N2498 Proposal:

1) See NN 02896 #2

2) " 03198 #1

3) " NN 11598 #5_6: Some Things I'd want RM able to do.

A): Many systems: ours is based on ALP:

Now many M. works.

We will discuss during.

Tsp's written not fully diffys:

1) hard to write.

2) didn't scale well.

3) didn't ~~work with~~. ~~use architecture well~~. Interrelate various areas well.

New Advances in ALP. — How they will fix these probs. & also help us use work of others.

98TH 58.10 ! listed Geen long Recurses.

7.03

N2798

Proposal for Research toward a very intelligent, general purpose machine.

Abstract: In the last decades, many systems have been proposed for learning, problem solving and learning. Very general kinds. We will ... a system based on ... Alg. Prob. ... to develop a machine of this type.

A naive machine ...

Used machine other systems to help design Progs.

and: enables us to use other systems.

These advances also make it possible to take advantage of the Broad variety of problem solving and learning techniques that have been developed by our researchers ...

We start with a naive machine having very little specific knowledge. It has, however, an initially simple method for solving problems and a very general system for learning. This machine is first given a simple problem ..

(Ch. 4g) → Proposal for Research toward a very intelligent, general purpose machine.

Abstract: In the last decades, a variety of systems have been proposed for very general problem solving and learning. Our own work in this area has been based on Algorithmic Probability — the description of data using short codes. We use short codings of data to gain both understanding and extrapolate the data.

We start with a naive machine, having little specific knowledge — only an initially simple technique for solving problems and a very general system for learning. This machine is first given ..... (4.01 — .2 4)

H3098 1

Proposal for Research toward very intelligent Machines.

See TM68

Proposal : Abstract: in reactant decade that have been.

1) Many steps indicate / Prob solving systems proposed.

2) We are now in AIP (give reasons or examples.

3) T. system studied : more machines, TSG, updating : (tell how we start w n'w machine

* build it up to be very "intelligent" using at. eg.

4) What done: new machines, TSG. updating

5) 3 important difficulties of TSGs written a) did not scale well to large, multiplot pages (methods) different Domains

b) The system wasn't able to integrate the segs from different domains

c) TSG's were very hard to write.

6) Recent advances in AI provide help solve each of these difficulties in a unified manner.

* techniques for dealing with each of all of these

difficulties in a unified manner.

Introduction: Explain each of the pts. made in the abstract.

For the most part, don't explain much detail, refer to Sol 86, 89 for details of most of it: But explain to reader to pt. at which he knows what the difficulty is.

Then outline the "Mixed Corpus Forms". Explain how used.... how they solve or help solve the difficulties mentioned.

→ Also explain how P.D.'s for DZ probs are updated: why Rein's impl. now.

→ Or: explain that this is a problem — I know how to solve it now.

Perhaps Give General picture of "state of the art" after Sol 89 + # 3 TSG's.

That I had intuitive ideas on how to update the P.D, but no general Piecewise/understanding of the process.

Refs to work by others on Incon (mg: 3 to 2

D 1398

Proposal: To Summarize

Ending. The Mixed Corpus Theorem seems to hold the key to the solution of several important problems in Machine learning:

1) The updating problem: What is the theoretically best way to use the latest solutions to general problems in sequence extrapolation, extrapolation of Kolmogorov costs, and resource-limited optimization problems to modify the common conditional probability distribution that will guide the search for solutions to all three kinds of problems in the future?

2) How can we best generalize information on success or failure of various problem solving techniques (used within problems) so we can use them for solving new problems)? Generalization for solving

3) How can we make best use of the large & vast, disparate body of existing Machine learning research to facilitate the writing of effective training sequences for a single, unified system?

4) When we discover new theorems or concepts, how shall we best assign conditional probabilities to them, so that they can be most economically & relevantly retrieved in attempts to solve new problems?

The application of the Mixed Corpus Theorem to these problems, will be the main thrust of proposed research.

In each of these problems, is the first step is to find the optimum theoretical solution; the Next, to find practical, approximate solutions consistent with available computational resources.

(SN) It is possible that Gauss is satisfied w. the C.V. I sent & needs no proposal. Until I hear from him: Best work on MIT talk(s).

Rite now, 4 most important things seem to be:

→ 1) Get stuff about 2 Bernoulli Grammars; 2-14-1 straightened out.

2) The stuff on GA v.s. L-sys (82.01 ..... 86.40) seems very important. It makes the developed details & actions of the "Mind Complex Thm." & my general approach to TM. : Also it could be instrumental in getting people to work on this! To show how GA could be improved, to give a detailed understanding of GA., etc.

3) Perhaps write up EXACTLY what MCT says & outline of proof! I have to keep it in mind: I'm forgetting it!

4) Perhaps list important ideas, Thms, Refs, conjectured/proved: I call where they are or try to find them.

5) Put TM stuff in chron order in 4 bay & 2 bay filing Cabinets: (6 bays) Notenow I have 6 3 bays w. supports for file folders. Put in most recent stuff first.

6) Put Empt Theoretical SM stuff in 4 year access storage; Less important & much lower access (Files) ← clearly labled piles on bookcases.

7) Have 4 folder at my access w/ index of files in Cabinets: Maybe copy in each cabinet!

(Negative probabilities)

.01 9° What would be ideal course of Action if I had to get TM done in
.02 Q - 6 mos, 3 yrs, 2 yrs, 1 yr? The Horizon Question.
In any Case, I would make a "Pert" diagram showing what had to be done on each of the parts. Try to talk others into working on the parts.
Some Impt. ideas: → Also see 96.38 (#11)
.07 (1) T. MCT is very impt. Write a clear statement of it, w. critical, illuminating Examples.
.08 (2) Look at Hoare's Most Advanced work. Make sure that my System can do everything that he can (or can't). → (8)
(3) The Z-141 problem does seem impt. Write it up so that others can interact. → (7)
(4) The Stein effect } Needs better Review (or Review of Reviews)! Stein
(5) The Soy effect. } Needs Review.
(6) TSQ design: I like of using a seq. of "definitions" as a way to teach Algebra (also regular problems). For Definitions long, need to have to examples, I'd like it to be possible to convey to TM a definition in its usual sense, that is less inductive than a seq. of examples. Use, probably, a directed More useful "unsupervised" approach
(7) Re. TM Eng. English: Ability to discern CFG's would be useful: I perhaps would want to develop idea of utility (≡ pct) of doing an agent or a "relation" or a Bag-relation (that has wts. for each instance) (case counts)
.21 SN I may be able to use Negative Probs, if I can have neg wts in Bayes or R-Bag (Recursive Bayes). (≡ R Bay.) Also in SGA, I
.23 might want to have negative Δ correspond to neg. wts → neg probs. → (28)
(8) How impt it is to follow/understand/explain/replicate + Regular GA & SGA, is unclear. SGA does seem to be a good, potentially important OZ solving method. If so, I may want to show how ordinary GA is well understood as an approach of SGA.
.28 .33 Re: 21 It would be possible to model the Z-141 problem w. $\delta < 0$, by a γ that has its $\delta \le 0$ & its prob ≤ 0. Its "case counts" would be subtracted from the case counts of χ (due to normal "random" enhancement δ). Troubles, negative case counts for γ in χ over all cases should occasionally occur, if use this model. They do not seem to. Notice the case where $\delta = -\gamma$ (γ w. zero true)
.33 If of much interest & the approach .28 is, in this case likely to have $f_\chi < 0$ hold the time! → 100.0
A Good Approach to the problems of .07 ff: For each serious problem/hobHancock, have a very clear statement/discussion of the critical problem, so I can keep it in mind & come back to it occasionally: but most impt. of all by keeping it in mind, I am more likely to recognize a solution when it appears in some other context.
I could list the unsolved problems on my works (also solved problems), so that others might suggest solns.

→ 98.0

.01 92.40 : (9) The idea of Operational P.D. : I've written some on this, but I don't yet think I have a good definition. One thing I want is : what does it mean to "Improve t. P.D."? Say I associate a P.D. w. a corpus (presumably a Mixed corpus) — then I can have a partial ordering of the P.D.'s w.r.t that corpus. Or a better than relation if it gives back pc for corpus the same cc or some pc for corpus w. lower cc.

In some situations, the relative utilities of ↑ of pc & ↓ of cc can be evaluated — in which case we have a complete ordering of OPD's : MAYBE A CRITICAL IDEA! Hvr .10–.40 complicate this idea

.10 Actually the utility of an ↑ in pc obt. P.D. may depend on just what part of t. P.D. habits pc ↑. For some O.Z. pairs I think t. trade off in Δcc vs Δcc may be quite clear. For other applics, perhaps quite distnt a/o not at all clear. One kind of trade-off that I'd consider

.14 in t. past, was between TM_1 & TM_2 (producer P.D. v.s. consumer P.D.) & I

.15 considered t. "50% soln.". In t. present case there can be much more than 2 kinds of P.D. — (maybe a Continuum of them) so Dividing t. available cc betw. them openly would seem to be not V.G.

In my Criticism of Schmidhuber, I was concerned that he was trying

.19 to Eliminate several params that I find important. One of these was t. "Horizon"

.20 — i.e. Time limit during which t. Goal of TM was to be achieved.

(.14, .15) He did eliminate problem that t. 50% soln was an attempt to solve. Essentially, what he did, was to put all info about desires into the "feedback function" — t. external Reinforcement level.

.24 In .10 it seems that t. "question of not t. Utility of an ↑ in pc of a P.D. depends on what part of t. P.D. it is" May be a Slogan!

.26 It enables us to put information in that we as "Operators" or "Clients" of TM, want to put in. It's like t. "Universal appeal" sought (ignorant) researchers want an appeal that is "maximally objective", "Universal", t. same for all". Others realize that t. arbitrariness of t. UMS is a way we can insert t. needed appeal into.

Similarly in .24 : Turgon inserts t. info via t. Reinforcement Function. How this is done, is unclear. Whether it can always be done is unclear.

In .26 we (presumably) insert this info in a more direct, conscious, way.

So in .10, it seems clear that "How good a P.D. is" depends somewhat

respect of it one is interested in. Tho we do have t. wts assoc. w. problems

solved in t. past, perhaps we can use this as a first approx. — we should

be able to insert info in t. system that tells (it t. relative utilities to t. user

of various aspects of t. P.D. (95.0)

0.16.99 : TM OPD (operational Probly Distribn).
PLAN

Guesshelp
Guesshlp.txt 2KB — Help for Guessgram
Random.txt 2KB

Refer to "Help for Guessgram" — Why write a help in another system

(9) cont

01: 99.90: This discn of 9t.al-to does look like a soln. of one part of t. OPD problem. This is a part that I hadn't realized was part of OPD problem.

Hvr, even if one had this soln to this problem, a critical OPD problem remains: How do we order OPD's?

.05 Essentially an OPD is of the form $P(x, y, cc)$ | It is a pc. (conditional) prob. dist. where its first argument is the condition, the second arg y (string) is what we want the cond. prob of; the third arg cc is the knowledge computing cost. We ask for the most Utility version of $P_1(\cdot,\cdot)$ w.r.t. a given corpus.

Hvr, I'm not sure if it is just what I want (Indeed, if it is meaningful).

Anyway, it suggests that we use least criterion: Utility is $\frac{P}{cc}$

.11 Maybe Expected value of $\frac{P}{cc}$ taken wrt t. three (elapsed):

.11 would give t. Expected cc needed to solve a problem (?)

True for Inv. probs. For Analysis of OZ problems: Look at MCT: The boundaries t. OZ's INV problems (of P.D.) Domain

So, t. Great BREAKTHRU is t. realization that t. Utility of a (P.D.) can have many parameters, depending on t. needs of t USER

.16 → Another Corrolary (Maybe): That t. soln. for Prob. that we "solved" w. t. 50% soln. is also of this sort: I. "Horizon problem" (9.4.19 - .20) Evaluating seems to be of this sort, & t. share of cc being TM1 & TM2 (Discount short term goals v.s. General improvement of t. P.D.) seems to be closely related to t. "Horizon" problem. (35)

.21 Another Problem w. a Similar Answer is: "What is t. Best path to Most Knowledge". What is this "Knowledge" is in Math, Sci or Engrg, t. "best path" depends on our "goals" or just What we like to do, to discover, to experience.

.27 Another Aspect of t. "50% Soln" Problem: It assumed that I would always spend a fraction, ft, of my available cc, on "improving t. P.D.". Actually, this is not so: In general, I'll want to spend different amounts of cc on different aspects of t. P.D. Similarly, I'd want to spend different amounts of cc on different INV & OZ probs. I. ideas of .16 seems to address these ideas.

9.18.99 In many, if not all, of t. more recent cases in which info from USR is needed for optimum Soln, "Default solns" (like 50% soln.) exist that are not needed. They are, more of necessity, sub-optimal. What this may mean, is that we can use "default" solns, to start off, then spend time (later, "tuning" t. params" to our preferences → (16) 106.30

(10) Unsolved problem re. Probability radius. While AIP defines a probability over a corpus and an enc, in terms of a limit as cc → ∞ — it solves this (under exists) t. radius of a prob, but, it is not clearly defined.

Halphlp1.hlp
Frac029.gif
Fractnl version 19.2
fractnlhlp Mathedoc.
Punov
3.2G
2.8 (486/50
2.5 E-mach
2.5 P200
Maybe 1 more 2.5G.
$120 for 10.2G
130M
170M*
170M

96.01

.01 35.40 (10) (cont) — yet it's mainly what we use in applics. One possib. default soln, & to consider only primitive recursive functions in implementing A&P. Since PRF's are countable (recursively enumerable); pairs, triplets ··· are also recursively enumerable. On t. other hand: partial rec. functs (prf's) are rec. enum as are pairs of such functions. So maybe using prf's doesn't solve anything.

One "obvious" ~~soln that would be to use same~~ CB for each of the two AL Approxns, then as t. CB → ∞: $\lim_{t\to\infty} \frac{f_1(t)}{f_2(t)} = \frac{\lim_{T\to\infty} f_1(t)}{\lim_{T\to\infty} f_2(t)}$; unless t. denom is zero. (It never is in full ALP)

.08 So what's wrong w. this? Well, in practical calcs, one might use different heuristics for numerator & denom., so w. any finite T, one could get very biased results.

Perhaps what I was concerned w. was .08 is: that one could inadvertently bias t. results considerably by using different heuristics on t. numerator & denominator. However, if t. heuristics were fixed apriori, then nomatter how t. Num & denom. etc grew w. T, as long as both → ∞, t. ratio would → t. proper limit. With any finite T, however, one could use much more cc on numerator, say, so t. ratio would be very large.

One might do this in "Rhetorical analysis" of a system of "Political Significance".

Is there a way to avoid this when one wants objective results?

But I think both Fritanyi & Levin & Gacs felt that there was something "wrong" with prob. ratio approxns, as opposed to prob. approxns. (Maybe Laum is less in pain since now (1+ 29.00)

In t. case of prob, one obtains a monotone seq. of approxns to t. limit. In t. case of ratios, the seq. of ratios was not monotonic, so one had no observable bound on t. limit as one of successive approxns. → See 105.01, 106.01 "On T. Dimensions of Uncertainty"

? not in this section of Notes

SN While using Lsrch for INV problems seems reasonable, & that it should be an optimum if one managed to "put all heuristic info into t. P.D." seems reasonable; corresponding statements for OZ problems don't seem so clear. Earlier, I had t. idea that I'd start out w. a few V.G. Hillclimbing methods that would be "hand made" (i.e. instructed by me) — then, later, when TM had enough experience w. this sort of thing, it would become to devise better & better methods.

SSN "Anytime methods" are M.C. routines. As an approxn. of a soln to an OZ problem of known CB, one could use an "Anytime soln" if t. CB were large enough. This suggests that one could just as well consider all "Anytime Algms", and spend on each a time share of its P.S. The P.S.'s would vary w. t. experience of t. TM — I'd have to expand MCT to include "anytime algms" as well as "OZ algms".

Of these "problems" of 92.07 ff; right now ③ ⑦ t. Z(4) diffly & t. extensions of Z(4) to CFG & w rests, seems most imp't.

.38 (11) I must write out proof of conv & rate of conv. of convex w. finite set of finite objects. This is proof of methods in Sol 99 Paper (related to MCT: 92.07

..OL: 96.42: (12) 105.01 On t. Dimensions of Uncertainty — I think 105.03 is t. practical "soln". It may lack theoretical justification, but it's t. way humans would & have to solve t. problem! T. initial problem was that t. uncertainty in t. prob estimate made by ALP was not a probability d.f. & was impossible to manage (deal w.) in any rigorous way. Hvr, by treating it as a prob distribution & using t. same kind of data to estimate it, & then we use them in MCT, we can probably get useful (tho not theoretically just-justifiable) solns. These will be workable "Engineering" Solns.

(13) As general Modis Operandi: I want to have each impt problem: both solved & unsolved, in clearly stated, written form, so I can at all times, know just what needs to be done — what are t. bottlenecks ... So I will recognize a hint at a soln when I encounter it. Also, I may want to put these on my website, to get comments from others.

However NOTE: I haven't always done well with "comments from others": They tend to be very time consuming! Also, I often don't reply to actually interesting Q's, comments.

On t. other hand, f.b. from others has been useful in finding errors, & clarification. e.g. Manos, Wolff, Shmidhii, Lavin, Grace, Vitanyi)

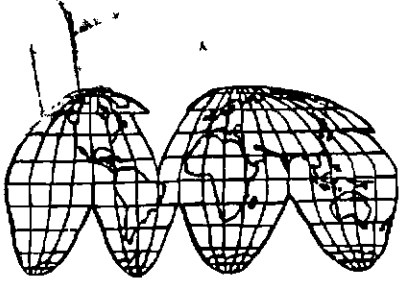
(14) A dirty little "writing Revises" is that I usually don't finish them: I get involved w. a new approach to solve critical problems. I could do a time limit: (e.g. < 1 day or < 4 hrs) for each idea, & simply make a list of "new ideas" (rather than work on them) as they arise). 108.19 ff has 50 v.G ideas on How to write a Review.

(15) Re: 92.02: — out 1 or 2 yr horizon: Best thing to do would be to get a not-so-dirty form of ZRH & try it out on various problems in various domains.

(16) One idea of OPD: (95.36 from 9) That a OPD gives a prob output w. input: (condition, cc) Simple example: A Monte Carlo device... we put in input (x, all ... uncond prob) We get 1 output (0 or 1), In time T, (cc ≤ T) we get T outputs & so we get an estimate & expected error of p.d. After N (in not output) we get P_0 & P_1 w. certain expected errors: as we ↑ cc (≤ T) the errors usually ↓.

(9) (94.01 ff) Discusses criteria for "Improving the OPD": When is one OPD "better" than Another?

(17) **Serious Bottleneck (?)** In situations where I want to use (say) t. Pd is not of the form $\sum p_i \leq 1$ i.e. the events are not mutually exclusive. (See Bulg 2.24 ff for discussion, at least 3.40.) It may well be that (say) is usually impossible! (Some discussion in N.Tw Notes: N 30, 99 Edoos) I think Bulg 9.01 ff. solves some of these. (cc, Pd is always computable). That Pd's can still be a problem, & the WON approach (What to Work On Next) seems relevant.



Plovdiv 2000 Bulgaria

Address of the Organizing Committee:
Professor Drumi Bainov
P.O. Box 45,
1504 Sofia, Bulgaria
E-mail: dbainov@mbox.pharmfac.acad.bg
Fax: +359 2 9879874
Tel: +359 2 437343

Dear Professor Solomonoff,

The Organizing Committee of the Ninth International Colloquium on Numerical Analysis and Computer Science with Applications kindly invites you to deliver a one hour invited lecture at the Colloquium which will take place in Plovdiv, Bulgaria, August 12 – 17, 2000.

The work of the Colloquium will proceed in the following sections:

1. Acceleration of convergence, 2. Numerical simulation, 3. Numerical approximation, 4. Numerical methods in complex analysis, 5. Numerical methods in linear algebra, 6. Interval arithmetic, 7. Numerical algebraic or transcendental equations, 8. Mathematical programming, optimization and variational techniques, 9. Numerical analysis for ordinary differential equations, 10. Numerical analysis for partial differential equations, 11. Computer arithmetic and numerical analysis, 12. Computer aspects of numerical algorithms, 13. Parallel and distributed algorithms, 14. Concurrent and parallel computations, 15. Computer networks, 16. Discrete mathematics in relation to computer science, 17. Computer aided design, 18. Theory of data, 19. Programming, 20. Image processing, 21. Pattern recognition, 22. Communication systems, 23. Information systems, 24. Manufacturing systems, 25. Data base, 26. Software technology, 27. Software engineering, 28. Applications in mechanics, physics, chemistry, biology, technology, and economics.

If you kindly accept to participate in the work of the Colloquium, please let us know not later than March 1, 2000, together with the number of accompanying persons. Please, send us by the same date an abstract of your lecture written in English and not exceeding *one typewritten page*. It should be in a camera-ready form with the size of the text area – 17x24 cm. The text should be typeset using TEX, *greater than or equal to 12pt letter size*, on high-quality white paper by means of laser printer. The abstract should be arranged as follows: Title, name(s) of the author(s), full mailing address(es), Keywords, MOS(AMS) Subject Classification, the text of the abstract. The official language of the Colloquium will be English.

Provided that you confirm your participation in the work of the Colloquium with an invited lecture, second announcement will be sent by May 1, 2000, with the forthcoming information about the Colloquium.

The registration fee is 90 USD. THE LODGING AND FOOD EXPENSES WILL BE ON YOUR ACCOUNT.

During the Colloquium you will get acquainted with the rich archaeology of the city of Plovdiv, as well as excursions will be organized to the Bachkovo Monastery, Pamporovo (a mountain resort) and Hissar.

With profound respect,

Drumi Bainov
Chairman of the
Organizing Committee

International Journal of Applied Mathematics

Editor-in-Chief: Virginia Kiryakova
Institute of Mathematics & Informatics, Bulgarian Academy of Sciences, Sofia 1090, Bulgaria
Managing Editor: Emil Minchev
Medical University of Sofia, P.O.Box 45, Sofia 1504, Bulgaria
Secretary of the Editorial Board: Drumi Bainov
(President of the Publishing House "Academic Publications") Medical University of Sofia,
P.O.Box 45, Sofia 1504, Bulgaria
Fax: 00359-2-430-156 and 00359-2-987-9874 E-mail:
dbainov@mbox.pharmfac.acad.bg

R. P. Agarwal (Singapore) · J. Eisenfeld (USA) · F. Mainardi (Italy) · S. Sathananthan (USA)
N. U. Ahmed (Canada) · P. Eloe (USA) · C. Micchelli (USA) · N. Shigesada (Japan)
M. Amamiya (Japan) · K. Farahmand (UK) · E. Minchev (Bulgaria) · R. Shivaji (USA)
G. Anastassiou (USA) · A.S. Fraenkel (Israel) · S. Miyano (Japan) · D.D. Siljak (USA)
P. Antonelli (Canada) · E.A. Galperin (Canada) · D. Motreanu (Romania) · K. Sohraby (USA)
D. Bainov (Bulgaria) · N. Hayek (Spain) · A.D. Myshkis (Russia) · H.M. Srivastava (Canada)
J. Bramble (USA) · J. Henderson (USA) · G. Nuernberger (Germany) · I. Stamova (Bulgaria)
C. Y. Chan (USA) · M. Ito (Japan) · D.K. Palagachev (Bulgaria) · R.L. Stens (Germany)
K.W. Church (USA) · R. Kalaba (USA) · J. Pecaric (Croatia) · B.A. Trakhtenbrot (Israel)
D. Cutkosky (USA) · S.L. Kalla (Kuwait) · V. Petrov (Bulgaria) · D. Trigiante (Italy)
L. Debnath (USA) · D. Kannan (USA) · P. Popivanov (Bulgaria) · A.S. Vatsala (USA)
F. Deutsch (USA) · H. Kawarada (Japan) · A.G. Ramm (USA) · R.U. Verma (USA)
R. W. Dickey (USA) · W.A. Kirk (USA) · T.M. Rassias (Greece) · T.M. Witten (USA)
A. Dishliev (Bulgaria) · V. Kiryakova (Bulgaria) · M. Saigo (Japan) · M.D.F. Wong (USA)
J. Dongarra (USA) · G.S. Ladde (USA) · S. Saitoh (Japan) · A. Zayed (USA)

Aims and Scope

The journal will publish carefully selected original research papers on: combinatorics, design and configurations, graph theory, fractals, lattices, ordered algebraic structures, real functions, measure and integration, functions of complex variable, potential theory, special functions, dynamical systems in their broadest sense (covering ODEs, all kinds of PDEs, FDES, difference equations, functional equations), approximation and expansion, integral transforms, integral equations, fractional calculus, operator theory, calculus of variations, optimal control, optimization, applied differential geometry, probability theory and stochastic processes, statistics, numerical analysis, computer science, mechanics of particle and systems, mechanics of solids, fluid mechanics, classical thermodynamics, mathematical methods in economics, operations research, programming, mathematical biology, systems theory, information and communication, circuits.

