the corpus,

Mult ↑ by $\prod_{i=1}^{k}\left(\frac{n_i}{n}\right)^{n_i} = \left(\prod_{k}\frac{n_i}{n}^{\frac{n_i}{n}}\right)^{n} = \left(\prod p_i^{p_i}\right)^{n}$ — A familiar Express.

perhaps multiplying it by $(f(n,k))^{-1}$ will make it even more familiar.

To obtain the deriv, of a good grammar for a unparsed corpus, use ~~[illegible]~~ recent work

.05 on Z141 + Wolff's "reparsing" idea.

Note the function f(n,k) assumes a lot (bias) about the distribution of pc's. —

~~There are probably many Alternative Solns. to the problem~~ → (Even with this in the thing I wrote to Wolff about! ~~at least~~ a possible cost function: Using the cost of making corrections to errors in coding a corpus.)

While Z141 is a "simpler" model, it is able to deal w. definitions and alternative parsings.

Also it has a methodology ← Grammar Discovery Routine — for assigning pc to a corpus, ~~[illegible]~~ (i.e. Sol/Wolff method) s/w method

~~[illegible]~~ That will work when the "primitives" are arbitrary "functions" — so we can assign P.C.'s to ~~[illegible]~~ arbitr "Lisp-like" pgms (≡ Functional Prog Langs).

As yet, I don't know ~~how to start~~ a discovery routine for CF Grammars of "pcolograms" — ~~But~~ nevertheless, ~~[illegible]~~ it may be possible to apply the genl. technique used to those Grammars as well —

In fact I probably did do it in the original Z141 ~~[illegible]~~ report — a very likely in Sol64 part 2.

On 81.20, I mentioned "other induction models": ~~Z14~~ 1 w. CFG ~~[illegible]~~ 25 direction of complexity of such models: (Note / stoch CFG's are normally used for BAG induction ~~[illegible]~~ — which is ≠ what I need for TM2. — Hvr, it's more like stoch operators (which is an impt. subclass of BAG induction).

OK. Consider the updating problem. The first thing one does is simply update the pc's of the generating Grammar, w.r.t. the new elements of the corpus — which were parsed using the "old" Grammar. After perhaps a few poss. "Good parses" are considered, we look for new regys in the augmented corpus. e.g. new definitions. We might backtrack a little & ~~[illegible]~~ look at the corpus a deriv, unaugmented by ~~[illegible]~~ a larger set of recent examples — try to reparse & look for new regys. (I'm rather vague in my mind on this — but it looks like a "Rocky Revision".)

The Biggest Possl. "Revision" is to discard all old grammars & take all of the data & make a new set of grammars for it; (very hard!). By (retaining) keeping a larger set of Grammars at each stage of minor revision the ~~[illegible]~~ need for major revision occurs less often.

→ 83.00

3/27/01 (3–7 AM) 83

IDSIA =
KORONA stock, Square ; 4 screws ; 1 kg weight ; ~ 190 dm → ~~[illegible]~~ 54 $US.) ~3.40 p 6 left
↑ what Country, what city ; Address, phone, Model no. ?

● .00: 82.40: Other kinds of (≠ 1 2 41) regularitys:
.01 Algebraic, Musical Regys.

Anyways when I'm trying to write a ~~TSQ~~ : write down just what kind of search laws I'm using for GPD, & GPD2. (≡ Human Heuristic)

So, perhaps Main Weak pts (areas) of the System:

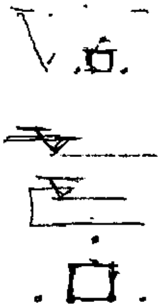
(1) Laziness During O.T. Trials: & perhaps during INV trials. This can be done in ~~usual~~ usual way, but must ~~be speeded up~~ have to be speeded up means: I'm not sure of whether & how well it [redacted] works.

(2) The general updating system for a bunch of OTs of various types & Inv systems, is not very clear.