Call for Papers

Authors are cordially invited to submit papers in triplicate to the Secretary of the Editorial Board: Drumi Bainov or to a member of the Editorial Board. Manuscripts submitted to this journal will be considered for publication with the understanding that the same work has not been published and is not under consideration for publication elsewhere. The papers accepted for publication will be processed further toward publication only after the authors pay the page charge at the rate of US $ 10 per page, to be paid to the Secretary of the Editorial Board.

Academic Publications

Application of Algorithmic Probability to Machine Learning

Ray J. Solomonoff

Visiting Professor, Computer Learning Research Center
Royal Holloway, University of London, U.K.
Mailing Address: P.O.B. 391887, Cambridge, MA. 02139, U.S.A.

Keywords: Machine learning, Algorithmic Probability, Pattern Recognition, Knowledge Representation, Levin's Search Procedure, Incremental Learning, Problem Solving, Adaptive Systems, Kolmogorov Complexity.

Mathematics Subject Classification (AMS): 60A05, 60A10, 60G25. 62M05. 62M20, 68T05, 68T10, 68T20, 68T30, 68T37, 94A45

Given a function F that maps strings to strings, and given a string s, to find a string x such that F(x)=s. Problems of this sort are called "Inversion Problems" - they are the P and NP problems of computational complexity theory.

In 1972, Leonid Levin discovered a general solution to such problems. It was "optimum" in the sense that if there existed a good technique to solve the problem, his method would be at worst, a constant k times slower than this technique. k would depend on F, but would be independent of the size of the problem string, s.

Later, it was possible to extend Levin's technique to solve Time Limited Optimization Problems. Since just about all problems in Mathematics, Science and Engineering are either Time Limited Optimization Problems or Inversion Problems, this would seem to be a powerful result indeed!

There were certain difficulties, however. Levin's search for problem solutions was guided by a Universal Probability Distribution. There are many such distributions, and any one of them will have small k (near optimum) for some problems, but impractically large k for most others.

This can be dealt with by choosing an appropriate probability distribution for each problem — a **Conditional** Universal Probability Distribution.

We show how to obtain such a Conditional Probability Distribution based on the problems the system has solved thus far. As the system is used for solving problems, it becomes better and better and is able to solve ever more difficult problems in acceptable times. Our "Mixed Corpus Theorem", tells how to update the Conditional Probability Distribution using not only problems that have been solved by the system, but any other kinds of information that we suspect may be useful in problem solving.

The system is able to accept information from any domain of knowledge and (if it is relevant) use it to solve problems in that or any other domain. It gives a rigorous theoretical basis for the "Unity of the Sciences".

Before the Mixed Corpus Theorem was devised, we were able to design systems for machine learning based on Levin's search algorithm. It was, however, **extremely** difficult to construct sequences of problems to train the system. Because The Mixed Corpus Theorem allows very general kinds of information to be used it becomes **much** easier to construct data sequences that are adequate for training.

Professor Drumi Bainov
P.O.Box 45,
1504 Sofia,Bulgaria

Feb 18 2000

Dear Professor Bainov:

Enclosed is the abstract of my invited lecture.

Most sincerely,

Ray Solomonoff

Professor Drumi Bainov
P.O. Box 45
1504 Sofia, Bulgaria

Feb 18, 2000
P.O.B. 391887
Cambridge, Mass 02139
U.S.A.

Dear Professor Bainov:

Enclosed is the abstract for my invited lecture.

There will be one person accompanying me.

What arrangements are being made for travel and lodging.

Are arrangements for travel and lodging expenses, ect.?

Most Sincerely,

Ray J. Solomonoff

d bainov@mbox.pharmfac.acad.bg Feb 18, 2000

Dear Professor Bainov:

Today I sent you by regular mail, ~~a copy of~~ the abstract of my invited lecture. In the event that you do not receive it in time, I am enclosing the TEX version of the abstract.

There will be one person accompanying me. What are the arrangements for travel and lodging expenses, ect.?

Later, Follows

Feb 19, 00

Banyan 6 ! In reply to Letter of 2 | RJS.018 2/18/00

Southern Exp.

2:30 PM Sunday 2/19/00

Dear Professor Ramaul?

Thank you for your prompt reply!

My wife and I, however, are uncertain as to -

It is unclear to my wife and I as to ~~whether you will be~~ who will be paying for our travel expenses.

From your letter, there was some ambiguity as to whether ~~the~~ (Per organizing committee) or, my wife and I will pay for our travel expenses. Please could you clarify this?

Also, what is Rs 250 USD for? We expect to stay in Bulgaria ~~about a month~~ for about a month and would like to arrange for longer term lodgings.

We await your reply & We look forward to hearing from you.

Most Sincerely

Ray Schumacher

In an earlier letter, you said that a portion of the lodging and food expenses would be "on my account". Is Rs 250 USD (Rs) for what? If so, might this not be reduced, since my wife and I will occupy a single hotel room?

F 5 bg Yallo
P6-P6 BL

Was sent to Royal Holloway for more maybe Jun 1, 00
May forwarded it to me
Rec'd. Jun 13, 00

NINTH INTERNATIONAL COLLOQUIUM ON NUMERICAL ANALYSIS AND COMPUTER SCIENCE WITH APPLICATIONS

PLOVDIV, BULGARIA, AUGUST 12-17, 2000

Second Announcement

Address of the Organizing Committee:
Professor Drumi Bainov,
P.O. Box 45, 1504 Sofia, Bulgaria
Tel.: +359-2-437-343
Fax: +359-2-987-9874
E-mail: dbainov@mbox.pharmfac.acad.bg

May, 2000

Dear Professor... Solomonoff, ...

The Organizing Committee of the Ninth International Colloquium on Numerical Analysis and Computer Science with Applications thank you for having agreed to deliver an invited lecture and send you the necessary information:

- The working language of the Colloquium will be English.
- The starting time of your lecture will be fixed after the registration.
- A volume with the abstracts will be published and each participant will obtain it on the registration.
- You are expected to arrive in Plovdiv on 12th of August.

The Colloquium will take place in the building of the Plovdiv Technical University, boulevard "St. Petersburg" 61. The registration office will work at the same building from 6.00 a.m. on 12th of August till 2.00 a.m. on 13th of August, and from 8.00 a.m. till 8.00 p.m. on 13th of August. There is no civil airport in Plovdiv. One can arrive from Sofia to Plovdiv by train. There are trains about any two hours. The participants will be met at the Sofia airport by members of the Organizing Committee who will organize the trip from the airport to the Sofia railway station if they arrive on August 12, 2000. The participants will be met at the railway station in Plovdiv and taken to the registration office by cars of the Organizing Committee if they arrive during the registration time. One can reach the registration office by bus No. 29 from the railway station to the "Lauta" stop.

- You will be accommodated in separated room with two beds of hotel type.
- The Organizing Committee will pay your registration fee of amount of 90 USD.
- The local expenses per person that include lodging and food for the total duration of Colloquium, welcome dinner and participation in the excursions will be 290 USD. The Organizing Committee will cover 40 USD of these expenses. So, the participant and each accompanying person must pay 250 USD (**FOR EACH PERSON - IN CASH ONLY**).

Waiting to see you in Plovdiv.

With profound respect,

Drumi Bainov
Chairman of the Organizing Committee

E sent e-mail or ordinary mail
Mail E sent 6/13/00

The era of reforms launched by Mikhail [redacted] Gorbachev in the [redacted] Soviet Union had a major impact on Bulgaria, inspiring greater demands for openness and democratization. The growth and greater aggressiveness of Bulgarian dissidents, a declining economic situation, and internal party rivalries led Zhivkov's colleagues to force his resignation on Nov. 10, 1989. He was soon subjected to intense criticism for corruption and abuse of power, placed under detention, and held while a state commission prepared formal charges against him.

End of party rule.

Zhivkov's successors endorsed a policy of openness, pluralism, and respect for law, halted repression of the ethnic Turks, and took the first steps toward separating the Bulgarian Communist Party from the state. Article One of the constitution, guaranteeing the party a monopoly of power, was repealed. After some shuffling of positions, Petar Mladenov was named head of state, Andrey Lukanov prime minister, and Alexander Lilov head of the Bulgarian Communist Party. In early 1990 the party held an extraordinary congress that enacted significant changes in party structure. To symbolize its break with the repressive policies of the past, the party renamed itself the Bulgarian Socialist Party (BSP).

In the meantime, dissident groups took advantage of the country's new freedoms to organize opposition political parties. Many of these joined the Union of [redacted] Democratic Forces (UDF), a coalition led by the sociologist Zheliu Zhelev. By the spring of 1990 the UDF and the socialists had agreed to free elections for a Grand National Assembly that would prepare a new constitution for the country. In these elections and subsequent runoffs, held June 10 and June 17, the socialists won a narrow majority of seats in the assembly. In July 1990 Mladenov resigned after unsuccessfully attempting to conceal the fact that he had recommended a military crackdown on protesters in late 1989. Because their majority was too small to allow them to govern alone, the socialists supported the election of the opposition leader Zhelev in August.

Bulg Soc Party (= formerly Communist party)

The National Assembly adopted a new constitution on July 12, 1991, which proclaimed Bulgaria as a parliamentary republic and promised citizens a broad range of freedoms and equality before the law. During the summer several parties withdrew from the UDF coalition, and those that remained split into two factions: UDF (liberals) and UDF (movement). In elections for the National Assembly held in October 1991, the UDF (movement) won a narrow majority of seats over the BSP, with the Movement for Rights and Freedoms (MRF; primarily representing the country's Turkish minority) gaining a small number of seats; no other party managed to score the required minimum percentage of the vote to qualify for participation in parliament. The UDF, with the support of the MRF, formed a government under Prime Minister Filip Dimitrov that had no socialist participation. Zhelev was returned to the presidency for a five-year term in elections held in January 1992.

The new government, faced with severe economic difficulties, began by seizing the property of the old Bulgarian Communist Party and its satellite organizations and by adopting a law regarding restitution of property to those from whom it had been taken by the [redacted] communist government. Economic reforms--including price liberalization and the privatization of industry--began as Bulgaria sought to meet the conditions outlined by Western nations as a prerequisite for extending economic aid. (J.D.B.)

For later developments in the history of Bulgaria, see the BRITANNICA BOOK OF THE YEAR.

New Search : Search Articles

Previous Section Home Page Help Turn Index Off Contents Next Section

Balkan States

Late communist rule.

Rise of Todor Zhivkov.

In the years after the April Plenum, the party leader, Zhivkov, became the dominant figure in political life. In 1962 he became prime minister, and he continued to hold the positions of head of state and head of party until 1989. An attempted putsch led by General Ivan Todorov-Gorunya in 1965 was easily put down, and Zhivkov consistently managed to purge or undercut party leaders regarded as potential rivals.

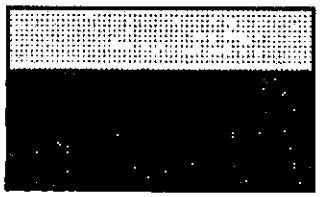
During the era of Zhivkov's ascendancy, Bulgaria modeled its domestic policies on those of the Soviet Union. Treaties linked Bulgaria's economic development with that of the Soviet Union through the end of the 20th century. Bulgaria gave the highest priority to participation in the modern scientific-technological revolution and pursued policies aimed at industrialization and the development of a population with the education and skills appropriate to an industrial state. In 1948 approximately 80 percent of the population drew their living from the soil. In 1988 the government reported that 19 percent of the labour force was engaged in agriculture, with the rest concentrated in industry and the service sector.

In its foreign relations after the 1950s, Bulgaria abandoned the isolationism that characterized the Chervenkov period. Although remaining steadfast in its commitments to the Warsaw Pact and the Council for Mutual Economic Assistance, Bulgaria improved relations with its Balkan neighbours, particularly Greece, and expanded its economic and cultural relations with most Western states. Relations with Yugoslavia remained strained, however, over the persistence of the Macedonian issue. In 1979 Bulgaria proposed a treaty with Yugoslavia that would guarantee the inviolability of the borders established after World War II; this proposal was rejected, however, because of Bulgaria's refusal to admit the existence of a distinct Macedonian nationality. From the Bulgarian point of view, such an admission would both fly in the face of historical reality and legitimize Yugoslav claims on the Pirin region.

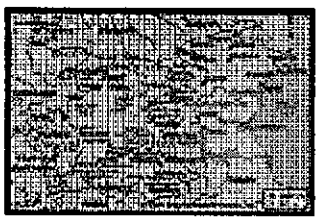
During the 1970s, in spite of the country's economic growth, serious concern developed over the low birth rate of the ethnic Bulgarian population. Numerous measures were adopted to encourage larger families, but without apparent effect. In late 1984 the government began a major campaign to "Bulgarize," or assimilate, the country's ethnic Turks. Measures aimed at the Turkish population, estimated to number approximately 800,000, included the abandonment of Turkish-language publications and radio broadcasts and the requirement that Turks adopt Bulgarian names. The ethnic Turkish population, however, resisted assimilation, and clashes with the authorities continued. During the spring and summer of 1989, when the government of Turkey offered to accept refugees from Bulgaria, more than 300,000 ethnic Turks fled or were driven from the country in a wave of violence.

New Search : Search Articles

Bulgaria,



[Flag]



officially REPUBLIC OF **BULGARIA**, Bulgarian REPUBLIKA BULGARIA, a Balkan country lying along the Black Sea in southeastern Europe. **Bulgaria** is separated from Romania by the Danube River along most of its northern border. The Black Sea coastline constitutes its eastern border, Greece and Turkey form the southern boundary, and Yugoslavia and Macedonia form the western boundary. Its capital is Sofia. Area 42,855 square miles (110,994 square km). Pop. (1992 est.) 8,473,000.[Map]

A brief treatment of **Bulgaria** follows. For full treatment, *see* Balkan States: *Bulgaria*.

For current history and for statistics on society and economy, *see* BRITANNICA BOOK OF THE YEAR.

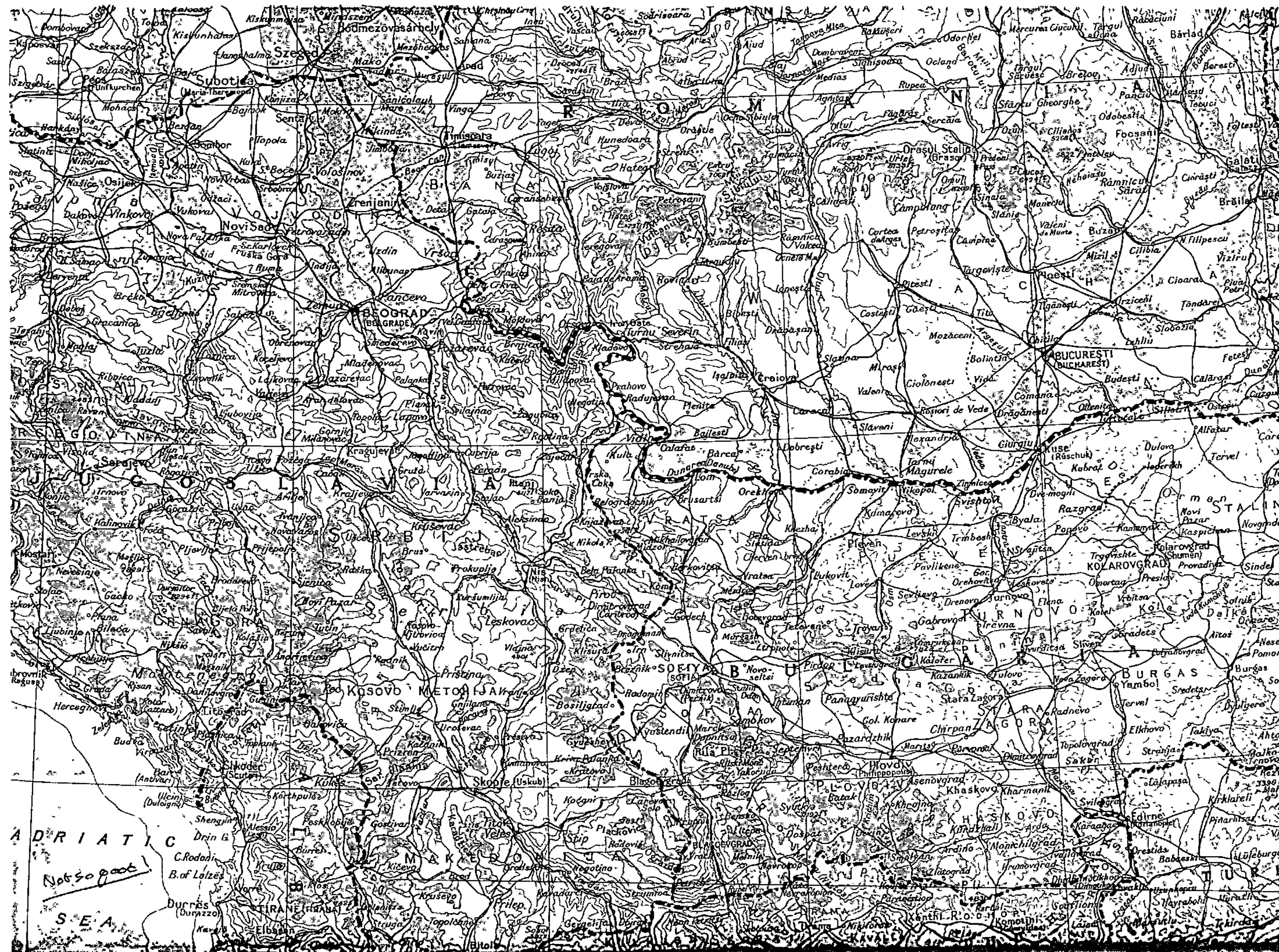
The land.

Three major physiographic regions define the Bulgarian landscape from north to south. The northernmost is the Danubian Plain, a fertile lowland with rolling hills, occupying nearly a third of the country. Two-thirds of the plain lies below 700 feet (210 m), and its entire area nowhere exceeds 2,000 feet (600 m) in elevation. Immediately south of the Danubian Plain lie the Balkan Mountains (Stara Planin, or "Old Mountains"), which average 2,368 feet (722 m) in elevation. **Bulgaria**'s third major region, the Rila-Rhodope massif, is separated from the Balkan Mountains by the Thracian Plain, or Rumelian Basin. Generally higher and more rugged (with many snowfields and lakes of glacial origin) than the northern ranges, the Rila-Rhodope massif has the country's highest mountain, Musala Peak, at 9,596 feet (2,925 m), and some of southern Europe's most scenic countryside. While less extensive in area than the three major regions, **Bulgaria**'s Black Sea coast, with its sandy beaches and harbours at Varna and Burgas, is among eastern Europe's favourite resort areas.

Bulgaria has two major drainage systems. The Black Sea receives more than half of the country's runoff from the tributaries of the Danube in the north (such as the Iskur and the Yantra rivers) and from the direct discharge of other rivers in the east. The remainder empties southward into

Not so good

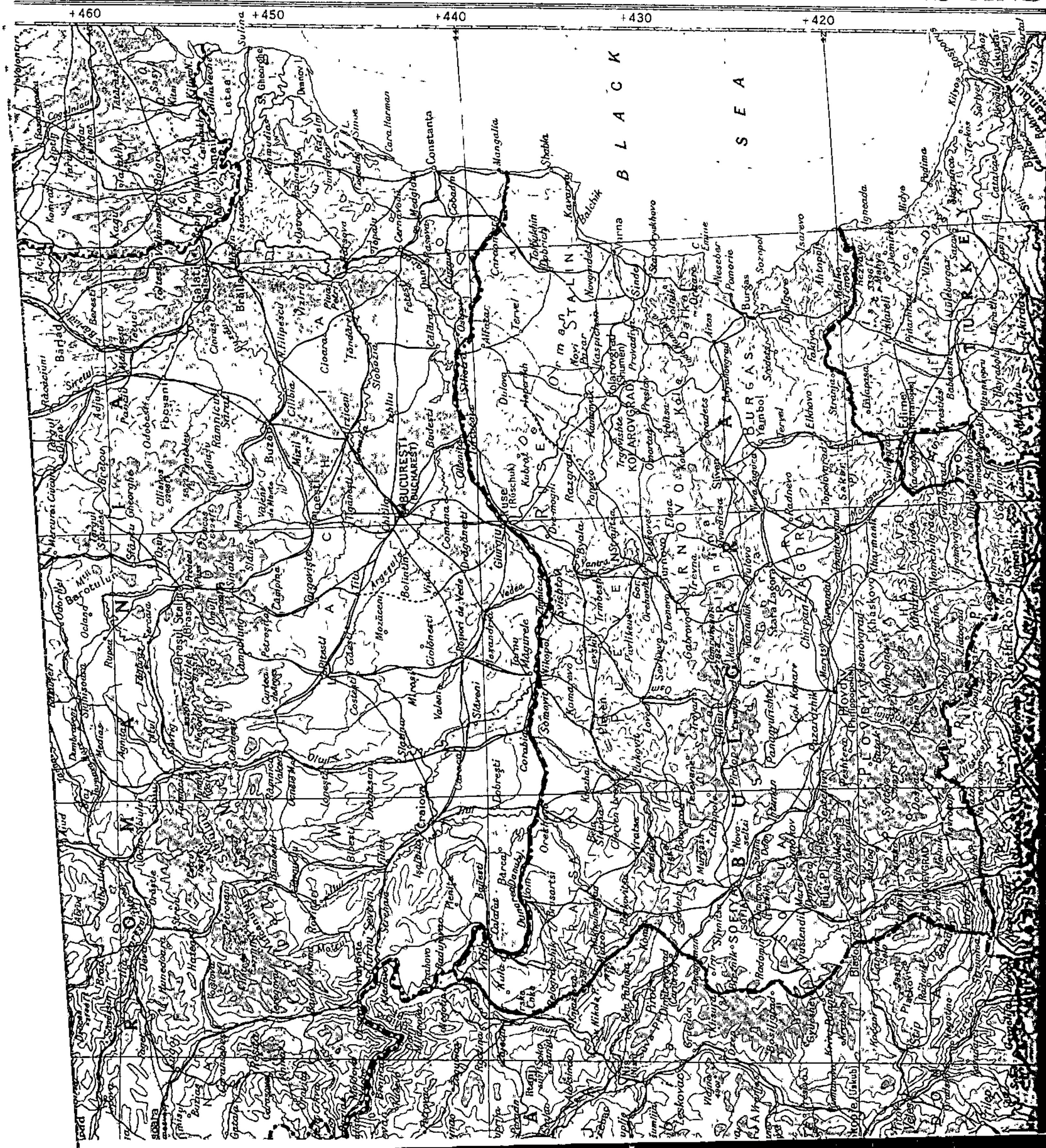
+460
+450
+440
+430
+420
L00
L06
L12
L18
L24
L30
L36
L42
L48
L54
BLACK SEA
BUCURESTI (BUCHAREST)
Constanta
Braila
Galati
Kishinev
Sulina
Varna
Burgas
Ruse
Craiova
Sibiu
Brasov
Cluj
Oradea
Timisoara
BEOGRAD (BELGRADE)
SOFIA
Ploesti
Focsani
Bacau
Roman
Stalin
Kolarovgrad
Stara Zagora
Plovdiv
Vidin
Pleven
Turnu Severin
Arad
Sofia
Nis

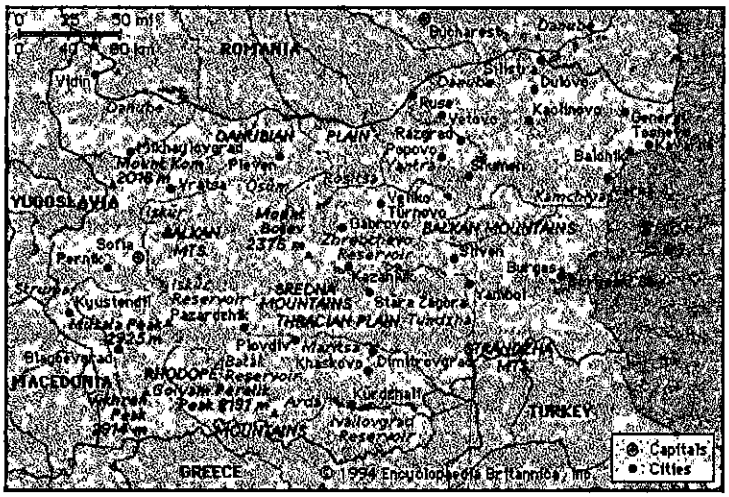


ROMANIA
IUGOSLAVIA
BULGARIA
BEOGRAD
(BELGRADE)
BUCURESTI
(BUCHAREST)
SOFIYA
(SOFIA)
ADRIATIC
SEA
TRANSILVANIA
BANAT
VOJVODINA
SRBIJA
CRNAGORA
MAKEDONIA
KOSOVO METOHIJA
VRATSA
PLEVEN
TURNOVO
STARA ZAGORA
PLOVDIV
KHASKOVO
BURGAS
RUSE
STALIN
KOLAROVGRAD
Szeged
Subotica
Hódmezővásárhely
Novi Sad
Zrenjanin
Timişoara
Arad
Pančevo
Osijek
Vinkovci
Vukovar
Sombor
Senta
Sarajevo
Mostar
Kragujevac
Kruševac
Niš
Leskovac
Skopje (Uskub)
Priština
Prizren
Peć
Shkodër
(Scutari)
Tiranë
Durrës
(Durazzo)
Elbasan
Bitolj
Prilep
Veles
Štip
Craiova
Turnu Severin
Vidin
Lom
Pleven
Vratsa
Sofiya
Plovdiv
Stara Zagora
Burgas
Yambol
Sliven
Kazanlŭk
Haskovo
Khaskovo
Kŭrdzhali
Edirne
Giurgiu
Ruse
(Rüschuk)
Razgrad
Dulovo
Ploeşti
Piteşti
Tîrgovişte
Sibiu
Braşov
Oraşul Stalin
Focşani
Galaţi
Brăila
Buzău
Bârlad
Sighişoara
Deva
Hunedoara
Petroşani
Alexandria
Roşiori de Vede
Călăraşi
Caracal
Slatina
Dunarea (Danube)
Kozani
Not so good!