(SN) I assumed pc was > $2^{-R(x)}$, but the pc would not have the same cc assoc. w. it — as $2^{-R(x)}$ has — it ~~may~~ take more time to generate for ~~Least~~ Lcost of the 2 forms — Well it's not clear that pc form is better!

(3) In "improving the GPD", [redacted] when the GPD [redacted] consists of many very heterogeneous methods of prob. solving, it may be diff't to find & implement regularities that are common to many [redacted] of them.

● .18: (62.32) (SN) On Lazy during Lsrch 62.19–.32 describes modifn. of TS method of Lsrch: A possl. Simplifn: Before the "Lazy" the TS had a certain level of Lcost ("T") that it was running at. After "Lazy", TM goes thru the set of cands & works on each one until the "Lcost" is up to previous "T" level. This may not be optimum in one way, but from a practical pt. of view, it may take much less time than the Lcost update System of 62.18–.32



.26: (81.01) For BAG extension: The first members of the BAG are given pc's by the ref. machine w. null input to "Machine data" input of ref. machine. After we have known members of corpus, we update by using non-null machine data, & data of various (e.g. R's) for BAG members.
(Subsequent updatings can leave Machine data invariant, & modify data of BAG members only ! ??)

What kind of updating is modifn. of pc's of "elements" as in the "~~BAG~~ 241" treatment of 81.28 ? It amounts to modifn of the "T" Number Ratios between zero & f(n,k) of 82.00 — So I think this is part of the "Machine data". The "Corpus data" is just the sequence of "Elements" (including "stop" symbols")

3/29/01 IDSIA

.00: : One form of the Q.A. problem: Say we have a large data base & we ~~have~~ have a corpus of Q, A's about it [redacted] & we want TM to learn to (Answer / respond to) new Q's. One way is to regard the data base + known {Q,A} set as a corpus & ~~we~~ we want to find an operator O such that best helps code the {Q's A's} + the data base. Say D is the Data base set. Then we want an operator O ∋ $O(D, Q_i) \to A'_i$ and the codes for O plus corrections in $A'_i \to A_i$ are minimal ($\sum 2^{-l_i} = \max$)

First, we, presumably know how to formalize it if $D \equiv \Lambda$ (null). There are 2 kinds of

.07 probs of this sort: ① regularitys in the $\{Q_i\}$ set are of interest

.08 ② " " " " " are not of interest. see .33

.09 In ② we want operators O ∋ $O(Q_i, b_i) = A_i$. (b_i are correction bits)

Total info in O plus Σ info in b_i is minimal.

.11 In ① ~~we~~ 2 ways to look at it: ⓐ Soln to ② + P.D. on $\{Q_i\}$

ⓑ P.D. on $\{Q_i A_i\}$ viewed as a BAG induction problem.

.15 The problem of .00 is closest to (.07/.11), but it may be more close to (.08/.09) in that the info in the set $\{Q_i\}$ is not of interest → see .33 — which suggests that we want a code for O and D such that $O(Q_i, b_i) = A_i$ and total code for O and D and $\{b_i\}$ is minimal. (more exactly $\sum 2^{-l_i}$ is max)

.18 {The "trouble" is that There may be much Data in D that is irrelevant to $\{Q_i, A_i\}$, so to obtain O, we would be wasting a lot of time on it. / yet getting a lot of compression for the corpus as a whole. The "soln." of .15 is for CC = ∞: it

.20 may well be that the soln. for finite CC is $\boxed{\text{quite different!}}$ ← This Situation is of MUCH INTEREST!

An extreme case: D consists of 2 hunks sub-data bases D_1 is relevant to $\{Q_i, A_i\}$

D_2 (which may be comparable in size & have many interesting regs) is completely irrelevant to $\{Q_i, A_i\}$

However, the ~~irrelevance~~ irrelevance of D_2 cannot a priori be ruled out, until it has been examined & found that for large CC, no relationship has been found — (but like randomness, we cannot, for finite CC, be sure!)