Good

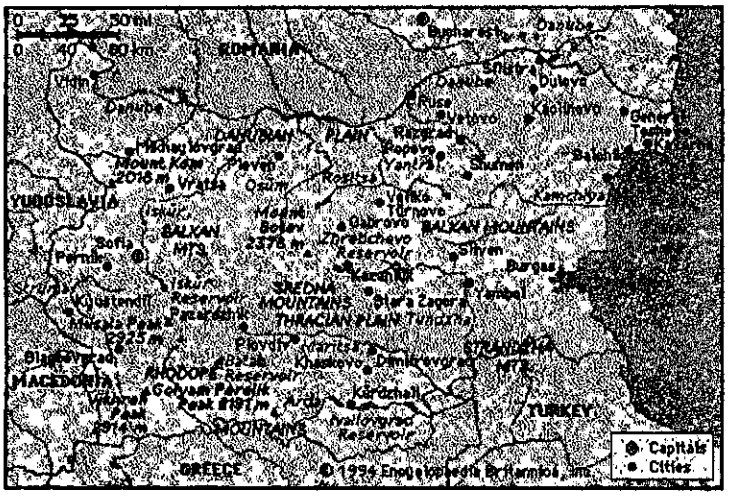
THE BALKANS AND

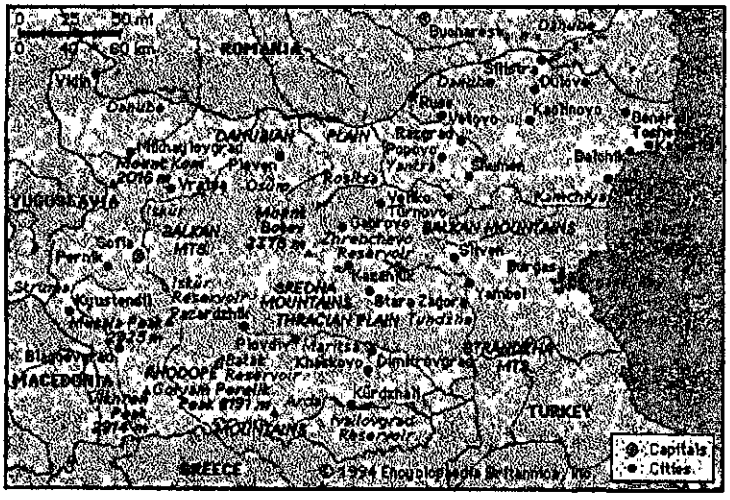




population ~ 70% urban : ≈8.5M (1992)

1993: Sofia : 1.1 M
Plovdiv : 345k
Varna : 307k sea port.
Burgas : 200k sea port.
Ruse 170k on Danube.


0 25 50 mi
0 40 80 km
BULGARIA
Bucharest
Danube
Silistra
Dulovo
Ruse
Kaolinovo
Shumen
Balchik
DANUBIAN PLAIN
Mount Kom 2018 m
Pleven
Vratsa
Osam
Rositsa
Veliko Turnovo
Kamchiya
YUGOSLAVIA
Sofia
BALKAN MTS.
Iskur Reservoir
Pancharevo
Iskar Gorge
Pazardzhik
Plovdiv
Karnobat
Sliven
Burgas
SREDNA MOUNTAINS
Stara Zagora
THRACIAN PLAIN
Tundzha
Dimitrovgrad
Khaskovo
Dzhebel
Kardzhali
RHODOPE MOUNTAINS
Gotsam Perelik
Peak 8191 m
Peak 8914 m
Arda
Kurdzhali Reservoir
STRANDZHA MTS.
TURKEY
GREECE
Capitals
Cities
© 1994 Environmental Research, Inc.


Bucharest
Danube
Silistra
Vidin
Danube
Dulovo
Kaspichan
Danube
DANUBIAN PLAIN
Razgrad
Balchik
Pleven
Popovo
Shumen
Yantra
Rositsa
Yablo
Kamchiya
YUGOSLAVIA
Vrashka
Dzhebel
Tarnovo
Mogila
Bokov
BALKAN MOUNTAINS
Zhurebchevo
Reservoir
Sofia
BALKAN MTS.
Pernik
Silven
Kazanlik
Yambol
Stara Zagora
Iskar Reservoir
Pazardzhik
SREDNA MOUNTAINS
THRACIAN PLAIN
Plovdiv
Haskovo
Dimitrovgrad
STRANDZHA MTS.
Kurdzhali
AHKOOPE Reservoir
MACEDONIA
Gotsyam Perelik
Pirin
Peak
Arda
Reservoir
RHODOPE MOUNTAINS
Ivaylovgrad
Reservoir
TURKEY
GREECE
Capitals
Cities

From dbainov@mbox.pharmfac.acad.bg Tue Nov 9 11:34:51 1999
Return-Path: <dbainov@mbox.pharmfac.acad.bg>
Received: from mbox.pharmfac.acad.bg (mbox.pharmfac.acad.bg [194.141.44.194])
by world.std.com (8.9.3/8.9.3) with ESMTP id LAA15563
for <rjs@world.std.com>; Tue, 9 Nov 1999 11:32:14 -0500 (EST)
Received: from dbainov (194.141.44.247) by mbox.pharmfac.acad.bg
with SMTP (Eudora Internet Mail Server 1.2); Tue, 9 Nov 1999 18:38:35 +0300
Message-Id: <3.0.5.32.19991109182722.007c8980@mbox.pharmfac.acad.bg>
X-Sender: dbainov@mbox.pharmfac.acad.bg
X-Mailer: QUALCOMM Windows Eudora Light Version 3.0.5 (32)
X-Priority: 1 (Highest)
Date: Tue, 09 Nov 1999 18:27:22 +0100
To: rjs@world.std.com
From: Drumi Bainov <dbainov@mbox.pharmfac.acad.bg>
Subject: Ninth International Colloquium on Numerical Analysis and
Computer Sciences with Applications, Plovdiv 12-17 August 2000
Mime-Version: 1.0
Content-Type: text/plain; charset="us-ascii"
Status: O

To: Professor Dr. Ray Solomonoff

Professor Drumi Bainov
P.O. Box 45,
Sofia 1504, Bulg
aria

Tel: +3592437343

Fax: + (359
) 29879874

E-mail: dbainov@mbox.pharmfac.ac
ad.bg

Dear Professor Solomonoff,

The Organizing Committee of the Ninth International Colloquium on Numerical Analysis
and Computer Sciences with Applications, Plovdiv 12-17 August 2000 kindly invite you to take
part in the work of the Colloquium with an invited lecture.

The work of the Colloquium will proceed in the following sessions:

1. Acceleration of convergence.
2. Numerical simulation.
3. Numerical approximation.
4. Numerical methods in complex analysis.
5. Numerical methods in linear algebra.
6. Ynterval arithmetic.
7. Numerical algebraic or transcendental equations.
8. Mathematical programming, optimization and variational techniques.
9. Numerical analysis for ordinary differential equations.
10. Numerical analysis for partial differential equations.
11. Computer arithmetic and numerical analysis.
12. Computer aspects of numerical algorithms.
13. Parallel and distributed algorithms.
14. Concurrent and parallel computations.
15. Computer networks.
16. Discrete mathematics in relation to computer science.
17. Computer aided design.

18. Theory of data.
19. Programming.
20. Image processing.
21. Pattern recognition.
22. Comunication systems.
23. Information systems.
24. Manufacturing systems.
25. Data base.
26. Software technologies.
27. Software engineering.
28. Applications in mechanics, physics, chemistry, biology, technology and economics.

If you kindly agree, please, send me your full mailing address in order to mail you the first announcement and invitation for participation.

Inform us please, in what kind of Colloquium is your field every time when you communicate with our office by e-mail since we organize two colloquia by the same time in Plovdiv: the first (12-17 of August) is the Colloquium on Numerical Analysis and Computer Sciences with Applications and the second (18-23 of August) is the Colloquium on Differential Equations.

In case of failure to connect with Professor Bainov's e-mail please, use the e-mail of the Secretary of the Organizing Committee Dr. Dimitar Kolev, that is, kolev@adm1.uctm.acad.bg

Bainov1

d bainov@ mbox.phaumfac.acad.bg
kolev @ adm1.uctm.acad.bg
Professor Drumi Bainov

Nov 12, 1999

Dear Professor Bainov:

I gladly accept.

Thank you for your kind invitation to speak at Ra Colloquium on "Numerical Analysis ~~and~~ Physics. The talk I would like to give would perhaps be more appropriate for ~~the~~ Numerical Analysis and Computer Science with Applications".

My mailing address is

R. Solomonoff
P.O.B 391887
Cambridge, Mass 02139
USA.

17 Nov 10
4 Sun
216.46.Hollow.com/NY Macon 5.
10

Most Sincerely, Ray Solomonoff.

N1299: Bulg Blg Paper BP British Petroleum; Brit. Pound. Bldg paper BULG Bulg

The Updating Problem for probabilistic Robots/Automata; — A Mixed Corpus Theorem.

1) Abstract ideas: a) Two of the most important problems in Pattern Recognition and Machine Learning are 1) first: How do you represent the information/knowledge that you've acquired and 2) secondly, when new data arrives, how do you update your knowledge? Algorithmic Probability (Kolmogorov Program-Length complexity) gives a good answer to the first Question. The Mixed Corpus Theorem Answers the second. We will discuss the representation of knowledge to some extent, but our main discussion will be the New results — The Mixed corpus Theorem and its implications.

b) "Machine learning in a Holistic World":

Up to the present time, most machine learning and Pattern recogn work has been for sharply prescribed domains. We will describe a system for solving problems and learning to solve problems that enables information from any domain to be usefully employed by any other domain.

The system is based on Levin's search Algorithm — which is known to give optimum results "within a constant factor" for each optimization problem and Inversion algorithm inversion problems. It had been conjectured that under certain circumstances, the "constant factor" would be at most 4, but at that time no practical way had been found. Instructions were given for obtaining these remarkable results.

~~The "Mixed corpus Theorem" tells how to update a conditional probability~~

The data base (knowledge) upon which the search depends, is in the form of a conditional probability distribution that relates all possible problems to all conceivable candidate solutions.

We will describe the "Mixed corpus Theorem" which tells how new information, in the form of new data or in the form of solutions to problems, can be used to update the probability distribution. The updating algorithm makes it possible to relate information in any disparate domains of knowledge and gives a quantitative weight to each such relation. This technique seems to bring us within view of the theoretical "factor of 4"

Tho it appears quite promising, it is not yet clear whether this updating technique will bring us within the conjectured "factor of 4" of optimum

The Mixed Corpus Theorem and the Updating Problem.

A Good approach to the Paper:

.02 1) Most any Prob. Solving System can be represented by a conditional P.D. Pre(input) conds. are the problem, & the output is a P.D. over poss'l. solns.

Give some examples of how this occurs & give ideas on how the P.D. can be used to solve problems [This is a Soln. to the "Knowledge Representation" Problem] → (.24)

(An open Q here is the form of the P.D. ... There can be many forms of P.D.'s & some may be their use in Prob. Solving much easier (less cc) than others — I will discuss various forms of P.D. & how to convert one [redacted] to another.

I will give examples of various types of problems — (Must cover enough kinds of probs that one wishes to want to solve — and how a suitable P.D. can be used to solve them.

Then, the main problem is — how to "update" the P.D. — as the system solves more problems, observes other systems solving problems or is simply given new data of various kinds. In particular, the MCT can allow the system to use info obtained in one area of prob solving & use it in all other areas of prob. solving that are relevant.

(2) A (related) Approach could describe my difficulties in writing TSP's for a narrow area of Lrng. — (I which I made no attempt to update it. Search techniques.)

(.24) A rebuttal of .02 One may not have a direct P.D. for the soln. to a problem, but one may have ideas on how to go about solving it. This last would seem not to be the same as a P.D. for the soln.

A possl. Answer: That "a method of solving a problem" can be regarded as a "kind of P.D." on possl. solns.

Or: One inputs the problem & the output is a P.D. over soln. techniques

.30 SNV in Lsearch: Suppose we have many alternative solns. to a problem. In this case, Σp_i can be $>>1$. It can also be $<<1$ (i.e. FLAT P.D. Problem) IMPT

However, in Lsearch the $\Sigma p_i \leq 1$ (or $=1$ if normed). In general, these p_i's are not from probys of success. — What are they? (I'm sure I worked on this problem in the past... don't know if I found a useful solns, however.) → 3.02

.35 It may have been in my analyses of Opt'n techniques: i.e. MCT! (likely)

.36 I.e. Q was "what does the type of a Opt'n method mean when we use it for Lsearch? ← .02

→ I may not have realized at the time, that the same Q. can be asked about Lsearch for INV probs. ← SNV

The Q is impt. in this case: Say we have to express our soln; — all about equally good (opinion); it would be better to spend all time on one to test it (yes or no), than get on bet. to next rather than two shares on them all simultly. It is = to "what to work on next" problem!

T. problems at 2.24 & 2.30 are very impt. I will look at my notes on MCT to see if I have solved them! — or for ideas on solving them!

.02 3.36 Re: 2.50 ff: T. Gambling house Thm is probly true even if $\Sigma p_i \neq 1$.
— But T. reformulation of MT to Lsrch is Weird if $\Sigma p_i \neq 1$ (or < 1).
The "what to work on Next" Thm (WTWONT) is relevant here — but it may need T. generality of "Task nets".

.07 Also problems in the net (or "OR" nets) need not be indep ... so results in one may affect results in others

I think T. GHT is for indep p_i. (not 1 thm .07)

Lsrch uses this Unc → "sprayed" — But what this "normal sprayed" means is not clear!

Lsrch can be very bad if $\Sigma p_i \neq 1$; say all $p_i = .9$; $n = 100$; all $cc_i = 1$ If one tries them sequentially in GHT, expected (no of trials) is $\sum cc_i$ (over); If Lsrch is used, expected no of trials, I think one has to try all of them, at which time probly of soln is very close to 1 ($\sim 1 - (.1)^{100}$) This is (I think) for either Time Sharing of T ← ΣT methods (+ latter takes time ~ 200 v.s. 100 for Time Share — but probly of success is $\sim (1 - 10^{-100})$.

If things Get Too Bad, Give "What to work on Next" paper. Delay lit of papers 'cuz as poss!

Th. System should be able to answer Q's of this form:
"My Top Goal is A, there are subgoals B & C: My recent trace is D, What should I do Next?

Another Big Problem in The General TM area is "kind of Pd's" — in T. sense of including cc. So for each (gendl.) Pd, we input a condition and a cc, and we output a pc. A simple example of a Pd. is a Mt. Carlo form: we turn on machine (uncondl. prob) & output is 0 or 1 (Bern seq. of unk. param): As we let cc (= no of trials) we get a sharper & sharper Pd.

So we have this "cc'ed" Pd: That may or may not be "Normalizeable". (2.50)
If it is (in "meaning") normable then we can use Lsrch. — If not, it's not clear as to what to do — If the cc info is of certain forms, it may be that Lsrch is ≈ optimum in any way.
(Tho in this case, Lsrch uses Normed pd. — where does this normed pd come from?

Say $\Sigma p_i = \infty$; we can't very well do Levin's "time shared" search.

.34 (Do) → In T ← ΣT methods, we would normally work on trials until $\frac{T_i}{p_i} = T$, and we stop when p_i is so small that $\frac{\Delta_i}{p_i} > T$, where Δ_i is a small time — about as much cc to

.36 transfer from one trial to another. We then do T ← ΣT and loop to (34) (loop)
if $\Sigma p_i = \infty$, this loop could take an enormous amount of time to solve certain problems. Certainly the expected solution time will be $\gg \frac{T_0}{p_0}$ (this is the soln — T_0 is as solve times p_0 is pc)

However, Do look at 2.35 → Maybe go to MCT discn.

If I do use Lsrch then and I assign pc's $\Rightarrow \Sigma p_i \leq 1$, I want to do this so that, on the average, $\frac{T_0}{p_0}$ = Min for the soln trials. We want p_0 to be large & T_0 to be small.

4-01

Oct 3.30: If there are several identical or near identical solns to t. (Inv.) say) problem, then P_0 has to ↓. (This sounds ~ to t. problem of wanting Lsrch to be controlled by t. $\leq pc$ of a soln, rather than its "shortest code". — I considered this to be a serious TM-ype blem — don't remember if I ever made any headway on it (probly not!).

A possl. key: (~ to what humans seem to do). Make estimates of $pc \neq a/o$ cc at each trial. In t. case of pc, it's "probly of probly" — (a concept that I don't ordinarily use). But I may use it in any way — This it could be mathematically (bad/meaningless)

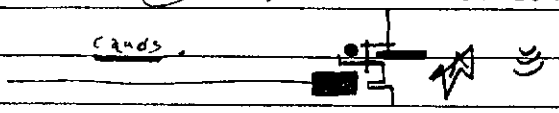
Alternatively: try to find how "I" normally solve problems of this kind. A simple example of a not-normalizable pd: Set of/(five) humans in U.S. = S_i: (i) probly that S_i will live longer than. Re sum of t. probys is not useful/meaningful.

Re: Lsrch and non-normalizable PD's: Some approaches:

(a) Try to guess at cc a/o pc of candidates cands at an instant — what we really want is a guess at $\frac{pc_i}{cc_i}$

(b) Try to show that certain sort(s) of cands are (dominated or "close" (close in t. sense that if one does not work, its (likely) descendants wouldn't work). This tends reduces t. no. of effective cands. ABCDEabcde

(c) Also, if one works on cand, a la this, this could change (say ↓) pc of (probably related) cands.



Ideas of Generalized PD (including cc info): one way is $P = P(cc, x)$ Σ x is "cond" of cond'l proby, cc is (computing) cost. This includes the possy that pd is not "normalizable". In fact, the non-normalizable case (if described) can take much of the cc, & is usually only approximate. [t. purpose, can search many forms: Monte Carlo, (light-path) ray-tracing, etc.] So another direction of "improvement" of a PD, would be to fit parameterized forms so that Lsrch would tend to get solns faster. Some ways to do this:

1) Make pc's normed or near normed. (Any bonus, this increases rel pc's by equal amount & doesn't affect Lsrch. — except that it affects our estimate of srch. time $\frac{\text{cost}}{\text{prob}}$ — if we know a soln.)

Oct 3.35 2) t. "GPD" (Generalized P.D.) includes guesses at $\frac{pc}{cc}$ of

3) Some cands. Info on non-independence of pc's: so t. failure of one (and trial) will strongly influence pc's of other cands. (This makes it hard to evaluate $\frac{cc_i}{pc_i}$ of sol.)

Title: The Updating Problem for Learning Automata: A Mixed Cases Theorem.

5:01

Sol: 4:40 A very large class of hours can be put into the form of "changing to PD".
The idea of 4.35 — to make our p's dependent on known failures (or "partial failure" in the sense of much accumulated w. no response.)
A very simple kind of "non-normalizable" p.d. we have 10 cards:
each has prob of success of .5, & cc of each a 1 second.
If we knew in advance about the "1 second" we would test them each (for 1 sec) in any order.
We might decide that they weren't all about similar & choose a representative sample of them, & normalize pc over that sample. — So we do it.
"Simple" first.

In general, a hour should be allowed to change any aspect of the problem solving process: Including changing Modifying (or even replacing) (search, itself.

As I see it (& perhaps have for some time) the main problem is, how all hours can be [(put into) (learned by)] the system. "Quick abort", is, of course a major duty: Hmm, I'm not sure that I have/ solved this particular problem.

.24 Hmm. Think of it this way: If any hour is justifiably able, it must have data loading bodies, justifying it. [This is a very old, very inept idea about hours.]

O.K. so: Hour to we apply it. very General principal of P.24
to Quick abort (&) 2.30
I may have written .24 in Sol 86 or 89 — Maybe re-read from back!
2/6 & notes for Room 1

.28 D2199 AH! This is it: A "Heuristic" is anything that "improves the system."
This is a D2 problem. Improvement has many parameters representing the user. (Improvement in what way? for what goals? for what kinds of goals? ...)
The simplest hours merely "improve" the P.D. (This "improvement" may several parents, but P.D.'s can be partially ordered in an "objective" user-independent way)
Next, improve the Pgm instructions (not restricted to using Search — the Search is e.g. to start w.) (Improve S.W.)
Next, improve the H.W. for this, we need a NW dem. (say. This long has to be continually updated w. latest (technological) developments in Chinese in our view of physics. Presumably a succession of Machines could need to work & keep up on this itself.
As part of H.W. improvements TM could do "[illegible]" in R.W.
G.A. systems that have been allowed to do this have already developed circuits that were incomprehensible to Human Eyes!

D2199 Bulf

CC ; CB.
MCT Notes N.10.98 & 9.7.99

SSTM 195 ; D1888 s=178 M2 two radix>2 unsolved.
Bird D16 : Pidgeon MCT, also P138
P130 for narrow, but not pp!

So how does this all deal w. t. "Cat P.D" problem of 2.30 (and "Quicksort")?
Well, if we want to stay w.th Lsrch t. only [redacted] kinds of heuristics we can use are [redacted] to ↓ $\frac{cer}{pci}$ - of soln. by

1) ↓ cer by better H.W.
2) ↑ pci by concentrating on certain regions of Lsrch space (Also by finding better regularities in Given CB). Many heuristics of this sort.
a/o decomposability / Eliminating other regions.

[redacted] Recognizing that certain conds are identical or almost identical amounts to (modification of P.D).

.09 Making pc's of trials conditional on lack of successes is CC expenditures at a Prior trial. — Modifies P.D. & $\frac{cer}{pci}$ of soln.

[redacted] At any particular time we know which kinds have failed (& better e.g., pc2 at failure times) so the modifn of our pc's of "unconverged" conds must be a funct. of just info.

But, in general, it may be that t. first P.D. (which is [redacted] outside t. spirit of Lsrch) will be mainly done (& w. by going outside Lsrch. How to do this, is not clear. Tangible hint is t. first GHT. (Gambling House Thm) — that $\frac{cer}{pci}$ order is Always Best, if pc's are indep ... pc's need not be normalizable.

So, t. pgm for t. system, initially is ① t. P.D. ② t. Lsrch pgm. serial serial.
So LP is t. initial pgm. To improve LP, we can improve P or improve L, or both. Other than time shares, I don't know if Lsrch can be "improved" & still be Lsrch.

Here, t. problem of improving LP is a well defined problem, — & t. concept of "Improvement" is well defined. At first, I'll just have to construct t. Gone "By ear". T. Gone will contain many Subjective params: Like (CB) horizon & "Areas of interest". Could be conditioned? probably

.29 T. idea of the MCT was to devise a single unified P.d. for 3 kinds of problems/corpi 1) INV 2) Oz 3) proofs of some infinite T.S [redacted] ④ Predictions on finite collection unordered objects, ⑤ Corpus of "pure data" ~~Sequences~~ pure ordered, pure unordered, part of pair of semi-ordered sequences.
Still, what I did in t. analysis of MCT was to decide just what is "Oz problem" P.D. ~~[redacted]~~ we presented, & how it could be made part of t. p.d. that was to be used for INV probs.

In .29 we can Generalize t. idea of Levin Complexity P.d. to include systems in which we input a problem, & the system tries to output a soln. Trouble is, for INV problems, Expected errors is zero for all non-stable & x. probably has a solv. strix is same as funct of CC that → 1 as CC → ∞ ~~[redacted]~~ (assuming problem is solvable — which it is for all INV problems (?) ~~[redacted]~~ — No! Some inv Reasonings are Unprovable!

D2299 Bug [MCT] 161032 7

Trying to find motivation for sec of MCT: Evidence exists below. N 1098 & 0-7-99.

MCT refs: [In 98TM]

130.01 -.40 seems basic review: some later stuff on 138.01; 144.01. (review) 131. (20.14-18

More work: 106-108; 110; 111, [115], 117.31; 120.14-18; 129.18-.30; [130] 138, 144 129.18-30

137 is on "Summarizing Machines" 111.03 may be necessary to avoid needing to prove Sol 78T3 for radius=3. 130

Good!

0.4. I guess that's it: read 130 [and 131] very carefully from browse thru t. pp. looking for critical ideas.

[111.03 ⇒ 110.08 -.36; 111.03-.17 seems to prove Sol 78T3 for a finite set of unordered strings; we t. need to generalize Sol 78T3 to radius 3.

In general that sequence from 106 — 144 seems to address impt. problems!

One problem is "how to approximate" t. Summarizing Machines".

Also, In general, on reading that old stuff I may have quite new ideas on how to deal w. t. diffies! → 35.01

.12

.14 [SN] Consider all times that have no "stop" states. It is likely that 3 times that can simulate all of these machines — so these Machines are "Universal" over that set of Machines. Machines of this type would be useful for predicting unlimited sets that are known to not stop, ever! It has infinite (1 way) input tape, bi-inf. work tape, unidirectional, inf. output tape. That machine never stops, it can continue a never ending sequence.

.20 [5.21 ff] may be a V. G. way of dealing w. ("proving") t. FN of "P5" of Sol 89 ("P5" is de very definite, but it's t. largest FN in t. paper; it says that Bill Gosper suggests "Any set of axioms can be simulated by Blind Search over a sufficiently powerful set of concepts" Please explain how many axioms outside those of t. problem being solved. This deals w. theorems that can be expressed as Models of t. underlying P.D.: "Quick sketch" for instance, would not be so expressible. Nvr. some 5.21 ff [more] can work on this!

SN: on general structure of TM: IAM 118, 119, 120, 126

I have to put t. proof (still to be of t. MCT, but looks like Marxly "the [finite] unordered set of finite strings" thm.) of 110.08-.36; 111.03-.17 in more explicit form. [But no, recently worked on Murderspace! see 110.03-10 exact form.

I'm beginning to sort it! — Let This Go for a while, I'm now nearly interested in how I dotted OZ problems into this "Mixed curves" pp 110-111 seem to mix finite objects & potentially infinite except together

.30 O.K. 130.21-.28 "Ce") [overlaps] how to deal w. OZ. (a loose end, but, access w. t. SOP problem)

Looking thru 130.01-.40: I think I have an idea understand how t. finite unordered set of strings "converges" works, & how t. OZ p.d. is updated & how they all fit together w. S.T. proof.

T. P.D. used in OZ is unusual, but it suffers from t. flatness diffy as much (or less) as do p.d.'s assoc. w. ENU problem solving. Nvr, t. "proof" & "demo" involved in t. OZ P.D. suggests that one can get away with kind of info and "chaos" out of t. "proof" & put it into a P.D. or some other

.39 In

ABCdefghi

(Stochastic Data Structure: It suggests that I may be able to get good estimates for $\frac{cc_i}{pc_i}$ ordering for a given problem, by direct probabilistic linkage to data of past problems. This $\frac{cc_i}{pc_i}$ ordering has to change as one works on the problem ($cc_i \uparrow$, $pc_i \downarrow$) so it may no longer be the first in the list. So one starts working on another but fairly quickly the first [several rounds on same problem] expected $\frac{cc_i}{pc_i}$, so one time shares between the two. Then (not necessarily equally) — one deterministic way that keeps the $\frac{cc_i}{pc_i}$ constant or near it. Time shared sort of rounds. (This begins to look a bit like ordinary T. shared Search!). → (27)

I had been thinking that the Search Approach of the "What to Work on Next" alg. (WON problem alg) might be relevant. The params of each task (needed for the assoc determination) that one needs for the WON calc could be obtained directly from past data, past problems solved.. For WON probs we need the prob of soln. as a function of cc. In general, the shape of this curve will change as each round is worked on.

I think I showed that for a certain common shape of the curves we would get Lsrch (Unclear how this could work! Lsrch depends on the instantaneous $\frac{cc_i}{pc_i}$): WON depends on assumed shape of prob vs cc curves.

GHT Gambling House Thm #1. 5/86. GHT does trials in strict $\frac{cc_i}{pc_i}$ order: TS Lsrch shows cc ≅ $\frac{cc_i}{pc_i}$ is constant for all rounds being worked on (i.e. having above threshold pci).

→ I really don't remember the WON formula, hyr.

2 Aspects of "updating the P.D." ① Incorporating New Data ② Putting pd. into form in which it's easier to use to solve problems.

T. talk will be on MCT, which is Merely on how to incorporate new data.

27 (10) → If a P.D. is (2 CPM normalizable), it's always possible to devise a machine so that $2^{-Ucost(x)}$ is within factor of 2 of $PC(x)$. (Kraft's shortest code). Perhaps this is "dressing up" of the P.D. will sometimes speed up Lsrch.

Proof (perhaps by) Modification of ordinary Lsrch (Post includes many more types): (failure Progress); nature of (failure progress) on many partially tested rounds, can modify consideration of the P.D. on other (& the same) rounds.

→ How this relates to the "Flat P.D." problem: — On second thought, it may not!

T. GHT thm. seems to relate more to the Flat P.D. problem.

(Q: If we neglect the Flat P.D. & use Lsrch anyway (for normalizable P.D.'s) will this be "good enough" to get us off the ground?" (There is also the problem of Non-Normalizable P.D.s — [its my "solution" of the P.D. updating problem reformulated to a more general class of unnormalized P.D.'s ?] ?

9.01

.01 8.40 Mumm on "unnormalizable PDF's": It may be that they don't really exist! That if one has a "probability" of an event, there will always be other possibilities and the total of all will be ≤ 1: I had in mind the probby of a group of 1000 60 y.o. men, living 10 more yrs. Say for this particular class of men, the probby were .5. The way this is normalized is that there are 2 classes events: (1) Man lives 10 more yrs. (2) Man dies in ≤ 10 yrs. The sum of the probys of these 2 kinds of events is 1. We do not sum over the 1000 men!

.08 — But the probys can be normed if we use the proper classes of events.

The way the problem is solved via ALP: we have these 10,000 men's data: each man has a dcrn of his relevant params followed by 1, or 0 telling if he lived 10 yrs more or not. We want to use short descrs of this data base to extrapolate to a new person's set of params. — to give probby that he has a 0 dcr. See 1 following.

So, the problem is not "unnormalizability". A more likely problem is "flatness" caused by (2) alternative descrs of the same brief (6) trials that are highly correlated, so failure of one ↓ the likelihoods of others.

In the case of (6), we do want to do one of the trials first, it possibly rather than three.

The T ← 2T method of Lsrch would seem to be the help here: In the final round, one tests the first of a set of 100 candidates that are "highly correlated". The first one fails, so one doesn't try the other 99 on till much later. While this may save a lot on the final round, the rounds before still take as long as the final round — so the total cc of solm, is only reduced by ≈ 2. (20 yrs → 1000 Bugs)

If we do T ← 10T: we reduce total cc of solm. by 10, but on the average, solm's cost ≈ 10 times as much as T time should Lsrch.

One area in which it would seem that one has very many ll codes that are close but ≠ is in continuous Math. e.g. linear prodn, in which one has orbly many codes, orbly close. In this case, (thru the shortest codes) is not bad. It takes the width of the D.f. into account. — It gives a short code near the middle of the D.f., then longer codes at other pts within the peak — say that longer is about the same or a little better over — so they would be considered.

So! "Flatness" can (say many alternate descrs of the identical card) can be annoying, but we may deal w it by "improving" the P.D. (This is the method of improvement so the shortest codes is within (of candidates − $\log_2 PC_2$).

In general, here, it should not be circumscribed Doability, as I was (10.01)

1.05.00 Bolg.

GPD (.06)

Generalized P Rabg distribution. (usually conditional)

.01: 9.40 worried about "Non-normalizable" P.D.'s: 9.01 — .08 suggests that such P.D.'s do not except — that we just have to find & write normalizable ones.

This seems to give a great sense of Relief!

.06 The other most Recent IMPT. IDEA: That any kind of mathematical or computer or Physical "Model" can be used that sort of as a "P.D." "New P.D." does includes ... Each w. distinct output pc, depending on how much time (or cc) one Prob is able to spend. Here, I idea of "improving t. p.d." means getting it closer to a form in which t. output is most "useful". So "improvement" must be very subjective depending on one's goals. If one is mainly interested in Cancer Rsch, then "improving t. P.D." means improving t. performance on t. part of t. P.D. that deals w. Cancer rsch.

If our time horizon is 6 months, we will presumably improve t. P.D. in a way quite different from we would w. a 10 yr horizon. The longer horizons put more wt. in Meta P.D.'s — things having to do w. improving t. p.d. in t. longer ranges.

→ Another Impt. kind of operation on P.D.'s is Converting from one form of P.D. to another. Among other things, this facilitates various x forms, combinations of P.D.'s.

Another nice thing about t. Bayes. concept of P.D. — that it enables us to conceive of various mechanisms that can produce P.D.'s — some of them Universal (or approaching Universality). .23 → (1.01)

Another (Reverend Bayes) of our Empt. ideas: That any Heuristic can be put in probabilistic form by looking at data that led to its discovery, & using ALP to obtain a probabilistic form of t. same heur. This will also enable us to understand t. heur & perhaps even learn how to improve it & to suggest new places to look for Heurs.

This pen, using Bubble-jet printer ink does not work very well. Small, but does seem to get a bit!

So, At this point, perhaps I should make a more detailed outline of what work has to be done for final TM! Gen'l Outline + some detail re what seem to be Major Problems.

(SN) An Interesting, essential, problem: Just how to express logical Form of Prob in a ALP way — e.g. in figuring out heuristics. One way to think about it: logical forms are of pc = 1.

1.12.00 Bulg GPD Scandinavia 58

.01: 1023 (GPD) Emphasis on commonness of GPD's: Any Operator (I→O decision) deformulates, Probilistic, Digital i/o Analog is a conditional GPD. The stockmkt is a GPD but if one doesn't affect it, it's a Unconditional GPD.

The Goal of Mucc of Research can be regarded as "Improving" the P.d. (which is the R.W.) in the sense of obtaining Mathical models that can simulate these factors, a/o give good ideas on fast prediction. Good ideas for "theorizations" of models.

1.14.00 In final (& not-so-final) TM, I would want TM to be able to use practically any kind of info that I create, & be able to integrate this info into the "Mod Corpus". Perhaps the most important kind of info, (at the beginning) would be Hints on how to do Optzn. & "Improvement of the (GPD)"

It might pass to make TM very simple in all ever administrable. That its cleverness would accrue from "Hints" and clever Fig. Sequs (or "well designed TSQ") (But not too clever!).

7.01 → .40 → 8.01AF : This page 2 has hints to how to solve the 2 probs: 1) Bgnn's proof of [redacted] design bug of Quicksortings extension 2) MCT.

So: Write these up more carefully.

Abstract: In AF: 2 kinds of Probs: ① Good way to represent knowledge (Good Axioms that are easy to get Machine to "do its Ring" of E. known. representation is Good: Rn is knew Rep- parse occurs in ALL of AF.
② In A.I. devoted to Machine Learning, the new Paradigm problem" — Given a/ incomplete knowledge Rep't system has at a particular time — when it is introduced w. new info, (or brings system data or beliefs or even some of new probs — how does it update its knowl base)

If one has solved Prob 2 problems one has solved all can build a Learning Machine. How well it learns depends on how solve Prob 2 problems.

2/3/00 Bucs:

"The Learning Problem for Automata"

① ②

I want to ⓐ a closer proof of MCT ⓑ to base of finite sample induction.

.02 ① ibid 98T 110.08-.30 ② ibid 111.03-.17

I need to also describe "true" pd. as a cpd, says. Then state that I can simulate, w. ε Turmc (not nec. a umc); so I get to "constant factor" upper bnd, on PC results.

Also, the rest of .02 deal only w/PC & Rest can deal w. finite & infinite strings extrapolation: .02 problem is not dealt w. Rand. 98TM 130.28 is a Bibl. review of the Soln. of Reis problem.

ibid 130.01-.40 is perhaps a real. soln. not to Machine Learn. problem.

The foregoing stuff uses the "summarizing machine" idea & but I'm not sure it's adequate} Which doesn't seem able to deal w. OSL (one shot learning). It may be possible to modify the ideas so Rest OSL can be dealt w. However, Yesterday from OSL seems to pose no problem: It's a regular corpus augmentation w. a coarse count of 2.

Look at ibid 138 — 156: Did I have any ideas of serious limitations of bibl 130.01-.40 as an adequate soln. to the "Real TM" problem? * repeat.

ibid 131.01-.40 is perhaps the main thing here. (I browsed thru 131-136)

The Abstract needs to do 2 things at least:

Make the proposed talk look interesting enuf to Rest

① Banerji thinks it's worth paying me, to come there

② People in the Cond. will be interested enuf to attend

One way to do this: Present the 2 ideas as a legitimate reaction of my initial theory of AL & the developments since then: Mentioning as many imp ideas in the abstract that I can explain clearly.

Mention impact of Computational Complexity

Machine Pattern Discovery
Computational Complexity,

.01 Abstract: Many years ago, I devised a general theory of inductive
inference, meant to be applicable to all kinds of problems
.02 involving probability and learning — by machine or by humans.
Since then, there have been many developments of these
ideas, by many others as well as myself.
I will review some of the important developments since then, and
— to investigate how far we've gotten in understanding
probability and learning. To what extent do these
developments adequate for the design of a
machine with the most general possible learning capabilities?
and will try to determine if these
and will discuss the design of these developments for the
design of a machine with these most general possible learning capabilities
Discuss what kinds of machine (algorithm) is what constitutes "learning".
This is more like an introduction; the observations and conclusions
& main new ideas of Paper
So I will try to show that these ideas are ~~in general~~ adequate
for the design of a - - - - (perhaps I don't want to make so strong
a claim)
My goal has been to program a machine that would be able
.21 to learn ~~to learn~~ - - - - Since then, that
learn on the basis design of a machine with the most general possible learning
.23 capabilities.
~~Machine learning~~ The problems that one wants a machine to solve almost
always of 2 general types: ~~find~~ In version problems (like P and NP problems
of computational complexity theory) ~~find the optimal~~
for example: Given the square of Given a number to find another number y
such that $y^2 = x$; or More generally, given a program machine M, and a possible
output of that machine x
to find an input y to that machine, such Such that M(x) = y.
Copy stolen from some papers on optimal.
known in ~1973? Levin developed a general optimum solution
for Inversion problems. Soon after I found him in 1973 was working
to generalize this solution to ~~more~~ time limited optimum solution problems.
In both cases, the solution was avoided by ~~approximately~~ computational
distribution. For the case of for inversion problems, suppose that
inverse given

2-7-00 Bulg.

01 13.23 It is perhaps too detailed for an abstract.

Almost all problems of interest are of the There are 2 types of problems
that lower just about all problems and might give to a machine:

The first are I unsolvable problems — the Post Normal Form of comp. complexity theory

The second are Optimization problems in which one has to find the best solution
possible in the given available time.

Levin devised a search procedure that works for both kinds of
of problems, that was based on a conditional probability distribution.
It is was a very good probability distribution. In all cases, it gave a solution
that was within a constant factor of the best solution obtainable. The
If the probability distribution used was "very appropriate to the problem,
the "constant factor" would be very small. If not, the constant factor
could be astronomically large.

The Levin Complexity, Not "probability distribution was based
on previous experience solving similar problems, but however
More recent universal rules for updating p.d. with
respect to any experience the machine might
have, or any data that it might be given.

The work I will describe is a ~~for~~ method of applying the
probability distributions used in Levin's search procedure.
It enables us to ~~do~~ what is described, it enables us to use
~~data~~ experience or data from one/several problem areas and ~~use~~ it
in a/the problem area ~~which may be related~~. In particular, we can
~~in a natural fashion~~ use the same probability distribution for
both inversion and optimization problems.

Recent I ~~have been~~ working on machine learning, ~~using~~ using Levin search
procedure and
~~using~~ a training sequence of problems of progressively greater and greater
difficulty, to learn a ~~to~~ initially naive machine to acquire of a good
problem solving skill. While I was successful in writing a few training
sequences — they were extremely difficult to write.
Using the new updating procedure, it appears that
training sequences will be much easier to write and will enable
33 much more rapid acquisition of skills than ~~the~~ other before.

An "adequate" abstract would be 13.01-03; 21-25; 19.01-23
But a bit longwinded.

Shorter
A more Abstract:

(omit) ~~There are 2 very general kinds of problems that research in science and~~

Just about all problems in sciences and mathematics are either inversion problems or forward (induction/extrapolation) problems. Levin has devised a general search procedure that is guaranteed to find the optimum — within a constant factor." The search is guided by a conditional probability distribution, and if the distribution is "appropriate to the problem" the constant factor will be small (perhaps as small as 2 or 10). If the distribution is inappropriate, the constant factor can be astronomically large.

I have used Levin's procedure to solve certain simple problems, first by modifying the probability distribution for the problem and then by inventing a training sequence of problems, so that the system was able to solve more difficult problems. Using a carefully constructed training sequence, it was possible to get the system to solve problems of impressive difficulty, it was possible to get the system to solve problems of various difficulty.

However, the writing of training sequences proved to be very difficult, and the techniques of modifying the probability distribution were very ad-hoc.

I will describe the recent work I will discuss involves a very general method of updating the probability distribution, so that a system can use the same conditional probability distribution to solve a great variety of problem types. This seems to eliminate a critical difficulty in practical application of Levin's procedure for problem solving.

This is still more an "extended first" than an Abstract:
An Abstract is further in terse to the point!

We will describe a very general method for updating the probability distributions that are used to guide Levin's search procedure for problem solving. This makes it possible for a system to solve problems of progressively greater difficulty. The generality of the updating procedure makes it possible for the system to learn solutions of problems of one type to solve problems of many different kinds.

Later (or in talk) discuss different parts of the corpus: multiple TMs, multiple training sets, corpus of data, several OOPS solns. — that all these R.D.'s are (apparently) different.

1) Possibly on th Abstract, mention that "Improved P.D." is (possibly) Xtra input from (S.S.) an OZ problem — so essentially, we have all the ingredients of a general soln. of the bearing problem.

2) For Me outline just how the final TM is to grow: [illegible] How it is related to "Prime" TM a bunch of OT's (optimized Tech necessary) to start, so it can extrapolate from. How any kind of info of this sort can be used by TM.

3) T. mentioned not to give TM very much info to start, as Realness would bias its development. Perhaps actually slow it down considerably!

4) Re: Hardware/Software Analysis for "Instantaneous Cognition/Data": Look in Arbib CACM on Bio sensors for refs on this; & use also dynamic routing protocols (when possible) D.H.

5) Look at Sol86, Sol89 to see how complete soln was: Prog may have some new ideas: Also, I should "flesh out" & critical Footnote (#52) in Sol89

6) I do want to give TM (at least when it's more developed) a look at the Breadth aspects of Joe Traub's "Info based Complexity"

7) An imp't kind of sub-problem is recognizing when one Gen Prob is "in", so one can use refinements of it's soln's & info to solve it. Prob. e.g. To recognize similarity of TS pred'n. vals (Values to be predicted) & discrete time. TM could begin w/ Real's as Ploony #'s. Then it would have to recognize that integers were a subclass of reals.

8) While Prog. TM is capable of being "Told" anything, I want to impose some sort of Prog, since "Telling" narrows down the generality of what is learned. TM should learn by induction more or less "on its own" as it's processing.

9) I was warned about time series predn & finite Set predn. being basically weird. Seems. only. (since they are in set theoretic OZ probs) In both, the problem was the design of the Gen.

12:01

2/8/00 Bulg.

.01:16.40 (0) It would be good to draw up a somewhat detailed picture of how TM works in "STEADY STATE" as opposed to "start up". This will be ≈ how Humans solve problems, but in a particular language, using particular "Black Boxes".

2 Parts (1) Production / Solving problems of USR.

(2) Updating (≡ "Improving t. P.D.")

.07 Part (1) Probs of 4 types: (a) INV: solved directly by Lisch using suitable cond. P.d.; (b) T.S. prodn. | finite set prodn: First, as input, it is clearly (a/b)ded as to type of problem. (≡ definition)

Def of (b) If it's (SF/TS) prodn; it is a OZ problem: Th. proper def. now Optimal Agent (OA's) is obtained as part of "conditioning" of t. cond'l P.d. || Input to P.d. is dcrn of OZ problem; output is P.d. used in OZ Lisch.

(SN) T. deals. of a cond'l. P.D.: Given a finite set of I/O pairs (strings) finite to finite or infinite to infinite

(I_i, O_i): Given a new I_n to get P.d. over O_z. This is "finite set to finite strings" extrapolation problem.

Also, Say we are given a large corpus (like an encyclopedia, or t. internet) We are also given a set of Q,A. pairs based on info in the corpus. To give P.d. on a new Q_z.

After TM has "learned" about t. relation of a Q & A to a corpus, (C_1) It should be able to extrapolate this idea to a new corpus (C_2) w.o. any new Q,A pairs

One way would be to simply regard $C_1 \cup C_2$ as a new corpus w. some set of Q_i, A_i pairs as "examples".

It may well be that there are many TYPES of problems that I haven't thot about — that TM will need "special Arrangements" to work on them. Hvr, I do want a reasonable representative set of probs that TM can work on

(1) (Type 3) OZ problems: As in (b,c) Input to cond P.D. is dcrn of OZ problem; output is p.d. on OA's to be used in OZ Lisch.

.26 (2) Updating "Improving t. P.D." Nominally an OZ problem, but t. Gone for t. "P.D." is rather unclearly defined — it needs lots of input from user to be a true "Defined".

Some kinds of inputs needed: (1) Time horizon for some to various classes of problems (2) Relative importance of various types of problems.

E.g. How much time should be spent improving certain parts of t. INV problem

d.f. 2 i.e. we have INV problems of various kinds.

A reasonable way to study (2): Give various updating routines & discuss how they perform on user's Needs. kinds of updates:

(A) TM has found a good grammar (G_i) for a particular "finite set to string" problem (induction). It might then try to fit this new Grammar (G_i) into the set of Grammars that has found useful. — To start off, the new Grammar has already been assigned an apriori by TM, for t. search to find it has found it. This "Grammar Grammar" now has one more member in its known corpus & so it should be updated.

.38 The updating could be simply updating params, or it may involve inventing new concs. for

1. Grammar — which usually involves a much longer srch. If we had been just doing Grammar/param updating for the last few grammar discoveries, — then every once in a while, Grammar/Grammar
.03 one should do a more serious ("invent new ones") update for the Grammar/Grammar.
.04 (B) In effect the "more serious" update of .03 is a different kind of update. The param update of 17.38 is small cc a rather "rooftop".

(C)

In general, there may be many kinds of "updating": Also, it may be difficult to separate "Updating" from the "production" phase of TM (17.07)

Perhaps "Production" is part of TM activity that is most narrowly concerned w. "solving the present problem". Often / solving such problems will "Improve the P.D.": as in finding a grammar for a set of ss's or perhaps any OZ problem soln. contributes to the goodness of the Global cond. P.d.

or, the Abduction problem can actually BE "Improve the Global PD"

During the soln. of an INV problem, one can also contribute to the Goodness of the Improvement of the Global P.d.", by solving subproblems that may be OZ problems.

There will be several (perhaps Many) different kinds of tasks involved in "Updating". At first I can ("By hand") control. Process, until I get some ideas as to how much time (& when to spend that time). Mainly, I will try to find which subgoals are most necy, & try to frame a Global goal so the updating problem becomes "well defined".

.22 Re "Core of a GPD": In General, there will be a trade off between ("speed") & accuracy of PC. This trade off must be decided by the USER. Also, there are different "parts" of the pd. — & the User will have to decide how much (time) to spend increasing the accuracy of the various parts.

.25 Concurrently, .22 - .25 concerns Deals all the USER input that will be needed. — HVr, there is the "Time (horizon" Question — (.22 - .25) doesn't seem to Address it! This involves Q of how much what fraction of time to spend on near time goals v.s. More distant (in time) goals — w.r.t "improvement of the P.D."

2/8/00 Bolt

Time on the dogpit

free NoET. Hand for his —

165k yrs in 8 mo. while living [total ~ 1Kyrs his names] as [illegible] A [mustard] was small cumbersome,

So [illegible] Speed x 250k / by mvt. to 30 yrs of x.

250k hz

1.5 / 1hz = 2.16 yrs for me

x 250000

x 2 m 1.5 yrs

= x e m 2.16 yrs.