With finite CC, we may find that we get much more compression of $\{Q_i, A_i\}$ (whatever that means!) by compressing D_1 & $\{Q_i, A_i\}$.

Consider we do .00 entirely like (.08 - .09) so regys in D "are not of interest".

So O must look at D & Q_i and output A_i; w.o. necessarily looking for regys in D.

.23 {When I say "Not of Interest": Possible meaning: That ~~introduction of this info~~ use of this info introduces ~~Bits~~ Bits we'd rather not have} No BAD: But Much Further

.35 ~~[illegible]~~ A Simple Generalization of the problem of .00 (& .18 - .20): We have a corpus & want to do

.36 sequential ~~pred~~ prediction on/extension of this corpus. The early part of the corpus ("first half") is not much relevant to extrapolation of the second half. If one has CC = ∞, this causes no trouble, but for ~~finite~~ ~~[illegible]~~ CC (FCB), if the CB is very small, we'd want to neglect the first ½ corpus almost entirely. I say "almost" because one has to work on it a bit to realize (85.00)

3/30/01 IDSIA

(Defn/Soln) of RLP problem in BARCelona Paper is Wrong — .01

● .00 ; 84.40 : That (the amount of "productivity" per cc in the extension of the 2nd ½ corpus) is small.

.01 What this means is (perhaps!) that the (soln of the prediction problem) for RLP is ~~[illegible]~~ Not that given in Barc paper!

To (best) extrapolate corpus, C, ~~[illegible]~~ using cc = Time to ~~[illegible]~~, we do not exactly want to get codes that (lower bnd of) Σ 2^{-l_i}. We may find ~~a~~ much ~~[illegible]~~ better ways to spend our time, if ~~[illegible]~~ the only thing we want to do, is extrapolate the corpus! (e.g. say the corpus is in 2 parts, like 84.36.)

Say we look at "part 1" & decide after some cc spent, that we'd do ~~[illegible]~~ much better, spending more

.08 time on "part 2". This decision could introduce a lot of BIAS into the final Extrapoln.) → 86.09

.09 As a general Conclusion (I think I got a little flavor of this by ~~[illegible]~~ looking for codes for (corpus)^a v.s (corpus)^b → posbl. Continuns. ... how ~~[illegible]~~ much (& what kind) of ~~of~~ time, resources, to spend on these 2 posbl continuns — An a (less BIASED more objective way) → I suspected finite CB would look for codes for ~~[illegible]~~ all strings w. (corpus) as prefix. I suspected finite CB would — 86.03 → (deeply.)), is that finite CB will

.13 'cause trouble, but I don't think I went into it very deeply. The bias → φ as CB → ~~∞~~, but it may (always?) introduce Bias into extrapolation. ↓ Very slowly w. CB!

● .18 — — —

Well consider Sequential Extrapoln (as .09–.13). Say one simply uses Lsrch to look for codes — starting w. 0, 1, 00, 01 etc. — say we'd T.S. Lsrch (or even T ← 2T with completion of every round) — using finite upper bnd for T — This would seem like a very inefficient way to search — but it would probably be w. little (or no!) Bias. needs it!

the idea of .01 seems Very Impt. It means that I really don't have a clear defn of the RLP Problem! — So its not really an OZ problem!

On the other hand, in EM I Will be using Lsrch, which is (as far as I know) not much Biased

(OK) : "BIAS" ≡ Undesired BIAS is deviation from "true" (CB=∞) ALP. — Defined by Corpus and Ref. Machine.

Using Lsrch, eliminates consideration of all regys (codes) w. Lcost > T used. Undesirable, but What can one do about it? Its a Great Bad bias — in the sense that I don't know how to do better.

Well what about the 2 part corpus of 84.35ff? Doing Lsrch on the Entire Corpus would seem to be very wasteful. Also, Consider linear/non-linear Regression Models for, say, SM.

● Here, one "narrowly" looks for regys of certain kinds first. Unlikely to be in "Lcost order. The Bias introduced is Equiv. to assuming that those models tested are a priori, most likely. It is (≈) equiv. to Choice of a Reference machine — not nec'ly a Universal ref. Machine —

One Can make this Choice Consciously. ~~[illegible]~~ Normally this choice is made (86.00)

3/30/01

IDSIA

BIAS

● .00: 85.40: "Cowboy style" — by t. seat of t. pants feeling, rather than by any conscious analysis. Conceivably, one might do much better! The Unconscious Mind has its Bad as well as Good features.

.03 : 85.19 A Kind of Bias ([redacted] converse part of 85.09-.19)! Hvr say one did spend more time looking for codes for (corpus)→a than for (corpus)→b. This would seem to be a serious ~~[illegible]~~, systematic error w.r.t. t. ALP codes (cf CB=∞). In [redacted] some cases, decision to spend more time on corpus→a than corpus→b could be a conscious decision — a desire for a different ~~approach~~ ?

.08 3/31/01 The only "Zero Bias" approach is uniform p.d. (Computer indep. post, "Hmm D.C." ∃ no useful predn. possl.) → 104.00

.09: (85.08) RE: Soln. of RLP program BARC! Perhaps t. Maxzn of $\Sigma 2^{-\ell}$ is t. Goal & is a well defined OZ problem! T. "Ply moment" hvr, is that (choosing OZ problem, How one works on it depends on one's priori apr. info — kinds of OT's available, TM's (and wts. to these O.T.'s).

Hvr, what about t. 2 seq corpus problem of 84.35 & t. CDA problem of 84.00? — Just how does (.09) apply to them? Apparently doesn't! .09 doesn't seem to help!

.16 — consider 84.35 we want to extend ratio of pc's of "a" v.s. "B" continuation. So we want to decide on what ~~aux~~ "illicit" info to use in t. predn: we think (second 1/2) corpus₂ is vital for this.

● ((Perhaps this kind of decision has to be made in all RLP efforts! — If so, it is not clear as to what t. Goal is! What is Goal of decision to use ~~many~~ mainly Corpus₂ data? — Presumably t.

.20 Goal is Good induction — which has a clear (no long term) measurable value.

.21 Related SN ~~For~~ [redacted] induction w. small CB, it would seem to be commonly used a Good idea to use only a small part of t. Corpus, for induction → As CB ↑ we find it useful to consider more & more of t. corpus. This is ~ what we may do in SM, where t. corpus consists of ~ 9 or 10 ~~[illegible]~~ parallel time series, & we only want to extrapolate one of them. For small CB, consider only t. seq. to be predicted — Most recent data. As CB ↑, use more data a/o info about "Nearby" sequences. One trouble! That if CB=∞ is not used, we may end up using "Olde Faithfull" "Cross validation" to evaluate predns.! Sounds Very BAD!

Even in normal, everyday, induction we only use a very small part of t. corpus for induction. One of t. things that we greatly [illegible] is approximately how much

.35 (& what part) of t. corpus to use for induction. → 88.00

● .36 A New way to look at t. corpus extrapn. problem; say t. symbols in t. corpus are a b c d: we want to know ~~[illegible]~~ rel. probty of a of t. following: Suppose we do corpus "d" only, using some criterion. Next we do corpus "cd" using same criterion. then bcd, then abcd (is there a limit!). After coding "cd" we may be in a better → 87.00

3/31/01 IDSIA

CR2430 ← Name of Lithium cell in Sale.

7:11

.00: 86.40: to code "bcd", ect — working backwards.
~~Hvr~~ Hvr, even if 86.36 ff were correct, it is not always clear as to what parts of t. corpus are "near" (≡ inductively relevant) to t. part to be predicted: perhaps most apparent in BAG problem.