In 250 hz 12.43

2 6.84 yrs

of development

2 x 2 in [illegible] yrs

gives x 250 in 26.86 yrs.

1.6M people downloaded S.W.!

In t. Oz part of t MCT "proof"; from 95TM 130.26 — option technique, computation Bnd, Gork.

we have as dataset [OT_i, CB_j, G_k, problem] — Rosa 4-tuples.

From Pam, we can get a [illegible] P.D. at any [illegible] (OT_i, [CB_j] G_k, Problem)

How we integrate Pg to get a "Best" is unclear! If we have given a CB for problem, then we get a P.D. over G_k. — So for a given CB, there may be a "Best" OT_i.

This looks like a new kind of WON (what to work on Next) problem, but for OZ probs. Rather than INV probs! On t. other hand, in view of the GHT1 (Gould Hts Rule 1) it may trivialize such! (If t. Gate is linear!) ← No! cc is always linear, but t. restrict of t Gate may not be.

Woops! This brings up an unpleasant possy: when we integrate in 4.04 to get a prob. of one OT being "Best" do we assume linearity of t. gate by taking "expected values" of G? I think Not, we just integrate over all t volume of space where a given OT_i is best for G_k.

So, it looks like Cat 2 recog by (level of abstraction) t. system is "Reasonable." We give TM all problems: it solves them w. Lsrch, using t. Pam-correct Cond'l. for Updating t. new data after each problem is solved, is t. "race" for that problem, as well as t. soln. Since t. various (decisions) across t. trees were Guided by Rec & Cond. P.P., it should be easy to update t. params of t. P.D. (This is t. easiest & fastest part of t update (17.38))

In most Complex problems, we will not use t. P.D. directly to generate trials. More likely, we will use it to generate t. next action in an investigation. Also I think "Mathematical" (Non-probabilistic) reasoning is often used — which is something I haven't yet integrated into t. System.

(.24) is fine for "Planner" heuristic I discussed (a bit) in Sol 86. — We first decide how to make an AND or OR out of t. problems, then ask for the WON params of t. tasks. (Actually, all we need to check for "OR" nodes is t. initial (current) slope of t. tasks: After we've gone along that slope a while, the new slope(s) will tell. Cbgrd on recent experienced will be generated & used for Switching Criteria.

——— Hvr., I'd like TM to be structured so that it could discover t. "Planning" heuristic. (Tho, actually, like any heuristic, Reed is a Standard way for TM to discover it. — Use t. data that would have justified t. heuristic

(.38) (Another (?)) approach, would be to have TM's traces in solving a problem be t. "soln" to t. problem. — A disadvantage of this

.01 Maybe that TM would be "missing pt." & make errors, [redacted] or use wrong pairs" that were unnecy. [redacted] {Like reproducing a "Bad axle" by the "German mechanic".}
So we want TM to appreciate what the problem is, & realize that certain solns, while "acceptable" would be better if they were shorter, & had fewer "incorrect" branches.

.06 In 19.38 ◆ The reason for the ? after "Another" — that I had in the past part of the trace as being a soln. to the problem. — That to invert 2 w.r.t. x^2 was not 1 digit ... but as good to obtain $\sqrt{2}$ from 2. As a pgm, it is reasonable to expect it to be made up of a few "components" that had been parts of problem solns. in the past.

This is perhaps in the spirit of the ideas I had for TM prog. a TSP from an elementary Algebra Book: first, learning several (various?) input definitions by induction. — Then I try to solve problem by using (among other things) the 3 (definitions (& their component parts)) to combine to form trial problem Solns. .35

.15 Also, I want TM to realize that all INV problems are also, to some extent OZ probs; since we want (among) CC solns. obtainable — Tho it's not a usual "OZ" problem — we are not given a # many soln. times! It's more like an "Anytime" problem! — So perhaps "Anytime probs" are a [redacted] way to unify the OZ & INV. concepts. .12 .34

— A trouble is, there's a certain vagueness in the defn. of the "Anytime" problem — Tho, whatever it is, it's closer to what we usually want, than is the strict "OZ problem" definition. (Tho solving OZ probs by Times & Varying is probably a v.p. way to solve "Anytime" problems. (The T ← 2T method would work also but not quite as well)

If we think of the INV problem as simply having a solution [ϕ or 1] (or, [redacted] Search for OZ problems should work — but we would simply have a default cond'l. P.D. for the OT's — & OT's particularly good for INV. probs. (TM could immediately recognize INV. prob. by its G function, being ϕ or 1 only).

.31 .07 TM could easily obtain a soln. quickly (but bad by pc, but had redundant calcns. (for the present problem). This is analogous to doing $\sum_{i}^{100} i$ by adding nos. rather than by $\frac{100 \times 101}{2}$ if one was very fast at adding nos.!

.35 .15 This Alg. Book idea seems very attractive! I could put in a set of problems in linear order; Also at each pt., the concs. needed to work the problem! Some concs could be obtained via "Definition discovery" (induction), others by noticing ways in solns to problems.

.01: 2D.fo: It would be interesting to try to do it using TSO vs. GA (or SGA) or ANN or RANN (recurrent neural net).

.02 [A bigger difficulty w. GA is that even only working on 2 problems. — Once possible way (working on INV probs) would be given a bunch of INV probs, with their solution traces. The GA then tries to find a function that maps the probs/solns to soln traces.

.06 [Or, at least possibly, fraction of cases. Or, more generally (but maybe more difficult) to find a function that maps probs/solns into solns (not necessarily using the "known" soln traces). Each candidate would try to implement that function. Some candidates would be good for some probs; some candidates good for others. However, on the whole, many may be able to solve all of the INV probs.]

The main Q is: How small in pc of candidates do we have to go before all of the problems in the set are solved? — Well as we continue evolving, the pc should ↑. — (Also the cc for each candidate may ↑! — so we have a trade-off — which we may need to work into the GA): So at first a candidate is given credit for how many INV probs he solves in given time, T_0, then we may slowly ↓ T_0. → 24.01

.15 Updating probability distributions used to guide Genm's search

Out of abstract Contrib: Use of Lisp for ~~exec~~ tar & Genprobgens

(In abstract, just deal w. INV probs). Then: P.D. based on Reference Unc. Using "better" P.D.'s result in faster solns. We would like to adaptively modify P.D. as it was used to solve problems, so as INV solved more of them, the old probs would be easier to solve & new problems that had been before too time consuming, could now be practically accessible to the system.

What I have found is a strong step toward an updating algm. It enables us to take several disparate kinds of data, used in very different kinds of problems, and use this data to update the P.D. that is used to Guide Search. This is only ½ of the solution to the problem: The final problem becomes that of finding short codes for a particular batch of data: Before the present development, it was possible to update a P.D. that was used for INV. problems. It was not clear how to update a unified P.D. that would be used for both DE & INV problems.

optimum search procedure for tasks

In 1972 Leonid Levin devised a ~~solution for~~ all inversion problems.

Inversion problems are the P and NP problems of computational complexity theory.

They involve finding a string or number satisfying a stated property.

Soon after I met him in 1979, he was able to generalize this

technique to resource limited optimization problems. Since just about

all problems in Science and Math are at one of these two classes, this would seem

to be a great breakthrough ~~following~~ ~~If~~ indeed!

His procedure was optimum in the sense that there existed a technique for

solving problems of size n, with time $F(n)$, then the search would

take no more time than $K \cdot F(n)$, where K was a constant independent of n.

Levin's search was guided by a universal probability distribution.

There are many such distributions. If one used a distribution

appropriate for the problem, the constant factor K would be small, and

the solution close to optimum. In most cases, however, the constant

is astronomically large. How do we choose a probability distribution?

I will discuss a kind of solution to this problem: "The Mixed Corpus Procedure". It accomplishes several things:

It enables us to use data on previous problems solved as well as other kinds of data with contextual relevance. It makes it possible for us to use a single conditional probability distribution to guide the search.

The solution I have found, is an "adaptive" probability distribution. We start by using Levin's method to solve simple problems, which it is able to do in reasonable time. These problem solutions are then used to modify the probability distribution, so it can then be used to solve harder problems. What it does is, the solutions are then used to modify the probability distribution, and so on. By using a suitable sequence of problems, the system should be brought to a high level of skill in problem solving. → 23.01

-29 Some researchers suggested this method of modifying the probability distribution is able to used data that is not directly related to problem solving. Also the results of solution of optimization problems can affect the effects of the distribution on solving inversion problems.

The conditional probability distribution is a unified expression of all of the information we put into it.

Information about inversion problems can be used to help solve optimization problems and vice versa. It is also possible to put into the prob., information not directly related to problems it has solved.

2/10/00 BVLG

9 Demo.exe · 21 MBy C:\Temp\SM342.exe
Stmtmp9.

.01: 22.39 The technique for updating the probability distribution is

SN What is a universal cond. p.d.?

= $p_u(\alpha, \beta)$ α is condition, $p(\alpha, \beta)$ is probability of β in view of α.
for any fixed α, $p(\alpha, \beta)$ is a universal p.d. wrt β.

I think that's all it is — no other constraints: α can be narrowly restricted.
(But α is a finite string (usually)). I don't know how to do that if
α is an infinite string.

In Levin's original formulation, he used the same universal distribution
for guiding the search. The distribution, was independent of the problem type — so while
for some problems [illegible] "would be impractically large — very distant from optimum, for other
types of problems, it would be possible to do that with a bias by making the universal distribution
a function of the problem description — becomes a conditional universal
probability distribution.

We use a conditional UPD: one that adapts itself to each problem —
The distribution will depend, not only on the problem to
be solved, but upon all of the previous problems. What system?
That were solved with the distribution that were solved by the system"

We will show how an adaptive universal conditional P.D.
can overcome the difficulty: [illegible]

The system envisaged starts with a simple set of problems
that the human algorithm solves easily. These solutions to those early problems
are used to modify the CPD, so it can solve more difficult
problems. Similarly, those solutions are fed back to modify the p.d. and so on.
By using a suitably chosen sequence of problems of increasing difficulty,
a system of great problem solving skill can be developed.

I will show how the p.d. is updated, using
previously solved problems and other kinds of information that we might
use to speed up the learning process.

[Need beginning, mention that I've a 12-page course, established all
kinds of probs occurring in Math, Sci, Engrng.

→26.01

GA

.01 21.15 GA The thing about Pascal operators cands for Env. prob. solvers! They are operators, (≡ functions) so we can easily generate Boom as trees. Also, we can use same set of cands to solve O2 probs! — i.e. each Gas as in prob 1. O2 prob d'ery ∂ its output is 1 or a more trials.

The cands in the GA can each be stochastic operators! (Ro for some cases Ro is bad; we have repeated sampling (with replacement) of hy pc

.07 ← This is what regular search deals w. well. → 30.01 bands that could be failures.)

More on GA's wrong INV probs: "New Approaches" 1.31 (1/20/00)

.09 (Ibid → 1.18 ← Good! .19) Simon/Newell

Also Note use of "GPS" to convert INV to ≈ O2, But w. Vaccari Gorc. — which "localizes" what has to be "fixed".

Also Note: Some problems are both INV & O2: eg. we have to find an (in Logic) solⁿ. of a certain eq. (INV part); we want to smallest such solⁿ. we can find in time T. (The O2 part). We want an automobile design satisfying certain specs (INV); we want minimum cost (O2). Money

---

Re: Starting work on AI texts tsq. : Try to keep the tsq. ∂ t.

.13 The responses "in English" as long as poss! → 31.13

.19. (Cog) Re: Cog: Here we have an I/O device: Expert = probs; Output = solns. The GA tunes to improve this device. The device itself can work O2 probs a/o INV probs: we can set it up to do any kinds of tasks for which we have a Gorc. A possl trouble: the Machine being improved is quite complex & can take a long time for a response! It would seem difft for GA to work on "improving" it. → 25.11

2/12/00 Saul

Main difrnce betw using ALP & GA is the form of the P.D. & the updating technique. (The ALP could use GA's or any fanciest PD (≡ GPD). So ALP should be able to tell how the GA should optimize both form of P.D. (which includes mutation & crossover & any other methods used to get next generation) & the updating technique. The GA seems to combine the P.D. w. the updating.

.30 Perhaps it could "decomp" GA's P.D. as the responses of the Rosenbloom generation only. → "Updating = reflection of how to current gen to present generation, environment, Gorc"

GA updating: Cands of hy G have useful "parts" [hvr. "parts" can be defined in many (parallel & equally valid) ways. — Just breaking operations into "compact" sets operation is an easy, but not really best, way]

.35 So Ap take the present Generation of cands & Rel assoc Gores, & induces a p.d. of (cand down → G, pc.) of next generation. GA uses same data to get just p.d. of cands of next generation. (no "G") — The Room is attempt to get cands of next generation to be of hy G.

.12 2.8.40 : Actually its easy to get "Alp's pd."!! Just randomly construct cands & evaluate them!
What Alp should be doing is SGA: Get a G defining fcn for the next generation in a way that makes selection of hy G cands very easy.

In a sense, GA is "T poor man's SGA". It simply tries to make the whole next generation (or as much as possible of it) of hy G. The main advantage of SGA: It uses the info of the past generation more efficiently! Cands of low G also contribute info. Also SGA can use any copy (short codes) in the data to help make P.D. Perhaps the comments to the same objection to Sim Anneal: That it doesn't use past info on previous trials very efficiently.

Ordinary GA may be more restricted.

.11 : 24.23 : Hm. Usually only a small part of this (E/O Domain machine) is being ("improved" "worked on")! And we tend to test only part of its all over response. (This idea may be in a recent w. e "(4 year orders" heuristic.)

.20 If, via GA (or any other method!) we develop a system that can work
.21 many, disparate types of problems (both Dnr's or mixtures), & can it is likely that the system would have "figured out" just what the goal was ... the relation between the problem & cons, (Dnr eq. constraint eq.) & the trials desired, or the desired features of the cands. — and in view of this goal, it would be able to "reason out" good cands! I want to design the system so it is likely that it would make discoveries of this kind, but not an A.I. system! The important idea here is that such a discovery would help the mean G of the next generation a great deal!

.27 SN The way I deal w. OZ problem updating as a MCT (Next Computation Thm) — I may be able to get expected cc's of various cands! This would enable me to use GHT1 (Goal Boundedness Thm #1 for E/O) directly, which could be a lot better than Lynch (particularly if there are many "flat" parts
.29 of the P.D.)

.31 VERY GENERAL comments on my system: When I'm doing "updating" via info of past successes/failures of parts of the system, I need not have as narrow goals as MCT. It's possible to use that info to get probably that a good cand problem solver would solve a specific problem w. cc ≤ c. From this, we
.35 can use GHT1 (Goal Boundedness Thm for E/O) to order the trials — perhaps WON Theorem. In general, the goal of the updating is to fix it so the system will solve problems "faster". The info used by MCT is simple, but we might could use much different output.

30.20 spent

.01: 24.07 In GA where we try to get population of problem solving operators, We'd like to be able to learn "reference" a set of operators that were (likely to be "good" for a given [redacted] input problem. — So we could "save" on a HDD (or a very large DVD ROM ≐ 100GBy!) many operators that were somehow indexed — so if we had a particular problem type, the relevant operators could be readily selected as an "initial population" to work on the problem.

— This is Mindful of Selfridges "Pandemonium" & Minsky's "Soc. of Mind".

Characterizing in put problems, tho, would be quite difficl! Each operator could have a "sub" I/O: Given the problem/data, it outputs a number telling how useful it might be as one of a/starting population for a GA such.

.20: 25.40 [redacted] perhaps less ef. approach of 25.25–.35: No doubt v.g., but I want to get to show on the road as soon as possl.: If I had a system that was more or less working, using conventional Lisch, I could modify it later (changes to Update algs & the such as) in view of 25.25–.35

Try writing a TSQ for elementary Algebra; But make the emphasis TSQ i.e. make the conc. net. See if I can move from one part of the net to another (not nec. starting at the Bottom).

On Lisch for Inv. probs: In general, this would seem to usually (almost always) involve a very large "it". We really need to get in mus/class. I think the main idea was that all heuristics could be expressed as modifns of the pd. — Which amounts to a conditional pd, in which the problem data (condition) influences the search (via the pd.) So, are all heuristics equivalent to modifying the pd. in Lisch? If so, I think the MCT method of updating the P.D. should be able to obtain improvements any & all heuristics. (May be not the "Quick & dirty" cross. — I think WON may deal well in "Quick & dirty".)

N.B. Choosing a Univ. P.D. is like choosing an approximate soln. of the problem. This choice can be arbitrarily good (or bad!). It can be so good that the soln. has Pc ≈ 1. (with small CC): So we have a 2 step soln. to any problem: Step 1: Get Cond. prob. distrib. Step 2: Lisch.

.01 SN1 In the Z138 report, I discuss random input as a strong bias, so pc of 0 was ϵ; pc of 1 was $1-\epsilon$; ϵ small, so ~ ϵ bits per input symbol. $\epsilon^{1-\epsilon} = \epsilon^{1-\epsilon} = \epsilon^{1+\epsilon\ln(1-\epsilon)} = \epsilon^{1+\epsilon(-\epsilon)}$ more like $\epsilon \ln \epsilon$ bits/symbol,

$\lim_{\epsilon \to 0} \epsilon \ln \epsilon = 0$; but $\to = \epsilon(\ln \epsilon + \epsilon - 1)$

So $\ln \epsilon$ is dominant term $\to x \epsilon \ln \epsilon$.

So the "granularity" is $(\epsilon \ln \epsilon)^{(\ln \epsilon)^{-1}}$ which can be as small as one likes; Picking the shortest code doesn't suffer from the excess granularity of $\epsilon = \frac{1}{2}$, but still, probably it codes with much ↑ precision much!

SN2 A UMC w. probabilistic state transition matrix; is equiv. to a deterministic machine w. some random input, but an aux deterministic input tape. is a stochastic operator. Using stochastic state xtn. table, one can simulate any stoch operator

.13 : ~~24.18~~ : English for TSR, etc. Make big conc. part: Make a rather "Grand Scheme", plan slowly fill in details:

top "Reading Math" — Reading Algebra — understanding Algebra

Understanding Alg: ordered list of problems solvable by TM. : Ordered list of definitions & concepts.

ES I want to see how easy it is to (a) teach a useful conc (i.e. its definition) compare c's for each (b) U.S. having TM discover the conc. because of its usefulness for problems [solved]. way

.19 A possible ordering: ordering Alg. Alg notation understanding. RPM s/o ordinary Alg. notation. Solving (linear, then quadratic, then cubic (& perhaps 4th) eqns)

[Possibly solving higher order eqns after being given suitable functions.]

What about the various cases treated in Elementary Alg book)

Several linear eqns: Solve some n.l. eqns. Solve some systems of n.l. eqns.

[Introduction of trig & analytic exp, ln, functions. Identities; Prove various identities.

Diff. Calculus: (learn What derivatives of various funcs are & various combination rules (linear; chain rule) (TM need not know about Limits)

.29 Int. calc. — an version of integration. [lots of problems of various % of difficulty] 33.28

.19 to .29 is a rough amount of Material. Problems in Integration, solving eqns can be arbitrarily difficult, yet understandable by TM (not nicely solvable!) .19 to .29 could be done in parts — But mainly, I'm interested in finding ways to teach TM stuff easily. — So, in this

.35 sense, doing all of .19–.29 could be a useful serious challenge!

Some other problems within .29–.39: putting (or more) eqns into the form of a linear eqn & is solvable.

It is certainly possible for TM to understand .19–.29 rather well, & not understand anything else about Math!

.01 Can I do $31.29 - .29$ w/o complex nos.? I think so. I think the MATH community was, historically able to do that w.o. complex nos. — Pro: Complex nos. make it a lot simpler for some things.

e.g. $e^{ix} = \cos x + i \sin x$: So I'd have to define only $\exp(x)$ & $\ln(x)$ for complex nos. & get all the rest from identities & trig formulas.

However, I may want to have TM learn minimal $31.19 - .29$ because it's the pb., I could start teaching TM "English" (Pro: a reason to do more or $31.19 - .29$ is $31.33 - .35$)

(if $\exp(x)$ is defined, can I $\ln(x)$? inverse of $\exp(x)$?)

An imp. Q, is how to present info to T.M., Lisp seems like a good defining way. (used by Lisp). Also maybe Forth — Gregory Chaitin's "Little Lisp". Lisp has (very probably) been used for all of the problems I've listed — so it must have ways to represent them.

.12 Another poss'y is to use my "General Functions Language": it might be easy to get it run very fast on a machine.

MAPLE V has a notation for various symbolic operations:

e.g. `solve(x, x+1=0)` $= -1$

Also: an Olde Project: Work out a Concept for A large, good set of OT's (Often techniques). I do have a big list of OT's somewhere (Search papers?) — Get them s/o, make a new list. — I have in 13 of them.

1) SGA. 2) GA 3) for single probs: Hillclimbing w. local linear # or n dim paraboloid model.
4). Sim. anneal(5) GPS (This is what we're using on OT···it's a LNU solver).
6) Lsrch? 7) Tabu Srch 8) I surely have a book on Optim. Techniques!
9) Are the methods used to solve induction problems (finding short codes) a special OT?

The main goal I have was to get TM to acquire this concept, so it could usefully work on the problem of improving OT's.

In Algebra: (& probly most of TM), the idea was to get TM to learn "Functions" — Definitions of functions (& concs, which are expressable as functions), using a "Functional Lang" (Like Lisp or that Functions I invented (see p.32) Form of data: $\{\sum$ Name of function, $\sum$ input of function (usually a list), $\sum$ output of function$\}$)

Since Induction is primitive, we can make answers in input w.o. guess & check

One of great Advantages of MCT: That by using "hints" of all kinds (general info in various forms) I can speed up training of TM's (of: & because the input to TM is always in essentially inductive (problistic) form, it always has to get the acquired info into its own internal (layer — so it can use it.

In the "Project" of 31.19-.29 I could get TM to do a minimal set of problems so it would do only (th) Pythagorean. Such a machine would not be able to acquire the heuristics for solving (linear eqs → quadratic eqs → cubic eqs & Pythagoras) because this heuristic would not be useful (much) in the rest of the corpus.

For most heuristics, TM will not have (a big enough, diverse enough) corpus to derive those heuristics. The heuristics will have to be "fine-tuned" on the basis of solving TM/many probs w. solns. Not use certain heuristics. In general, many impt. heuristics were derived by the Math community as a whole, over many centuries of problem solving experience.

One way this can occur: A unsolved problem is labeled "Hard" by the Sci Community. If it is eventually solved, the heuristics used in its soln. are given much wt. To the extent that the heuristics are known. (our heuristics are given much wt. Most are those that led to ones useful for solving that hard problem.

One problem (weakpt. balance) of Soarb. learning of AI notation, was all junk was in storage, unindexed; I will have to fix this — w.o. indexing, IT becomes too expensive to access storage.

I want to start w. minimal system, then add in more kinds of info/problems Bader, I started w. AI notation learning; then maybe soln. of linear eqns.

.19 Then I tried to get equiv. of "Learning Laws of Algebra." (2×B = b×2, etc.) I'm not so sure of wisdom of doing .19 — At most stage, TM was in Thinking Mathematics. A better notion would be to learn to manipulate alg. eqs. to put them into solvable forms — Maybe solve some non-linear eqns w/o fancy algs.

— At first, put every thing into a non-indexed memory: then reduce cost by introducing linear indexing of memory.

READ over that SAARB stuff! Maybe index it to some extent. Perhaps use Laptop for indexing — so I can easily insert lines into index form. Maybe use spreadsheet?

.28 ; 31.29 Some other possible directions of extension: sin(x), cos(x), their relationship. Then Complex nos. Soln. of polys in complex nos. — $e^{ix} = \cos x + i \sin x$ $\ln x$; e^x. Solution of literal eqns (like solve for x in $ax + b = c$)

.31 Numerical solns. of eqs (real, complex) w.r.t. literal solns. — relation between 1 & 2.

So 31.19-.29 is a variety of areas covering a large part of Math. More than enough of the TM to learn to solve serious problems. So .28-.31 could be added to that list.

Next, there are implications that I may or may not want (need) TM to learn: e.g. 1) The idea that 3, 3+7, 4/2 are "values" & "values" can be coordinated via "functions"

2) Existence idea that we can solve an eq. by pretending we know what x is, & unwinding the eqns.

3) The solvability heuristics used for solving linear → quad → cubic eqns.

4) The GPS heuristic: The idea of a vector space: On a more elementary level: it is to absorb, to appreciate, that there are analogies between [illegible]

2/22/00 Bulg:

Psy livns: 10g/day

.01: Meaningful Alg expressions is something I'm good at, & it requires little knowledge of McR; so maybe try to get TM to learn this.

→ So continue making lists of Reprs to try. Search RFD on partial order / partial order info & its lists.