But its a kind of start. ~~It~~ Its not clear as to how much time one spends on coding "d", v.s. "cd", vs. "bcd", ect. Use ordinary "monotonic" UMCs, it is easy to continue in t. forward direction. We could try running ~~the~~ corpus backward. Since we would be interested in further extensions to t. corpus only, we'd have to find codes for each possl extension — which is a real Drag! Hvr, since we usually don't code

.08 Very far back, each trial may not take very long! → .33

.09 SN On ~~finding~~ Codes for corpus using Normal "Monotone" Machine. Say we have a sequential corpus & we have a code "α" that has output matching corpus to a certain pt. When we augment α w. all possl. continuations, we get things that don't track t. corpus. We then backtrack α by a minimal amount so we can try ~~changes~~ useful changes in ~~[illegible]~~ "reduced α" (≡ β)
So we then ~~try~~ try all ~~[illegible]~~ possl continuations of β (other than α). If one works, we try to continue it, if not, we backtrack as ~~we~~ backtracked α, ect.

By keeping a lot of info in memory about branch pts. in t. backtracking — forward trials at a given pt. that were (successfully / unsuccessfully) tried, etc, we may be able to backtrack

.18 rather rapidly, & perhaps find codes for a corpus fairly rapidly.
→ T: amt. of undesirable BIAS in t. foregoing trick, is unclear.
But anyway, .09 does sound much faster than just ~~[illegible]~~ ~~trying~~ doing random ~~[illegible]~~ trials in lexical order, using Lcost cutoff.
(We'd have to use Lcost cutoff in .09 also, & that kind of "failure" of a trial branch would have to be recorded properly in RAM, — This may ↑ complexity of t. search a bit.)

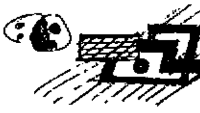
.27. T. idea of .09 may be t. same (or ≈ same) as using ~~[illegible]~~ a set of "summary codes" after each augmentation of t. corpus.
It is possl. to start from t. beginning & code a corpus this way — backtracking to get more ~~[illegible]~~ possl. codes, occasionally when necessary.

.32 perhaps use (.09 or .27) to try ~~[illegible]~~ generating "Lisp" codes for a sequential corpus.

.33: (.08) → Hvr, one starts out w. t. hypothesis of t. no. of codes for each contin. of t. corpus!
∴ Alternatively, one could "start out" w. t. machine in various possl. states (corresponding perhaps to different further possl. "continuations" of t. corpus. — But that would seem to add info to t. codes! (Drop this for t. present!

.36 ← This system has other serious diffys!

→ 88.00



● .00: ~~86.40~~ 87.40 ! I'm remembering Carnap's Idea, that Induction has to include ALL available DATA! While this is true for CB=∞ (ALP) — it can't be true for RLP.

Going back to the problem(s) of 84. ~~.00~~, .18-.20, .35 ← 2 parts to Corpus, 86 [.16-.20] impt also (.21-.35)

One Way toward a defn of the Goal: In our induction, we want it to be "minimally biased" for the CB used. The idea is that we want to eliminate as little out ~~that~~ of CB=∞ codes & include as little as possl. from codes not in CB=∞. Actually "2 goals" — & they can conflict! We want their [redacted] difnce to be as small as possl. (I goal)

.08 Consider BAG example: ● Corpus = 3 objects + part of 4th (sentences α, β, γ; $\Delta^\wedge$). To get pc's of possl. completions of $\Delta^\wedge$.
We try to find parts of codes for α, β, γ that occur in $\Delta^\wedge$. These go into "Common Machine". In working on α, β, γ we do find some codes common to $\Delta^\wedge$ in α & β but few in γ, So in the ~~remaining~~ remaining time, we spend all our time on α, β & $\Delta^\wedge$. (we cutoff work on γ) So essentially, the corpus becomes $\alpha \beta, \Delta^\wedge$ & the Goal is ~~top~~ to maxz $P_\alpha \cdot P_\beta \cdot P_{\Delta^\wedge} \cdot P_\gamma$ Any partial coding of P_γ influences ~~the~~ structure of "T. common Machine", but P_γ becomes a constant. If $P_\gamma = 0$ thus far, we just try to maximize $P_\alpha \cdot P_\beta \cdot P_{\Delta^\wedge}$.