One impt idea is to not use a simple computer alg. notation (Gardner, but to use "functions" instead. Say Add, Mul, Sub, dN are funcs builtinto the machine. Then we have + 3, 7 → 10

So "+" is an operator that we want TM to simulate. 3, 7 are the currently 10 inputs. TM sees 3 terms (+ 3, 7, are very random nos.) Note for this set of examples "Add" will simulate + (i.e. all examples this far.

If we now give −, 4, 8 → −4, it by passing on many examples; The operators Add & Sub work each about ½ the time. We should, now, be able to correlate + w. "Add" & − w. "Sub" at 100% accuracy. (I think Sal 86 described this). Similarly, × & ÷ can be found.

At this point, if we use alg. notation that is explained to TM, it knows that X(+, 3, 1, 8) means that "+" operates on 3, 1, & results are then operated on by "X". I think it not unreasonable that we should explain what the notation means (of course this explanation could go forward) tell TM that "+" means "Add", "−" means "Sub", etc!

So at this pt, TM can evaluate any alg. expressions, if it knows what the functions are.

I had hinted that TM should have concept of "quantity".

Also, in general, I want to be able to teach TM any concs. that I can think of. Most concs will require a large conc. net to acquire them properly. — but I should be able to draw up such a conc. net by "hovering downward" from the "top conc".

.26 In General, I want to make (partially ordered) lists of probs, concs, like 31.19-.29; 33.28-.31). T. TSQ should be a partial sequences, & In English! T. details of realization, & exactly, to be used, should not be (merely) part of Rep's work.

So again: The idea of an "In English" TSQ: deals all probs, solns "in English" — try to be non-committal about the underlying reps for TM to use: By solns: I mean conc's Heuristics that are adequate. Occasionally > 1 soln. If the concs & its heurs are not "primitive", find ways to "insert" them. Take advantage of MET's ability to "insert" concs, in many ways. Perhaps list & detail ways concs/heurs can be "inserted".

The "Invertible Substitution Heur" (used in linear → quad → cubic eq. solns) seems to use some "Logical Reasoning". — So I'll have to find way to get Rep in to the sys soon.

Seem to have lost TM 98 131-to-144 (perhaps Yellow paper?)

.01 : 7.17 : Any way, T. now does about MCT, is that it's really a very simple process.

.02 of unordered set induction method/proof.

T. situation differs from 4. unordered set of finite strings, in that

.03 Each of 6. finite strings can be (a) a finite string w. an ending (stop state)

(b) a finite string w. an unendy "stop" (i.e. prefix of a time series)

(c) data tuples like (OZ problem, OT_j, CC_k, $Gorc_g$) that are

.06 to be used to update 4. OZ (search.

In 4. case of (c), we end up w. a P.d. that any OT with any problem w. any CC will get any Gorc. We (can't operate this (which is not part of an induction or part of MCT) to see probly that any OT will, for a given CC, give max G for a given OZ problem.

.14 (d) A large data base (like an encyclopedia or WWW) — that is regarded as a single finite string. (7.26)

The only kind of finite string that is different from 4. ones in .02, is that "acceptable code" for prefixes of a time series includes codes for any string w. that type (also, one may have part of a finite object (as well as part of an infinite object)

The "adequate proof" of 4. unordered set Rem is 98TM(10.01-111.46)

.20 The All-way Scheme for TM: Steady state: TM is Given problem: TM solves problem w. search using present P.D. : TM then spends time "updating" & "improving" 4. P.D. The amounts of time spent on updating & on 4. various

.23 aspects of "improving 4. P.D." are "admin params. that I must decide upon.

When TM is very smart, these "admin params" must be carefully

.25 controlled to prevent TM from causing Trouble [DOW, CIMP, ect.]

So, w. .20-.25 & 4. TS & outline of 31.19-29 · 33.28-31, I should be well on my way: Note 34.26 on informal use of English.

.29 So: T. corpus simply consists of a set of unordered finite strings:

each string is of 1 of 3 types (a) complete object (b) prefix of complete object

(c) prefix of infinite object (I think (b) & (c) do need to be different)

Each of these finite strings can be any one of (.03-.06). (perhaps 4. string can represent an unordered set of finite strings as well)

As part of 4. corpus, is all of TM's "process" up to now. — which is Enormous!

— So it is clear that much of 4. corpus must be either compressed (lossless or lossy) or deleted (by "lossy compression" usually) and/or indexed so most likely relevant parts will be used.

I) changes of states.

II) [illegible]

Internet Service Provider
ISP

ISP's [redacted] can be my ability to be integrate. My [redacted] ability to [redacted] look up [redacted] number of integrals. Many more or [redacted] skills that I have

Perhaps the best approach to TSQ design: Pick a "Intellectual status pt." for TM. Tell what it knows, what skills it has. Then work out ONCE looking at that status point. Do this all "in English"; then go over and flesh it out in more & more detail. Then consider various ways to execute code — by problems & "Hints" "by telling", by analysing TM's state as computing it CJS to learn a path-wise theory from some data — & if CTS is too large, just give it to TM as rule-of-thumb & such.

The "Int status pt" may be well beyond what I need to begin teaching TM "English". This is because I want freedom framing TM.

Learning English may be quite different from TM's earlier learning, in that in its earlier learning, the corpus was already "parsed". In learning English, both the Grammar & the parsing have to be created. "Hopefully" I'd be able to modify ZYAR to do this. Wolfe's methods may be suggestive: [redacted] "Multiple alignment"

Another possible Branch would be to look at Lenat's heuristics in "AM" & see how far I could go with them.

Another Q is about TM's learning in a given model for a "Black box" I/O source. TM is allowed any input it likes & can look at its output. The Game is unclear. It's as unclear as that of a scientist trying to make a good Model of the Scientific World.

A TM "out of control" could view OW as such a Black box. If TM had certain goals, how would it learn to see if OW could be used to help Toward those Goals.

A more illusive Goal for TM: It look at this Black Box and it wants to know enough about its I/O characteristics, to take advantage of them: to use it to help solve its problems. The "Black Box" could be OW, or a [redacted] Universal General purpose computer, or some Pgm (software) that TM might find "useful." . . . . . ABCDEFG—ABC

A Child perhaps, looks at the world w "curiousity" wanting to know its I/O behavior — But only certain aspects of its I/O behavior are "interesting" to the child at each pt. in his "career".

One aspect of "Curiousity" TM wants to do an input that has maximally "surprising" output! This means TM would have predicted output badly — So this data pt is very useful.

This whole Q does seem closely related to eval structure of the SCI community — To do an experiment w. an output of max variance w best current Model (theory)

2/25/00 Bulg : ..... TSQ .....

1.43V 2.25.00

.01 A common case: We have 2 models that differ & we want to do experiment that has maximally distinct expected outcomes for the 2 theories. ["A "Decisive Experiment"]

Trouble is, OW (& Science in general) is so Big, that, it's irrelevant to what part of it one want "Decisive Experiments" for. It's an "Administrative Param" — Are we now interested in Black holes or in Cancer Cures?

A real problem here would be: TM finds a bunch of utility pgms on y:\work. How much time should he spend analysing them to see if any of them are useful for him (I have this problem w. SIMTEL). MAPLEV, Symb. Math pgms.

.11 2.27.00

.12 ; 35.23 So: The way it works: [Including "TRACES" of All TM's activity up to now] → 38.10 It has this corpus w. various types of unordered "objects" (35.28). TM works on this set of objects to "improve G.P.D." (This is an OZ problem). Then, usually but for it works on a OZ problem, the Global pd has to be partly modified for use in OZ problems (The MCT deals w. just how this has to be done ... The pd on (probes, OTs, ccs, Goals) has to be integrated so that to probe that OTj working probe for cci will obtain the best Goal.

If the "Improvement of G.P.D." is a OZ problem, the exact nature of that problem has to be spelled out in the "Admin. params".

2 Main areas of work on TSQ's:

.122 (back) 1) General: Start w. top-down analysis: (36.0V-11) Do the Initial Status ISP Count — I.e. interactively state that I want an "Information theory" TM to learn.

.24 2) OTj's: Try to "factor" the set of interesting OT's into a good (those such) pair of cases. — Then devise suitable TSQ or "Thy env't" — not merely "sequences" (?). I guess the idea of "TNG environment" is that the exact sequence/order of input into it is not very critical. (I hope!).

② is "harder" in the sense that the problems to be solved are usually very diff. On the other hand ① is more specific — i.e. I have this set of 12 or 27 OT's that I want to "factor". In ①, I haven't yet figured out just what ISP I want.

As for D(.29) I was merely thinking of Algebra (linear then poly from (n, etc etc) but another area would be Programming: Problems in Pgmg could be of sort to Minzec, Max pc. — This is (.29) what (2) is about. A different aspect of pgmg (classical "Automatic Pgmg") Given some specs of a pgm — to construct the pgm: A Q is: "How exactly pgms initially derived?".

— Possibly as INV/OZ problems? Inductive elem: Given (contain I/O pairs) the machine hyp pgm that extrapolates that data. → 38.01

2·27·00 Bulg

.01 ; 37.40 Another big issue: Translate a pgm from 1 machine to another so that speed is max, yet (perhaps) amt. of code is small. One diffy is that the translation will be "BUGS and all"! A typ. x/ltd pgm could be much harder for a human to debug, since it was probly written using quite different concepts from human pgmrs have. However in debugging, a pgmr usually uses a "local model" for the BUG — so he doesn't have to understand the whole pgm. TM can do debugging by being given sets of (Buggy Incorrect) v.s. correct responses & be asked to construct minimal pgm. to bring this about.

some characteristics of "Enormous amt. of data": Internet; SM; Weather data;

.10 ; 37.11 : This "TRACE" is an enormous amt. of data! TM will have to compress it — Lossless or Lossy — partly by detecting data. Too much of TM's data are easily compressed, since they are derived by a "simple" pgms. [One of TM's normal tasks is to recompress such data — look in case of lossy compression is very small amt.

→ Also, suitable ordering of data, some know whether to use the various problems

SN I was concerned w. TM's problem of finding common sub trees in large trees. Actually, this problem may never occur: The functions that TM will be looking at will all be formed of a few (previously defined functions — essentially "short pgms" — since longer ones would take too long to discover!

→ SN A perhaps very "easy" way to get TSQ's: Look at solns. to problems by early workers in A.I. Then find ways for TM to learn to work those probs! Find concepts for them.

→ Maybe look in "Encyc. of AI" (3 vols) = Also, more recent Encyc. of A.I. Also fragment. "MIT Encyc. of Cognitive Sciences".

→ Another Approach to TM: Actually write a (very) pgm and try it out on a TSQ — like a human teacher trying to teach a child, w.o. any detailed model of the child's mind)

Maybe solved!

→ Given Koza's chl design problems. ① Just how was the problem described to the system? ② How could one use the probs as a TSQ?

The population solving a problem can be viewed as a probabilistic set of operators; a stochastic lang. However, TM has been able to look at a problem & get some ideas about what kind of stoch lang. (e.g. what sort of operators to use & to what Prim PC's should be).

It would be good to look at the solns. Koza obtained & see just how they could be somehow connected to the problems they solved — By "soln" I mean the stochastic lang. of operators that was realized by the population that solved the problem.

Apparently, for most of Koza's problems (chl design) the given Machine Lang. for deriving all conceivable circuits, is a "Black Box" Gate. In such a case, all minimal designs of the problems were discovered. However, after running some experiments to

xcamera.
1'800-442-5709

.01 i "Getting on the Hill", TM would be able to see some difficulties in its problems
i begin to modify its P.D. accordingly

Another Big Area of Work on this approach to TM is the idea of "Summarizing Machines"
I think what a Summary Mchn. does (as per Rem 6. former ones in Rem Sol 99 paper)
It finds one or more regularities in a corpus & makes some kind of a production
machanism based on those regs. The Mechanism could simply produce a
P.D. on extensions of the corpus &/or find a set of very likely extensions,
etc. In the case of a "conditional P.D." it could act as a stochastic
f/o operator — either in Monte Carlo mode &/or "Most-likely" output &/or pd on
outputs.

"Updating" a "Summary Machine" w. new data: This can be done by leaving
general model invariant & updating continuous params: say multi param regression
or stoch Grammars: Or [redacted] may add or change some params
in regression or add/change partofspeech in stoch grammars.

I was Concerned that TM wouldn't be able to get the idea of "sequenced
operations": It could learn this by being given "Traces" of successful
searches for Soln. [redacted]

A Possible General Philosophy of TM design: That initially, problems
would be solved by collaboration between TM & User. Then as TM matured,
"User would slowly "bow out". Like a Parent and child.

In the olddays, there were 2 important subgoals for TM:
(1) Ability to do OZ probs (i.e. self-improvement) well
(2) " " understand English text.

One Important idea in TSG design that I've never really tried:
Top down analysis: Have top problem &/or Intellectual status point (ISP)
Then work out Conc. nets. —— Rem in the Sol 89 paper — that
was the main - expected way to do it. [The Factory of that set of
12 or 17 or whatever OT's could be in that direction.]

Perhaps draw up maybe 5 or so General TSG approaches
& work on them simultaneously, until it got more of an idea as
to which would be best to start on. All of them should eventually
be usable. —— Also try to find work & summaries of
work on various TSG types. (Algebraic/Matrix)

(SN) on Trying for "Evaluation" function: Do it so that TM
is Given opportunity to see if an incorrect: That it is rewarded by
evaluating expressions & expressions that contain those expressions
&/or Rem techniques like a kind of hill-climbing (maybe GPS-type "hill".)

On differences between GA & ALP using conditional pc & Lisp as opposed to "Hy Level" A.I. [illegible]

The idea is — how do we feed a TSQ into the system & how does the system remember good concepts from earlier parts of the ts.?

In GA's, I was thinking that the system "memory" would be in the population of cands & their progys. Also (perhaps) in the way the mutations/"crossovers" were done. As the amt. of info in the system grows, the no. of cands retained would grow. — But also, the diff'ty in finding a cand relevant to the present problem would grow!

[illegible] the [illegible] pc (cond. pc) approach

In the [illegible] approach, we just generate cands in (new, a stock) lang & use cpc ← cond. pc order the stock lang. in view of the observed G's of the cands.

This would not work well w a TSQ.

12 [illegible] For Lisp to do (w. a TSQ, we use a cpc (cond. pc) or a stock operator; The input is the problem, the (stock) output are cands for solving the problem — The operator is modified by the P.B. of G's of cands, as before. So we end up w. a stock grammar, for each problem (after the prob has been solved).

Then, we eventually have a corpus of (problem, soln grammar) pairs, & we want to extrapolate this to guess at a good initial (stock) grammar for a new problem.

This way, we don't have to retain an enormous set of "good" cands!

We retain the {problem, stock Gram} set and we somehow use GA or ALP or Lisp to derive a predictive relation betw. the pairs.

For a (conventional) GA to deal w. a TSQ: We look at the population of cands (at various stages of evolving) at times of soln. (Hopefully, these cands have certain properties in common (These cands include the final soln used.)

We then want the system to be able to map the set of problems into the set of key G cands for each problem. As Koza uses various Srtns, (or Macros — a dangerous form of Srtn), the system should be able to recognize that certain "Srtns" (or Macros) are common to many problems, and, ideally, recognize when Srtns would be appropriate to what problems. Also it should be able to characterize each problem type & for each type, correlate various macros/Srtns, that are likely to be relevant!

29 I could look at Koza's probs & going in deriving filters & analogue Tc's. — See how he did it. Init. his initial "rules" were for each G-th problem solved. It may be that the initial population & functions used, would be rather diff for a person to derive for a particular problem.

33 In analyzing Normal GA, consider it as an approximation attempt of SGA

→ Also: Look at SGA from ALP pt. of view, — particularly consider the analysis of "operations" used in the work on MCT.

T. Poggio does (is like "A frontend for GA"; & in MetaRegularization TM) Its a very NON-EL way to solve a problem.

The frontend and Backend are seamlessly integrated; I think Poggio does much less so. We have divided the TM problem into 2 parts: (1) the "frontend"

.01 Which chooses a/o constructs a suitable GA setup for a particular problem, and (2) the running of the GA setup to solve the problem.

A possible part of the front end: the operations of 40.29 - .32.
Hmm, I'd really like a much less EA/ way to integrate the front end & transfer to GA
One way would be for the "TM$_2$" aspect (i.e. front end) to be continually working on the problem of mapping known sets of solved problems into closer
→ ~~Pairs~~ to their "known" solns. By "closer" I mean less CC to get to close to the original soln. (or much closer for same CC . . . .
This is a partial ordering, but it can readily be made to a linear ordering by
.11 suitable weighting.) → .33 looks like a v.g. way to look at this!
We can avoid a.H. solns. using the usual coderack/PG considerations.
.13 So one way of looking at the problem: we want to find a function (operator)
that maps input prob. descrn. into a set of seeds, or seed components (crtns, macros
that speed up soln of ~~that~~ problem by GA. (or may to not speed up, but to CC of soln.)
Some ways this operator/function could work: it would analyse the functions
~~of~~ & macros in the prob. descrn: it would look at the oz criterions
.18 functions(structures.
.19 A possl. advantage of this GA method of doing TM$_2$ is that it might
make it easier to start w. a machine that can already work diff problems,
& improve it.
.22 An Avgt against this: If we start w. a set of diff. problems, the
prob descrns. will be large a/o diff to analyse — diff for TM to discr. the important factors in their solns.
So it looks like whichever way we do it we may have to start w. simpler
probs & work up to more complex ones.
A slightly less E approach for GA: That 2 processes are always going on simultly:
.28 1) Generation, by GA (TM$_2$) of new candidate strings, macros, functions, code fr. from the set of ~~solved~~ known
.29 2) Re use of these new & other abstractions, macros, candidate solns to Bob/soln pairs.
the new problem. ||
When a new problem comes in, first ① is done only, until there is enough
stuff around for ② to work.
.32 → Perhaps another way to look at it: for each (set) prob. descrn, we have a set of cands
— say, the complete, or best, set of cands used to solve the problem. The great corpus is the
.35 set of past problem descrns, each paired w. the seq. of trials that led to heavy G ("soln").
.36 Then, W. new input problem P_2, we want the "TM$_2$" (a GA) to induce a set of cands, cands, each
.37 that are of high expected value for P_2.
.38 So then, new cands for P_2 are obtained in 2 ways (by 2 simultl. processes): ① from the corpus
.39 of .33 - .35 ② from the "present" total set of cands for P_2.

.01 41.38-39 summarize how the GA would work. It is somewhat normal ("Holistic").
A disadvantage of GA is that it can only work OL probs: While, by far, most problems of interest, are OL problems.

.02 [Now it may well be that all solvable INV probs. are solved by getting them into OL form. e.g. GPS puts Inv. probs into a kind of OL form — (But the G function is a vector !) → (see 43.13-17; 44.09 is very good) →
↳ The 41.33-40 looks V.G., a critical Qn! Just how the (presumably GA) Thm of 41.36-37 works — how it gets by G-cands from P_i plus the corpus of 41.33-35
Well, it is an induction problem: See 41.13-18 for small cases.
Seems to be a 3 component vector Garc; We want an operator w. a problem decrn as input, that generates many hy G (also concs & cands). It should do this at small cc & hy pc. The operator (probably) has to "parse" the problem decrn. → .18

.13 To some extent, the problem decrn. is automatically parsed: It must contain a pgm that evaluates a cand. Often the Garc will consist of a rigid constraint, plus a continuous G function over cands satisfying the constraint. If a cand doesn't satisfy the constraint its G = -∞. — But sometimes it is useful to make one or all constraints soft — a gradual ↓ from high medium to low (not -∞) values.

.18; .13 → In normal OL problems, one has a fixed cc (variable cc in "Anytime" probs) & a single Garc; G. A "soft constraint" could be like pc.
The single problem occurs in ordinary GA, in which we want to generate cands with expected G in short time. (Mathematically I guess "expected G" — This assumes finite sized Garcs.)
So in both "Regular GA" part & "Induction from previous solns", one has single Garc; Highest G may in either given cc (or "Anytime") ("Highest Gmax" can be done (by: one need not use "expected values") — See Wompum MCT & OL probs

.25 → Using the GA method: One can use it for both regular part & reducing hy G cands from previous [problem, [cand sets]...] pair corpi. In fact, one can obtain a "Grammar". That is a function of the problem decrn. This looks like a fairly Normal soln. to the problem: it is seeming very close to MCT methods. As one expands more cc, the Grammar becomes better & better at finding hy G cands by function of problem decrn.
Superficially, .25-.28 doesn't seem to have anything to do w. ALP ! — Except, perhaps fitting Grammars to data (a rather BIG ("any thing")!)

.25-.28 is very close to MTC: So can & perhaps should extend .25-.28
So it includes Inv. problems: & treats by (perhaps) GA regular GA to work Inv probs (Nice, note .04 !)

.34 (def) The problem of "horizontal induction" (= (.25 R), (.26 L)) is a def'n. Perhaps different TAMS wording or different (or ∼) domains should share some of their cones. ! Presumably, they would have to have similar GA systems (i.e. similar mutation & "crossover" methods)? — But maybe not. !

4/28/00 Bulg.

3 defficiencies of GA.
1) No "Hor. Ind."
2) Non-optimal Irrg. from errors
3) No general way to work Intr. probs: If a soft constraint "can be incorporated for it, this must be done externally.

So, it would seem that the "Vertical GA problem" is adequately under control — Say, using Koza's methods. That it could be probly much improved by SGA, but as is, it is more or less adequate.

Next: (42.34) That's real difficulty is in "Horizontal GA": Some suggestions ideas on how to work it.

1) ¶ 2.34 (Idea comes "borrowed" from ideas in w.o. different domains)
2) 42.13-.16 "Parsing" the problem def. as a source of cands,
3) 41.13-.18 Defines Phy HGA as a TM₂ w. known Goal
4) 40.25-.33 : Look at Koza's remarks (optimal, linear FL filter design) for setups for various problems
5) Note: I don't only want "Hy G cands", I want an "adequate set (complete) of hy G cands.

I may simply have small/fixed set of cands & input ops. But are part of every population of cands.

3 Defrencres of ordinary GA: { "What's wrong w. GA?" → Flow path to Rear top ....

1) Minimal Irrg from errors (solved by SGA) ← Secondary → 44.20
2) No "Horizontal induction" (TM₂) ← Main Defrency.
3) No General way to work Intr. probs: To solve one, Pgm User has to manually turn hard INTRON constraints int. the problem into "soft constraints": (usually This can be done in many ways, & there's no clear way to "mechanically" decide on optimal, good ones. Also, there are "tricky" ways to devise soft constraints, Reasons not to be so cleverly related to the Intr. problem, for devising "Hard constraints". → Sect 44.09 — Good!

Sortof important.
Some one can simply give it "any problem" & hope it figures out what to do.

Any Natural Way to Realize "Avoid Race" TSQ in my new GA formalism?

MOTIVATION for this analysis of the "Hor. GA" system:
If I could get it to work 0.4. Theoretically, then I could then just set up Koza's system to implement it. Even if the TM₂ (Horizontal) part, were very slow in doing proper Operator(s), I'd end up w. a fairly good, much better & (much faster) system than Koza's: Fairly close to a real TM.
Also, it would shed a lot of light on the TM₂ process in the more General MCT Machine.
To Get it to work TM₂ probs w.o. a proper TSQ for them could be a big drift, but it would simply take the system a very long time, & a lot of "Hand Tweaking".
Hor. it would be a way to set up sentences forming a simpler system as a "Wrapper" for the final TM".
→ Maybe not so difficult! The Electrones probly Pgm it worth "from scratch" in more quite diffs — so it might be able to do the TM₂ problem "from scratch".!

Dad "GATM" = How to test GA, TMS

(SN) Arms race (a) Mo always an interesting TSQ: Lion v.s. Gazelle.. On simple "bsq" the goal for lion: to run as fast as possl.
But with gazelle as "arms racer", the threshold for use of speed slowly ↑.
(b) Say TM's goal is to work as many probs in set $\{S_i\}$ as possl.
S_i could be selected by TM at random, or, TM may try to select the easiest ones first. An "opponent" might work by trying to select out S_i that are as hard as possl. for TM.