So, essentially, we just try to maximize $P_\alpha \cdot P_\beta \cdot P_{\Delta^\wedge}$ in either case!

● .18 I'm not sure that ~~that~~ (unless $P_\gamma = 0$) I don't want to maxz $\boxed{P_\alpha P_\beta P_{\Delta^\wedge} \cdot P_\gamma}$

— But .18 is not critical at this pt. — Is there any way I can make .08-.18 more exact — ?

→ perhaps more "goal oriented" The top Goal is not clear.

A recursive Soln! (Perhaps!) : Given (INV/OZ) problem defn. Work on all of corpus for
.22 α To time. / Based on recent results & previous experience, choose subcorpus & work on it for To time,
(~~choose~~ Goal of choice of sub-corpus is to maxz ~~the~~ ultimate likelihood of best estimate of pc values in Available time) Go to α (on line .22).
Do this until available time is used up, or until problem is solved.

.26 Another view(?.) : The GPD$_1$ has as an input: problem, [Sub Section of corpus to be considered for induction], Cand's (in) Cand soln; ~~cand soln~~;
If its an ~~INV~~ INV problem: / output is PD that cand soln will solve it in CC=T.
.28 prob. for section of corpus to be used, cand's

.29 .26-.28 may be a bit off! The problem in Question was induction. Usual form: (for seq. extrapln (α))
Given Sequence S, to get rel pc's of S^a v.s. S^b. Barc said, the problem is: equivt to finding codes for S $\ni \Sigma 2^{-\ell_i}$ is max / w. time limit, T.

● .36 GPD$_2$ looks at the problem defn (which includes "S" & T) and outputs an infinite set of O.T.'s [redacted] (w. a pc for each). Each O.T. looks at the prob defn & outputs [redacted] a set of codes (in time T or time $\frac{T}{pc}$.)

.39 · For the BARC RLP induction problem, our "O.T. for induction" algms, look at the corpus, S, & time limit T, and tell which subcorpus to use for induction. — Then (89.00)

.00: 88.40: Use Lsch (or whatever) to find max $\Sigma 2^{-l_i}$ codes.

.01 A variation: first looks at corpus & then decides above Subcorpus — (But actually it is

.02 GPD "not looks at t. corpus"!) → .33

RLP Induction is no ~~longer~~

Hvr, It does look like induction is no ~~longer~~ a pure OZ problem! —

So it's a special kind of problem — but we may be able to deal w. it by t. special "O.T." of 88.39 – 89.02. So essentially: induction need not be a OZ problem

.07 but it can be a special kind of problem that is solvable by TM's "general methods." → .20

Consider Case in which

.08; 71.40 SN On t. "FACTOR of 2" problem: Say "k" ratio is .5. ~~[illegible]~~ "Narrowly updating" GPD after solns to problems, takes about same time as problem solns. If so, there will be no time for integrating diff't problem types — this kind of work being equiv. to Using a larger part of t. corpus for "GPD improvement". It would seem, then, that w. k=.5 TM would never do much critical integration of disparate domains!

Hvr., Argt at 71.21–.24 would seem to apply. ← BUT I don't feel comfortable w. this Argt —

I'd like to do an analysis of just what is going on!

For .08ff + case of, a better thing to do would be to take a long seq. of problems: Use a hy k fraction on t. first ½, say. — Then t. second ½ would do each — ΣL_i problems much faster — so we would end up w. more done. — But not by a factor of >2!

.20: .08 [redacted] So, for seq. induction probs, GPD gives set of (conds / prms) (w. assc pc's) for soln. [redacted] For some (very large) corpi, a cond must of necessity, decide to ignore certain parts of t. corpus (possibly after some examination of those parts.) — or (positively expressed), to include only certain parts of t. corpus: This decn of what parts of corpus to work on (↑ prob. length contains info) ∴ ↓ t. wt. of that cond.