09:49.17 (SN) On Conversion of INV probs to OZ: In Math (INV probs) the mathematician will learn how to tell if a certain "state of the system" is closer to the goal (a "How much closer" (to compare other states) of the "present state of the System"). (I think many measures of this sort were included in "AM"'s/several heuristics).
As input data for learning such a matrix, one could have/problems Solns. by actual Humans. Developing such a matrix could be a main thing that distinguishes good Math "problem Solvers".
Now, the induction of such a Matrix is an "standard (G)" induction problem & is solvable by GA. Also see (21.02 -.15) also 24.01
(Dad) (16) The MCT machine (= MCTM) would be (.09 -.18) as a heuristic for solving INV probs.
.20 "Minimal Lang. from Errors": Likely Probly true of Hofz's system (but D'm not sure) I think it's true of Most GA work. If one has any idea "Why" a particular Coord was "bad", one might be able to form intuition how to guess at a much better (by G) coord. Any kind of Model of how G is a function of coord. could be useful here. (SGA is one kind of way)

Using these ideas, it looks like it would be possl. to make a (complete) full TM, using GA. It would probly not be as good (almost) as a MCT machine, however. — A "MCT Machine" may be a optimum.
A possl. PdP: To design a "complete" GA system, then design a "MCT machine" that solves same probs in same ways, but in addition, is able to do inductive things that GA can't do (also, it does INV problems easier (tho, may be not! — it may well be that .09 -.18 is about the best way to solve INV problems — The GPS was vector Gore" is silent).
.33 An advantage of this approach is that it seems to minimize the amount of TSQ design needed. This is because, the GA system is able to solve fairly diff problems.
Re GPS: It makes an inv problem into an "AND" node. So WON may be relevant

My attempts to avoid pesky, detailed TSQ design: (several Main Motives in a GA project (94.33). Designing TSQ's for a somewhat intelligent machine, one would seem to be much easier (and more interesting) than designing a TSQ for an "infant" Machine.

Superficially it would seem that TSQ's for a GA machine would be quite different from those for an MCT machine.

I was thinking of having TM (learn a lot of Math in some area, (it would also learn # theorems of/Math terms by induction). Then I wanted to start teaching it language. Then discussions of the Math it knows.

Could I do this in GATM??

More generally, can I do Any Prog in GA^TM that I could do in MCTM?

Perhaps I should explore a formal correspondence betw. GATM's solns. of problems & MCTM's solns. of those same problems. — with an eye to seeing to what extent an advanced MCTM would have same TSQ as a GATM.

If GATM can solve Hard problems as an infant, I should be able to design MCTM to do the same thing!

In general, Koza's work on GA could all be used to guide a much more efficient search. We obtain solns that are Minima (& are already properly parsed!

Re: My recommendation of "simple (non SGA) GA": I just have a "functional" lang. w. a static grammar. I assign pc's to various useful macros/solns. For a given problem domain, there is a set of input macros of the pc & a standard set of bp pc. functions common to all domains.

(SN) There is some duality here: "small" functions will always have larger pc's than "large" functions — Yet longer functions will be more w. by G. Well, that's o.k.; we have some duality in search for Inv. probs. We still look for solns. in $\frac{pc}{cc}$ order. ..... Hmm., in many cases, we know that small functions have very small G. — Perh it may be best to look "near" old conds of by G.

.32 — This is best analyzed by modeling G as a function of and dcon, (≈ SGA) 46.08

One factor Advantages of GATM over MCTM:

(1) GATM can use 11 models of Reality: MCTM looks for "best single Model" & does it really implement ALP correctly

(2) I don't (at present) have a very good system for doing DZ probs w. Lisp. The WON work would help. On the other hand, GA could be just one

4/30/00 Rule

$e^{.414} = 2$; $e^{.35} \approx .7$; $\frac{1}{e} = .368 \approx .37$

2.718	4	7.4	10	14.4	20	40	100	1000	1288	1.35	$\frac{20 \text{ yrs}}{2} = \frac{5.3}{1.5} = 3.53$
1	1.4	2	2.7	3.7	5.4	7.4	10	13.5	18	24	

2 doubles in 1.5 yrs.

So CPU speed doubles in ~ 2 yrs

$G^{2} = 7.39 \approx 7.4$

of the many OT's that I've used to solve OZ probs. —

Also, I may vastly improve both MCTM's GA by modeling Kazz's GA systems in as ½ search or ½ SGA system.

When "mature", GATM's response to simple Q's will be fast via mainly the "horizontal" unit — + stochastic operator.

So I've to up as complete Dcm of the systems as possible at present.

Then list various possible bugs, decisions, & suggestions on how to fix them.

08 : 45.32 On way to think about it; When we find some seeds of hyG, we look at their phenotypes (dcms) & modify those dcms. to form new seeds

What I want to do: 1) Write Review of this GATM, MCTM idea.

2) Finish Hutter

3) write reply to Wolfe. (will this important. Because Gray. Imp. is important)

4) Clean Desks for decision on GATM!

Q: GATM: Since TM has learned much in a certain "Math Domain": If ~~[illegible]~~ has learned (by induction) some defns, How do we get it to understand English sentences about the Domain it is familiar w.? In MCTM this seems easy!

We might give it simple Q's about the Domain ("in English"), that have simple answers. It would learn to return between the English sentences & its correct answers.

Metaphorically Any MCTM behaviour involving induction can be done by GATM, —

Since induction is an OZ problem.

.27 .23 → Actually GATM & MCTM are very similar:

.28 .24 In MCTM, we have a "grand" P.D. based on the problem dcms. This is a stiff P.d., hrs.,

.29 .25 We use Lsrch out individual problems for ~~[illegible]~~. final Solns to problems → (.29)

.30 .26 → Well, in theory, we do only .27-.28; but maybe in practice we may divide up the Search this way. (No: .29)

In GA^TM the "horizontal" part corresponds to the "Grand P.D. of .28.

The "Normal GA" part corresponds to the Lsrch of .29.

.29 (.26) Looking at MCTM again: we first get a rather broad P.d. from MCT, then we have to do an Lsrch on these P.D.'s. In GATM, we use MCT to get a broad P.D. on seeds —

.31 Then we use GA instead of Lsrch on these seeds,

I'm still not entirely clear on MTM v.s. GATM & in .29 ff!

.33 5/1/00 Later: .29-.31 seems quite right: The basic flaw of MCT was to realize that in Lsrch, it is best to def. srch over a UME that has been optimized for that problem — so the Pd search over is a cond'l P.D. — the "cond" being the problem dcm.

This Cond'l Pd in MCTM corresponds to the Horizontal GA in GATM! The correspondence is very close.

The analysis of OZ prob. soln. in MCT should be able to give a much better — understanding of Normal GA (& certainly SGA!)

4/30/00

Subj: Review of MCTM; GATM. (See HMC; 35.01 for a way to write a nice review)

15% ↓ ✗ 150 NM 1.18.00 / to GA. 1.19.00

200/1000 = 20%

4401 - 4480 is just about all.

1) Descr of New approach; What is MCTM & GATM are:

.02 (2) What is breakthrus in GATM are: (a) Horizontal GA. (b) Ability to work ENV probs as opposed to Several approaches (4480; i.e.)

3) Advantages of GATM (or mix of (MC/GA) TM): GATM can never have probs "about" It seems easier to write TSQ's for more "intelligent" TMs

4) I want to be sure Rest & have some idea Rest before me such optimum Pkg namely Approach.

05 (5) Value of Koza's work in guiding my development of MC GATM. Related to (9) (23) Also Note 46.23-40, 46.33-50 in particular

6) That I can improve Koza's work much by using "Clock" s/o SGA approach

7) The value of Money: Speed of machines ↑ at least x 2 in 2 yrs. So $1M is 1000 x $1000 machines. This amounts to 10 doublings or 20 yrs. (only 15 yrs if doubling is every 1½ yrs) With much < 1000 x a present $1K computer I should be able to play GM w. great success & get More Money for More computers. (or get money from investors). So it is likely that superconvergence/growth will occur because of ↑ of invested money rather than by TM participation in Moore's law. Thus

Investment of $1B amounts to 30 to 40 yrs of Moore's law.

10^6 x a present $1K computer should be more than enough for "super intelligence".

→ Perhaps 100 x ↑ would be enough to do ~~it~~ (good scientist!

Koza's pgm may be ~ human capacity & x100 w. my improved algms, should.

.19 Certainly get it at v.g. human level. → → 51.22

(8) What are main ideas Rest make this "new" approach look very promising?

(2) .02 is certainly impt. (5) (3) is impt. The Koza seemed to do v.g. w. apparently very hard probs, is encouraging. (9) is very impt.

.23 .06 (9) VERY IMPT: MCTM's GATM gave Distinct Views of (Gen TM Some problem) This often makes it much easier to solve a problem by switching betw. 2 views → Easy to switch: see 46.23 ff; 46.33 ff in particular!

* * TSQs * * 46.33 ff more a very sloppy correspondence betw. MCTM & GATM.

Anr: I still have to write a TSQ! The Linear DC design Koza used is of some interest. I could continue and get much faster results, perhaps design a much more complex DC's, i.e. useful commercial value!

The Advantage of a "New System" That the Jumps betw. problems can be larger than before, so it will be easier to write TSQs. It's more like writing a TSQ for a somewhat intelligent person.

.31 → So I could just go thru the sequence of Algorithms books: Get TM to be familiar w. many definitions, then start giving Q, A's in simple English, using concepts Rest TM knows about — knows the properties of.

.34 One test: to see if it could learn linear eqns, quadratic, robot (participant = ?) → 48.07

MAJOR SUBGOAL is to read English text books & be able to learn from them. Because of the relatively "intelligence" of GATM, This would enable rather rapid Iraq w. Relatively little "supervision". If we could speed up TM's education tremendously!

5/3/00 Bulp.

OSR2.PE

.01 · A great advantage of using "books" for training, is that they are "passive", that it would become possible to have TM be very brite, yet have no knowledge of "outside world".

The books give TM no access to F.B. from TM's actions in RW.

I'm not completely sold on .01, but it does bear further investigation.

There is the inevitable F.B. via a designer .... but this can be very slow. — We could manually stop modifying at a certain pt.

.07 : 47.34 **More on Lang Image** : From learning to understand Q's about Algebra in English, Ability to match up statements in TM's internal lang to closest approx. in English. It would be given many such pairs, & run relation back from.

Try to get TM to understand statements in Algebra, text books.

At first, no statements involving RW knowledge. — But after TM seems to be doing O.K. w. abstract algebra, try some simple word problems w "Dick and Jane": "Pretty" "Jane" will be "markers/subscripts" at first. Then we will gradually introduce more elements of RW. into the task & "word problems".

A possible "in parallel" problem for TM: to compress English text.

PROBABLY Best Approach: More or less write TSR, Ron & design TM to learn it.

Re: Long Long: Perhaps Main outstanding problem: How to learn nouns:

I unconsciously know how to do this, but I really haven't worked it out .... it's a continuation of recent work on "Z-14c" — in the spirit of [illegible] in my interpretation of Wolff.

Specifically Need to Read in notes on Z-14, Notes on Wolff: How I intended to do it: One way: pick some words — say a vocabulary "A". Consider BAG of words following each: Consider BAG of words following word "R" R & S are "in" the Rel "OR" Joining "sum" of those 2 BAGs will be useful for deciding what follows A or B.

.26: TM:2-14-1 1.40 BAGs are nice for adding together: They don't lose much info: Joining 2 sets, however, loses much info. In a Bag one still has "count" info after joining — so one can join many Bags. (3/6/00)

.28 Say one has Defined "A, B, C" to be "Type T" words.

.30 Then on coding & corpus, words coming after any member of T are "pooled", So the set of words following "Type T" form an indep Bernoullie sequences.

If this coding matches PC, then that "Type T" category is useful.

The alternative to the "Type T" Bayes coding is the "Type" (Bag "U") is the set of all words thus far) SO using this technique, we define the set ABC... ("Type T") and the assoc BAG that follows its members.

In older work (Z-14, Wolff) I think I used the term "Context" as a way to define Bags. The contexts didn't have to be contiguous strings; $\alpha A \beta \alpha$, β could be a context. Here — which is more general than the "Type T" of .28, B of Rel BAG and Bayes seq. idea in .30 is v. g.!

49.01

.01: 48.40 : Using 48.28 ff to more general contexts, we can find many useful BAGS. Then we look to see if certain contexts are ≈ certain other BAGS a/o if certain BAGS seem ≈ to some. Such discoveries can ↑ PC.

What we'd like to find, eventually are rules like [Bag1 could be Bag2 ≈ Bag3 where Bags can be Bags1 or Bags; Bag2 can be Bag1 etc. This is the beginning of a cf. Grammar: I can also do ≈ context sensitive Grammars the way (I think)

On BAGS! A useful operation on 2 bags is Pairwise component-wise sum. It's much better than the sum of 2 sets, because in BAGS, the case counts info is preserved & used. The product of 2 Bags might be of interest (products of cooccurs. case counts: We may want to know wrt. one of the 2 component Bags. (possibly wrt. both — but then the Bag becomes a P.D.

Subtraction might be of interest: [redacted] case counts after subtraction: 2 possib. ways: ⊗ Negative values are OK. ⊕ Neg values → ∅. It may be v. bad to use for neg. case counts.

SN IMPT. One recent idea was that I crash anyway so be courageous to crash. Putting trials in [strict PC / CC] would be better. MCT can probly enables us to estimate cc [redacted] PC of any card.

Many kinds of P.C.'s, which may PC were ≈ in many kinds of P.C.'s

.23 RE: LANGUAGE Lrng: The idea of starting w. an Algebra TSG was that I wanted TM to have some domain where it knew things. This means it has some grounded internal rep. "Learning English" then becomes a problem of finding correspondences betw. internal reps & English. I think this may be a far

.27 easier problem than inducing a PSG from a corpus!

.28 ≈ Inducing PSG from Corpus can be an "impt. kind of problem: In "Reverse Engrg" of some Artifact or Structure: But it is very diffl: Like inducing [redacted] "linear B" w.o. a "Rosetta Stone". All kinds of "side info" are v. useful —

31 any thing that is available. Perhaps inducing Good Models in Other Sciences or Psychology or Geology? (.37)

But I think that TM can get a fairly good understanding of English — Equivalent (at least) to understanding of Eng Grammar W.O. having to do [redacted] discovery of CFG's from Corpus. Also the understanding of English can be got by studying Algebra only.

.35 In reducing / sub-operators in a set of cards, the cards are common useful already sorted. → 51.08 parsing after math.

.37 (31) ≈ So: T. Q of [Where one uses this kind of induction used in inducing PSG from corpus also]:

.29 - 31 are some vague suggestions areas where this might be useful: Places are areas where don't have good models for the phenomena. Possibly in SEM's method of computing C to coordinate

.01 — The More Generally (in this SGA Applicn), any P.D. is a stoch lang. & v.v. —
While I was thinking of (PSG)-style stoch Grammars when I invented this SGA trick, it will work for any P.D. → stoch Grammar. Hrs, using a (CFG PSG) style stoch Grammar would give a different style of induction from most P.D.'s — So in this sense, PSG discovery could be a very useful trick.

[Note, hrs, that a P.D. on strings (i.e. cond dcms) will tend to be something like a (PSG CFG).

Def note] PSG (Phrase Structure Grammar) is more General than CFG and includes context sensitive Grammars. Much of My "My Level" discn. of Grammar Discovery is Relevant as well to context sensitive Grammars.
— So perhaps I should use "PSG" in my discussion instead of "CFG"

Because T. SGA technique for fitting G functions to cond dcms. would seem to be very impt., the discovery of stoch PSG's would seem to be very important. However, an impt. Q is whether I have to get the system to discover "BAGS" of words, either from unparsed corpora or whether the Data TM uses will always be preparsed.

So this is a Big Q! How far can I go in TM, w.o. having to discover regularities of the sort that execute in unparsed, English text? ?

For English, I think I could discover words in an "un parsed" corpus. — But the discovery of a PSG via "BAGS" needs to be worked out.

In "Hor GA": (Or even perhaps MCTM): $\forall \frac{E}{\Phi}$ is, How to represent the p.d. of the possl. soln. string(s) as a funct of the prob dcm. strings. In Hor GA the representation will be (in conventional GA) via a bunch of hy G cands, that can interact to create better P.D.s, via "Feedback" (≡ G). In MCTM: a possl. form: A stochastic pgm in a functional lang. w. various probys for choices. If the d.o.f. is relatively "sharp" it will consist of a few "large" functions & ways to modify those functions. ("low PC") Essentially MCTM will be simulating the P.D. obtained by Hor. GA.

SN A possl. way to get true expected G of a cand: try cands in ↓cost order (or ≈ ↓cost order ... any fixed linear order will do). Whenever a cand is evaluated for G, update Ave G for each cand. Whenever picking Champ as max G (considering all of G), use Susy Analysis to get "SOP" value of G. The ↓cost ordering gives (maybe) a parameter family of cands. (↓param ⇒ ↓cost) ⇒ Go ANN!

When using "SRICE" or any other very expensive eval'n scheme for G, (1) TM should try to find a less expensive way to reject cands of very low G, rapidly (Quick & dirty). Finding such a "Quick & dirty" technique could be a IS problem for another GA (analogous to the ARMS RACE in which we have 11 GA's)

Note: Last few pages of work on 2145: I found one of the approaches which will work unless (1) $x \times \text{wgt} > 1$ $x \equiv \frac{\Psi}{\phi}$ Look at this again & the Empress if — Some such which appears have to be discarded: Maximum Scoring Subset Now (5/5/00)

.01: Since "Hard GA" is a constant ("fixed") problem for TM, we should try to find special techniques to help solve it. Would "Crumby components" be helpful? — If a "Crumby Component" finds a truely good Alpm for mapping problem derivs to hy G reads — we don't have to know how it produced this result! — It's like a Monte Carlo search to vary the uncertainties in the Crumby Components.

What I want to do is outline the entire TM project & telling how each part is to be done: Then list the parts that need most work.

.08: (48.35R) Ent. Algebraic TSG: Do a lot of complex English about Algebra only, before introducing R.W. concs (Wheeler "story problems"). E.g. Is x always greater than 3? Why did you subtract 3 from both sides of the eqn? Should you do normalization or integration first (Idea of temporal ordering)

We write next [redacted] do Geometry before going to R.W. This should help in Understanding R.W. (Bob may be not enough to correct the discussion!).

.16 → One (& isolatable) part of the TM project will simply consist of feeding a sequence of problems into TM. I can summon a large sequence for "evaluate" work, or have it work on "TM₂" whenever it runs out of problems that I give it. I want the "state of play" of TM to boom disc after each problem, say: So I can take the state & put it into a larger machine at any time, or add search capacity to the machine at any time. → 52.01

.23; 47.19) Koza used ~ 10^{15} ops per problem soln. ~ 10^7 ops per sec were used for the (.25") "Spice" evaln. So it took $\frac{22}{10^{15}}$ secs to evaluate G, that is a speed of .25M = 250K

21 days to solve one prob w. a 300 MHz machine. $\frac{21 \times 24 \times 3600}{250\text{K}} = 7.35$ seconds

to solve an equally difficult prob, but w. the evaluate "G" a chess now used. (Much as 7.35 sec w. a much better GA / HarGA system & MCTM)

So it looks like Koza's TM should be able to solve fairly difficult induction problems rapidly enough to have us busy w. a "manual" TSG.

However, it is not clear how difficult/the problem of choosing initial conditions & benchmarks set to use. Initial Conditions may be relatively simple: Seems to be proportional (almost entirely) on no. of inputs, outputs.

Originally, I had ideas of trying to get TM to be of where it could "Understand" English well enough to read conventional text books, & from this, to range more general literature, e.g. SM stuff. However, it may be more of an "ultimate Goal" to understand Physics/Chemistry/Biology & make great Theoretical (& possibly Experimental) breakthroughs Rearm. Quantum computing & all the other non-standard computing techniques "on the horizon".

Also, perhaps I want TM to do experiments w. DC devices, in R.W. (GA in RW: Using FPGA's, etc.

.01.5.1.23 So. There are 2 (≈ distinct) parts of TM design: ① TSG [redacted] descent & running (5 I 16-23)
② Improvement of TM by (from Theoretical) analysis & by Hardware upgrading/modern.
(Also by just adding more as fast memory.)

→ So: What needs to be done: I should write a more detailed dcm of the GA TM
Weakest
I. I may diffys & weakest pts. That need work.
Perhaps start ruff "ESQ" from Alg. Book: Get idea of what kind of lang. is needed. (Tho w. Itor GA, afterwards GA's should at best be very critical HGA should effectively modify & levy forward optimality.)

T. unaccountably I'm afraid to quit working on this for a few days! That I will forget just why I was so excited about the creation of the 2 pt comm — I will forget the coins. (& approaches to coins) But I had in mind for various diffys.

So spend a day or 2 on Hutter, Then get back to Ruy for a few days; Then reply to Wolff, (takes a few days) Then get back to Ruy's full time.

While I'm waiting on Hutter reply I try to maintain Contractor. Ruy present work by DC & actually work on it.

One of Main Roys I must do: Read up on how Lenat did Strings. (& ADR's = "Automatically Defined Functions": Get/review Koza on GA may have something in Ruy: Lanzhafatal: pp 284 — 288: Also Automatically defined Macros: p295 - 299 — Also the rest of that chapter is of interest. See esp. p302 & p 299. Last of chapter is into direction of HGA) → p107 ff with this chapter's on GP, which is mainly Koza's stuff. (NB: 333 is on Machine code GA (Ray. says no self-timers as regress) (AIM GP rule)
Also Nobel Good Q's on p338

On Lang "English": One motivation for Lang Algebra first, was that I could learn French to understand English statements about Algebra. T. idea is that it's easy to learn a "second lang." if one already has a "first lang." Here, one always has a "first lang" — i.e. one's "internal lang." T. Q then, is whether to learn internal lang → Alg. lang → English about Alg.; or easier, faster direct to learn internal lang → English about Algebra.

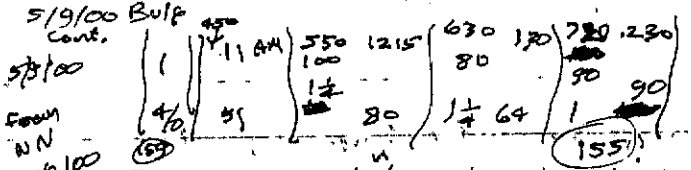
Search doesn't solve the problem. Must all solve the problem.
Search doesn't solve the problem. In all search or TSP or 2T search, we end up doing all of them. — a big waste of cc. If TM is able to estimate their cc's, it can do them in $\frac{pc}{cc}$ order, which is much more efficient.
— What does GA do in an analogous situation?

Also Re: One size Lang. In ALP (or MCT) we have to

deal w. Pick in a special way. Is GA able to Deal w. OSL? If so, does it do so in an efficient manner?

ON APL: Random configs of ACP functions tend to be "interesting": So in Pick sensible functions could be ideal for certain areas of Math. Realizing APL in a fast functional way (say a fast forth) could be worth while for GA Mc TM - ⊃ Or: use any complete set of basic instructions à "touch" TM (by induction), i.e. loops used in APL.

5/9/00 Bulp cont.



5/8/00

Room NN 5/9/00

(69)

pc's & cc's

(3) The Method of Search being tried if I have a bunch of cards w. same pc & cc } IMPT !!

(... I play on all spins." Doing from in. strict pc cc a known order would only span total cc of that set. For sets of cards that need not solve (which is (most all) we would have nothing!

(4) So / Similar Search may be tried also?)

(9) If k is (cost of spin, either of (2 ways of doing (search, 1st how to look at all cards w. Lc ch: If search is, have, factor by factor usually > 2, stop if/or no. of cards in (i strata (?) } check this.

So in (search one can make many for time. But there may a way to use hard disc to swap in and out. Only if n >> 2 would it be worth while to do this. [Churchman Peirce "Recherche n" idea!]

.16 (5) In Her GA, TM_2 will occasionally send a card to (i main GA pool. In (i pool, it will be treated as a card, but also be retained for "x ants" In current test

In MCTM is (i an analogous process?

Well, (i isn't exactly stated! In Her GA, TM_2 starts out by continually and independently (Sep/Complete) sets of cards for (i Normal GA to start off on. Then TM_2 no longer (contributes to (i problem.

In MCTM, (i analog of TM_2 runs tracks a / find Umc corresponding to (i "hyst problem." (i sign on this find umc is then all (it's done to do (i problem. — TM_2 continues to work on (i problem unless Non-Essential Corpse

Now this is a MCTM (more) designed to be very much like GATM- [redacted] As such it is sometimes "EC". A less "EC" version would at all times try to find optimum solns for (i entire Her corpus plus (i new problem.

(6) In Hoza set of "filters", where they all perform LC filters rather than I(e resistors) Looks like it: Perceptual control theory" used to create "Pier Optimal Ecology" Cybernetic: no cross.: Also the "ladder" filters may have been up of hyly constrained structures!

(7) So what Hoza has is a good (way for doing a) linear Lc filters & (curved LC R's filters) & b) linear Cc's.

Doing are in form of TREES ⇐ EMPT. Maybe all it is is a set of 2 & 3 terminal "Reactors" that can be extended to any n at by replacing (i "functions" at any 2 or 3 terminal node. (I'm not sure about 3 terminal extensions)

5/9/00 Bug

(8) Re ordering - coords by pc: I wonder best way to do it can tolerate fairly large (Rec < some n, say) mis-ordering.

From NN 5/9/00

One way: approximate all pc's by powers of 2. (Then make all expected's integers). Just how does it help? Well, if I have a set of pc's that sum to ≤ 1, then any code is a prefix code that can uniquely be decomposed into its codewords. So we list all binary strings in order of length. Each such string creates from unique uniquely decodes as a seq. of resulting pc's.