In SM, say, one has to get pc for tomorrow's price. If one has time T to make calcn, one can only consider a finite part of corpus, whose Length ↓ as T ↓. On t. other hand, for certain kinds of time series (say linear regression parameters), σ^2 ↓ as length of corpus considered ↑, — but σ^2 ↓ slowly (∴ precision ↑). So (often / usually) ↑ amt. of corpus considered, will ↑ accuracy of induction — but this means ↑ in time spent on induction. For given T, there may be an optimum corpus size to consider & optimum way to look for regys.

.33: .02 So GPD₂ looks at problem (which contains corps or address of Corpus) and what has to be induced & how much time is available for T. output of GPD₂ is a set of O.T.'s for induction! Each one has a decn of What part of t.

.34 Corpus it will try to Code as well as a technique for finding codes — (& a pc for each O.T.)

.35 T. GPD₁ that gives rise to that GPD₂ (of .33) is some input as .32: Output is set of cond.: O.T.,

.36 [redacted] a poss. subcorpus, a pd on $\Sigma 2^{l_i}$ for the subcorpus & subcorpus.

So: How do we go from t. GPD₁ of .35–.36 to t. GPD₂ of (.33–.34)?? What is it that we are maximizing in t. GPD₂ ?? → perhaps the utility of t. predn! usually measured by $\propto -\frac{1}{pc}$.

90.00's
CIAP.

90.00

4/2/01 IDSIA **CIAP**

"Lotus Eater" as a Sink

:00.:56.40 : **One** problem of Very intelligent Reinforcement Machines finding way to do self-reinforcement. Juergen & Marcus suggest that if there is a large set of RTM's, w various internal constraints, the ones w. lack of constraint would "**Win**" — since they could keep on growing.

Also note: **To** do eternal self reinforcement, T machine must protect itself (& its power supply in particular) — so it cannot afford to be a complete "**L**otus Eater"!

4/2/01 IDSIA

Study at 5

.00: 89.40 : $\boxed{89.33–89.40}$ is t. closest I've come to expressing what kind of Goal I need for t. GPD$_1$ → GPD$_2$ output]— 89.40 is t. Goal; 89.33–.40 is derived

History of t. problem: 84.01 85.01 86.09, $\boxed{86.16–.20}$ 88.01 — 89.40 — 91.01 | 89.33–.40 is latest work

newly (?) is of interest!

↑ why t. ALP soln in BAKL is wrong

A reasonable Approach (Related to 86.16–.35). 88.39–.40, 89.01–.02, .33–.40; 91.00ff.

So GPD$_2$ looks at a prob. problem (and its Time limit) and outputs 89.33–.34.

The idea of 89.40 was that GPD$_2$ needs a Goal for its actions: "What is it trying to do?"

That t. preds should be "useful" seems like a good Goal. That utility is to minimize $\frac{T}{p_i}$

.11 ∴ we want to maximize p_i's of outputs so $\sum \frac{c_i}{p_{ci}} + \sum \frac{t_i}{p_{ci}}$ is minimum. This sum is [redacted] 'like' total soln. time.

T. Goal of all is certainly familiar! The reason one can't minimize it by making all $p_i = 1$, is that the p_i's have to be obtained in a logical way by summing over codes. The system could conceivably cheat by assigning corpus parts that biased towards t. p_{ci}'s being very high. Choosing corpus parts selectively, is only one way, t. system could introduce bias.

Perhaps I shouldn't be looking to t. "total soln time" as a goal (closely related to goodness of GPD$_2$): Perhaps just try to make GPD$_2$ as good as possl. —— At first glance, it would seem that there are more opportunities for "unwanted bias" in GPD$_1$'s Goal: [GPD$_1$'s goal is max prob. of data that has occurred.] — perhaps TM can easily cheat by not including data for which p_i values were bad! ← Maybe not — it has to include some data for all p_i that were used. (It can't eliminate p_i's).

92.01 – 100.40 is on talk at IDSIA

≈ (Also a short Summary of state of TM)

101.01 on Things left out of talk