So Q is: How to do coding/decoding cheaply?

Say we have this finite set of pc's $\{a_i\}$. We normalize them so $\sum a_i = 1$. Then we assign pc's that are powers of 2 to Recon: $= \{b_i\}$ We will get $\sum b_i \neq 1$ usually, but we may be able to pick so that $\sum b_i = 1$ by handing "left" arguments a bit. So say $\sum B_i = 1$. (How do we do the mapping?) Look at Huffman codes.

→ 55:06

... Look at my report "Optimal Search": It has a way of generating a large class of prefix codes.

(9) Actually Lsearch does exactly correspond to GA; Lsearch corresponds to 1 round of GA. If Lsearch is on trees that represent functions, then the first rounds of Lsearch once looks for common subtrees in by G cards.

At first, we will only look for subtrees of size 2 or 3: Actually, the upper limit on size governs how much time it will take. We want it to take (maybe) the time of 1 Lsearch "Round". → (11)

(10) What state have "(No recipe for" NN. FLA ... ? Maybe make to 2 country?

(11:9) In Lsearch: do looking for subtrees, store found (during) search. If not enough subtrees have been found, just continue to next. In fact, if, on finding a subtree, one is able to discard it into the search strategy one needs never start a new round".

(12) On OSL: If TM can't do OSL, then will merely show its learning doesn't help: Easy. This be very impt.?

Also: does normal GA do OSL?

(13) Maybe transplant I way easier than this method: Recirculating pump. Also Electric bike pumps are available (12 volts). Maybe make pressure detector that turns on pump if pressure is < 40. Go to Car Junk yard! (or Garbage or Yard Sale). Also get clothes pump at yard sale/garbage.

(14) At any point, the useful Memory of TM consists of its set of determinants & their pc's. This should be transplantable to apply other "Technology" for doing TM's.

55

A V. G. fast Soln. to problem: ordering of Conds (.01)

.01! 55.15: Huffman Coding Korrect: We know k pr's [pc_i] we normalized so Σ=1, Rean we d.t in Huffman Codes (There are very pretty, very fast ways to do this — Maybe see Knuth book on "Compression"). So every binary seq. maps in to a seq. of Huffman words. Each o our decisions in t. binary seq. either stays in t. same possl. set of words or reduces t. no. of possl. words. Finally, t. no. of words is 1 & we stop: Then start at t. root of t. tree again.

So, for T ← 2T Lgrch, we search Rem all binary strings of length ≤ log_2 T. This will give us functions w. many inputs & 1 output. (Actually, we'll generate (not-functions, Rem we can make use a trace in a unique way.)

We usually need touch of a small fcn on a small no. of inputs only. I might work this out in Saarbrücken, in some detail. I obtain (semi-empirically) a formula for t. no. of functions we'll in certain constraints.

To create Functions w. 1 output & k inputs:

We compose t. final function of a pool of sub-functions, each w. its own no. of inputs (& 1 output)

Depth 1 functions: Choose a function from t. pool. Its outputs t. final output. Its r inputs are all k^r possl. input combinations.

Depth 0 functs: k possl. connections of t. input to output.

Consider functs w. k inputs & 1 outputs:

For zero depth, each of 1 outputs has k possl. inputs. So k^k combos

For depth 1, each output can be an input, or t. output of any of t. "pool" functions. Rem all of t. inputs to t. sub-functs may be any of t. k possl. inputs.

SN or OSL (one-shot learning): I think GA does it as one of its main methods! Say α is a hy & cond. Then ∈(α) is tried — which is a small modification of α. If ∈(α) wins "by & strong" Prauf(α, ∈(α)) → something. Both (in some sense) SSZ=2, for α. (Also, if ∈ has been used before, then SSZ=2 at least for ∈)

CAN OSL be incorporated into SGA? IProb: this is a serious Deficiency

(5.11.00) RANDOM Thots: (1) That t. pc of functs of as little as 3 components touching, is rather small, & drops rapidly w. no. of components. T. moral is, we can't go very far w.o. feedback!

(2) koza averaged 40M trials per soln.: It may be that Koz's problems were much simpler than L Prob, but that he used very good methods to narrow down t. no of "functions" used in his mutations/crossovers, i.e. he has a very good [ways for devising filters (less restrictive/less filters) & I know L.C.'s.

(3) It may well be, that L, too, can learn to find a small set of functions to search for very important problems, as well.

5/14/00 Bulg.

Typically in GA, we will have (in "chromosome") a bunch of genes of type G: [redacted]
We then generate new ones that are "near" those by genes.

5/12/00 #5 copyright page

When one "crosses over" 2 subtrees of different ones: Is this a kind of DSL? 3rd "yesterday" days.

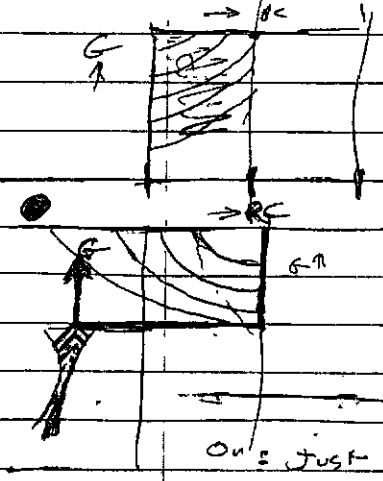
5/12/00 K_G: D.F. of G of ones obtained by crossover from ones of $G \geq G_0$: Say we are mainly only interested in fractions of ones w. $w > w_0$. — but we do want width of Φ if it we want to pool "good" ones of ones w. $G < G_0$.

(4 SN) Note that SSZ's may be << those of KOZA! ("Fewer, Better, Bands")

[redacted] DOG

If there are k branches on a cond, then $\leq k^K$ poss'l. offsprings of that type cond! For n^n if all branchings are distinct.

See Think of the (Cond, G) (Gof parents, pc of offspring) → d.f. of G of offspring,



so expected # of offsprings is ordered by G à by pc, but locally or even by G & pc is unclear.
As we go by G ones, population ↓,
So we must surely want to have kinds of at least one by G-[redacted] parent
Each offspring of G, will be [redacted] w. ones of ≥ G only
Perhaps try to get a curve fit to Φ_G as a fcn of G, pc. of one(s). At any rate, there are ways to make good use of available examples info

Or: just try for (Cond; G) → pc

Or Is that NL ought rules for small ΔG? In fact, how does brain decide on a rule branch & G a reproduction rate? — This may be one of his "parameters based on experience". — Presumably "evol. strategies" have some kinds of problems".

E.S.

Same bib, p 132

.29 Bremermann et al pp 126–133 discuss G [redacted] criteria & selection based on it.
for "Evol. Strategies" My impression (I may be wrong): Population: used to obtain
.35 w/ each larger "next Generation" — which is reduced to size of first generation by
Selecting "best" individuals. If I were to do this, a single param, would
be ratio of pop size to offspring no. (Better selection). From a population of size N I
would make a bag of "root parts" & a bag of "branch parts" I then generate
.36 the next N individuals of highest pc. (This is easy to do).
From these N, I select the N of highest G for next Generation. → 58, 07

Holland used $p_i = f_i / \Sigma f_i$ f_i = fitness, so res on ↑ function of fitness.
If all f_i are "100 ± 1", then makes all $p_i \approx$.

5/12/00 Bulb.

tan⁻¹(ax)

Somewhere in Ry book (Bezhaeva) / P127 Ray mentioned .. standardized set. beta dist. was zero. This would seem not to help w. Holland's formula. — Hm, it was peculiar, and so Ray probably found a way to get it to work.

If f_0 is normalizable then SGA is applicable. (i.e. random where G can be modeled by a piece of a stock grammar) → Hm, see (120) always

"Normalized fitness" (p127) is between 0 & 1.

.07 : 57.36 : Instead of a fixed K (57.55) one could generate offspring until the top N of them satisfied a certain criterion. — Maybe a certain fraction of d's > best G of previous generation? Or maybe average G in this generation is a certain amt. > the mean G of previous generation.

— We have a trade-off here, between executing more cands from present Generation vs. starting a new Generation which gives more progress per cc?

Another way: instead of starting a new generation, each time we generate a cand w. G > the min G of the present generation, we add it to present generation & delete the bottom G cand. Whether we just continue doing this until we win / Hm. Bookkeeping would be very diff't! The Bayes would change & one would have to keep track of which cands had been generated already so as to not regenerate them.

.20 : .05 Rec Norman (at in SGA) It will always be possible to find a (always actually P=1 is true). In fact they always want a D.P. over the cands.

.24 Say we have a bunch of populations: Bunches Each Bunch has G = G_i.

.25 Assume that we are somehow able to design a function $P(G_i) = p_i$, that associates a "probability" w. each G_i — so highly fit (G_{20}) values have high p_i. For each Bunch, we have a root bag R_i & a Branch Bag B_i. Say the root/branch bags are normed so the sum of the "case counts" in each B_i = 1.

Then, to create a new generation, we create cands by random results of R_i's by B_j's — and we also note by $p_i \cdot p_j$ the pc'tage of the Bunches that R_i & B_j came from.

We create the new generation cands in PC order: highest first, we stop creation & use the top G section of the created cands,

.35 → "at some point" This "at some point" determination may be similar to the choice of K in ES (cf. 57.29 → 57.36 → 59.0)

Hm, in .24 we also have to formalize what the choice (involving a) largest r (from) of the mapping (of .25) from G_i to p_i.

.01 ; 58.35 : t. "at which point" at pc_0 to cutoff creating new cands. — could be answered by an "efficiency measure". T. cc required to generate a certain no. of hy G cands. — On t. average — this pc_0 cutoff choice will determine how much cc we need per hy G cand.

MAIN Problems in (QA TM) (Approaches proximate) (MC) (increase)

.07 1) Getting a good way to (order) cands from an population, so one doesn't do random selection, but puts them in order of desirability (expected hy G).

≈ E.S. ⇆ 7 29 - .35 à 58, 24 - .35 ↑ re 2 attempts at this.

2) Demy OZ problems via methods for MCT needs to be developed into a practical system (ref Lossl.!). — Mr should help solve ①

3) I'd like to understand exactly how Koza's (Angeline: Potlach) do "Automatically defined functions": would looking for hy freq. subtrees be a better way to do the same thing?

4) I need to understand Koza's (use for deriving networks (IC editors à (linear EC's) more exactly. It is likely that many problems can be usefully expressed as "finding an appropriate network" (e.g. Trav Slsm. Prob. . .)

So I'd like to be able to focus. this technique

5) If I can use these or related methods to solve "hard problems" my hope is that TSO writing will be easy.

6) TM2 Has a Pit induction problem. The problem areas are in a kind of common for mats (i.e. Inv or OZ pairs). I'd like a [illegible] bunch of Languages to see related Phases problem areas as input to an operator that gives hy G cands as outputs, (Horizontal GA).

7) 60.20 : Unclear on how my dormancy commonly occurs for meaningful 60.20 Use of [illegible]!

In Book By Kinnear (ed) p47 ff "Evolv. of Evolvability": Looks like a better understanding of "what goes on" in GA, than Mosf! (Mitch Potter Penn/Hollands "Schema" from Kinnearcod): p 35 - 39 - Koza : I think I understand ADF:

A cand. has its genom 2 parts ! ADF code à Main pgm code.

ADF codes can call each other : Main can call ADF codes.

.34 ADF codes have dummy args : Main has real (no-dummy) args.

In evolution, we reproduce, mutate, crossover, as [illegible] in Non-ADF GA; but we have restriction, that crossovers can't go [illegible] "ADF" parts & Main "Main" parts because of .34.

Because of this xover restriction, ADF because can have distinct "primitive sets" from Main Branches. This appears to be true in Koza's design of

p.1 Linear IC's: They say ADF has more general fundamental theorem: In "Main" ε must be at least on pin (or F.3) of Gn6, B" or B⁻. In ADF, there is no such restriction.

In Banz... pp156-7 discuss crossover: "75% of offspring die." I think because useful subtrees are disrupted. In organic evolution, almost all crossovers survive. — But in Koza kind of organic evolution, it may be that we don't have enough "Evolution" i.e. only very simple probs are solved: Also, we prob. don't understand how any crossover works when it's solving more serious problems!

Anyway in my version of ADF: (Say I use hy frequency of subtrees in hy G trees): for these sub-trees, I make definition, then the subroutine definition is substituted for the sub tree. — So that one can't break up the subtree by crossover!

Ang-Pollack's "Modular Acquisition" sounds much like what I was thinking about: Their chapter in Kinnear pp 119-141 discusses why it wasn't as good as ADF & How to fix it up.

Banz: p143 discusses Crossover v.s. no crossover: Apparently there is empirical stuff in which No-crossover is finer! — But read this chapter on Rep!

.26 Re: My own idea of "common subtrees" in hy G trees: If the hy G trees were constructed by crossover from other hy G trees, there will be many larger (& small) subtrees that they all have in common. This will arise from the hy G trees were generated (i.e. by crossover — & even if partly by mutation).

.26 If I generated the trees by Lisp, perhaps this objection to common sub-trees would not be relevant. — Check Rick!

Module = ADF (Discussing Pollack)

Be sure I understand Ang Pollack "Module acquisition",

— Vague: My impression so far: That main observable difference in effect is that ADF's are called more often than main "Modules", If so, this would be because Modules are not (equivalently) assigned proper pc's: I should be able to fix this. (Could this difference be equivalent to an "OSI" effect?) Actually assigning proper pc's is not trivial problem. That hoza should do it closer to correct, maybe an accident (or maybe an empirical param that hoza Dewd! ☺). Anyway, I should be able to get v.g. "with respect to" user on ADF's & INT. A.P formulation, it does look like OSL is relevant. Not easy to implement (except in this Monte Carlo Approach — which is not really the best way to do Lisp!).

.01 Perhaps Kozza's ADF formalism is a good way to decrib. OSL in normal ALife!
As for wts. assigned by ADF & ASP: ADF may be clear from Kozza's paper, But not here; ASP may not ... but look at ASP literature first I have.

→ Also, look at Kinnear's "soln" to the problem.

.06 Re: the problem of (DC 59, 07) Kozza has a kind of "MCarlo soln. — so I should be able to derive pc's for his choices — then, instead of his MCarlo, use the much more efficient "Lsoln".

The bottleneck in .06 had been my inability to understand Kozza's GA process well enough to really get it right. But using Kozza's pc's & using Lsoln instead of MCarlo, may be an adequate improvement.

.12 '00.20: There is the Q of pc's of objects "within" vs. "among" the same (and) vs. pc's of objects due to commonness in the ensemble of cands of (same) G.

Say we have a corpus of cands w. $G = G_0 \pm \Delta G$. We can hope to fix corpus into a stock long via Root & Branch Bagg (Prism equiv. to domain/crossover).
This new stock (any) will have a certain μ_{G_0} & σ_{G_0}.

We do h some attempts w. another corpus of $G = G_0 - 2\Delta G$ # $\pm \Delta G$; we obtain a # μ_{G_1} & σ_{G_1}; we expect $\mu_{G_1} < \mu_{G_0}$ (maybe by about ΔG). The σ_{G_0} & σ_{G_1} probly would'n be much different, hvr.

We are mainly interested in obtaining new cands of hy G. From the G_0 corpus we get a certain fraction of cands of $G > G_0 + \Delta G$.

From the $G_0 - 2\Delta G$ corpus we get fewer w. $G > G_0 + \Delta G$. ✱

As we drop the mean of our corpus, ($\pm \Delta G$) we expect that the fraction of cands w. $G > G_0 + \Delta G$ will continue to fr.

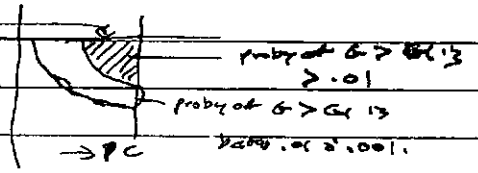
At a certain point, considering corpi of smaller G will become counter-productive: i.e. we would get more hy G cands per cc, by going to the next generation. So we have to decide when to do that.

[Perhaps] starting a new generation has a certain min, fixed overhead.

When we go to new generation: 2 factors: ① The initial rate of no of hy G cands per cc ② How rapidly that rate ↓ — which governs the size of the new set of hy G cands.

Say we have a standard 20 mid-R for a sub-corpus, but different G_0's.
For each G_0 & each pc, of cands from that corpus, we have a certain probability that the G of that cand will be > some fixed G_1.
So, if we set threshold for 1% of cands should have $G > G_1$.
Then, for each G_0, there will be a min pc of cand, for which the probability of $G > G_1$ will be > .01.

.38 So, if we want > 1% of cands to
.39 have $G > G_1$ we restrict our breeds to this region.



.01 Glitch: There is an approximate way to associate P_c values w. G_0 values, so Reading the Graph of $(61.35 - .39R)$ we can multiply the pc's of a cand by its P_{c_0} & obtain pc's of components as if they were all of the same G_0. This is a way to "pool" data from all the diff G_0 values in the pc's of the components of low G_0. Just how the $G \to pc$ mapping is computed is not (yet) clear. → (.26)

Anyway, using the mapping of .01 - .06, we can do steady state GA. Every time a new hyp G cand comes on, we update both the wts of components, & we "trim off the top" # cands of highest expected G. We do this by computing the hyp pc components of the new hyp G cands, w. (sometimes very low pc) components of all other cands.

→ This is a "semi-elitism" & may have some unfortunate effects. It does remove incorrect versions very low pc cands.

What is not clear is what is the threshold for pc's of "cands to be considered & evaluated". Well, I am trying to test hi cands in "pc order" — so at each point in process, I will have a "highest pc hitherto value". When I generate a new hyp G cand, it may or may not give any components to P(new) are any new cands w. pc ≥ current threshold; in which case, the threshold might ↑, so I can continue to trim.

.20 On the other hand, as the trim continues my mean G_0 of the cands of interest will ↑ & there is the danger that I keep the threshold constant & may end up w. an infinite no of cands to test (?). — Maybe not! There may always be a finite (& relatively small) no. of hyp pc cands, because pc is always bounded above (as G_0 is a strictly monotonic scalar function — it is not bounded below on the log scale). This scale is very "sparse" on the hyp end! BUT SEE (.34)!

.25 .26 (.06) Anyway: Every once in a while, when the mean G has ↑ a lot, (as a result of progress in the problem) one will have to "recalibrate" the $G_0 \to pc_0$ transformation of .01 - .06.

A quick way to do $G_0 \to pc_0$: Take the top 1000 or 10000 (or whatever) of current cands or highest G. Make estimated pc or G in that range (mean $P_{c_i} = \frac{e_i}{\sum e_i}$. (It is handy to normalize (mean because we do multiply pc's to order cands.)

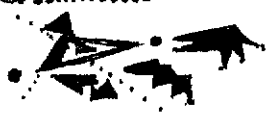
(It may be handy to insert a "renormalization" factor in pc to obtain its properties of pc's required in (.01 - .06) ← Maybe $pc = a G_0$?

.33 .34 (.35) ↑. Argt. at .20 - .25 is wrong! It assumes a simple search for a ENV problem! In fact, we have a continuum of "ENV" problems, for higher & lower G. As we ↑ G, the matrix changes so it measures distance from the highest G cands found up to that point. (The ray may be exactly "right"!)

One way to deal w. the difficulty of .34: The P_{c_0} values are continuously (or periodically) updated w.r.t. the top G valued cands found thus far.

5/14/00 Bulg

Lucy's Sister recently suicided.....



'01 : — So we are [redacted] mainly constructing our "metric" wrt. "distance" from the target cands.

Remember: I'm solving 02 probs as a superimposed DNV probs, using Ltsych. [redacted]....

(No!) SN Perhaps for all G functions there is a "natural" pc, having to do w. how many of each G value there are. [redacted] However, such a pc would not be monotonic in G (usually). Also, for many nonzero values (low G) would [redacted] have ever Carding like [redacted]..... Ann Robbay Divorce Lawyer.

.09 → The assignments of pc's as in G2.0 +- .06, .26 - .33 May be recognized as a kind of SGA. We do have to skim off the hy pc elements of the stock grammar in a very exact way. Also Nobo Puelin SGA, we would sometimes concentrate cands of low expected G also simply low G, for "information" purposes". Usually, in the case of SGA, this would be done in a way to help find hy G cands. In the GA method I'm considering now, generation of some low G cands could be of some value by introducing novel Roots & Branches that would not be otherwise recovered/morphed.

To some extent this is done automatically, when e.g. one selects a very hy pc (Branch Root) & "marries it w. a low pc (Root Branch) → (65.05)

.21

.30 One could have an adjustable amount of low pc cands in the poly. way. A "pure Elite" method would set a Threshold, T, and [redacted] generate all cands w. pc > T. (This is easy to do, given a Root Bag & a Branch Bag) If I want "lower pc cands" w. parameter α, I use Threshold pc > T/α, and I select (randomly) 1/α th of the cands w. pc > T/α. (Only if α > 2 or 3, say. Just select random cands (w. replacement or w.o. replacement if it's not too expensive") until we have as many as we want.

.25 So, we do a run w. α = 1 (= Pure Elite). If result isn't good enough — or we get the best local max, try α = 2 or 4 or whatever & do a manual rerun.

.28 So, one Q is, how much does .25 differ from E.S.? (Evolving Strategies: 57.29) (Also: 31 → 66)

One Advantage of the Normal Monte Carlo GP system is that while the d.p. of pc's of cands is very wrong, it is oriented toward [redacted] (i.e. many low pc cands are tried). — So if it does converge, it will have considered many "far out" candidate ideas

.33 It is theoretically Minimally "Elitist"

.35 On the $G_0 \to pc_0$ mapping. We want it to have this property: $c_0 = f(pc_0)$ f: function F

Say G_0^1 is the mean G of one generation sort of cands : $pc^1 = f(G^1)$

G^2 " " " " another " " " $pc^2 = f(G^2)$

We want the scaled cands generated by G^1 that have (their raw pcs) × pc^1 = α to have

.39 ½ the same mean G^2 (i.e. roughly same σ_G) as

.01 The set of conds generated by G_2 that have (Pairs raw pc's) · $pc^2 = \alpha$

.02 So we want t. $f(\cdot)$ be s.t. such that α characterizes Pairs μ_G (& hopefully σ_G) approximately of the offsprings of a set of conds. So if we take the pc of a cond & multiply it by $R(G)$ of that cond we will get a number that will tell us t. expected μ_G & σ_G of t. offspring.

The idea for 63.35 seems unclear — I want to be sure that if it is unambiguously

.07 defined (63.35 − 64.01) maybe best way to do it. → (65.51 for a small idea for a Gore — Goons good))

63.35 − 64.07 needs to be fleshed out in more detail: also, just how one goes about approximating $f(\cdot)$.

Another point was the "equivalence criterion" of 63.39, 64.02: Pairs can be any criterion of equivalence; I may want it to be "t. expected fraction of offsprings w. $G > G_1$" for some fixed G_1. (or an only equiv. criterion "C" ... here)

→ 65.09 announced later

Next, I want to look at ADF's & A.P's formulation & "common subtrees". Check on my idea of why A.P's marked regions much better than just using it as all. — i.e. too low pc's & t. modules" Also √. relationship of this to OSL

5/15/00: W.o. ADF's: t. (only?) main inductive abstractions are Roots & branches. Presumably ADF manages smaller subtrees useful as well.

.20 The function $R(\cdot)$ that maps G to pc is important in making it an SGA system (26) Improvement. → Probably it will be necessary to periodically update t. params that define $f(\cdot)$. As it is, GP is extremely slow & extremely discouraged — so, while slow, it should get us. going (if ever!).

All BRANCH functions have only terminals (leaves) as inputs, so may also not vary general functions.

.26 (.20) Hrs t. output of this "SGA" stoch grammar → quite unclear! — It looks like nothing more than a stoch Grammar, w. no G output! → NO! It does have G tables for each saved cond. T. conds of (low h_y) pc tend to have (low h_y) &

Furthermore, in spirit of SGA, it is easy to generate h_y pc conds.

It would seem that how would have to insert ADF's as "Mutations": I don't see how Crossover could insert them.

.31 For optimizing sub-tree definitions (perhaps) use G-wtd. frequency

.32 case counts of sub-trees. Note that "x" must be quite large (≥ 5 or 10) before we can do substitution! But t. legitimacy of t. Defn. of a subtree can be maximized over smaller "x" values ($x \equiv$ observed freq. of subtree / freq. it is to appear $\equiv$ observed freq. of subtree / product of freqs of components of subtree)

→ (Woops! No! In this system, we can't make a "wrong error" so that we don't need very large x! Maybe parsimony error is good. Look at this in 'dim:

.37 A B C say AB occurs unusually by frequency. We define AB = α & do recursive sub. substitute AB ← α in all cases. Now,

.01 if BC mondria was already a very impt idea, we would be unable to "notice" that fact. So we may, indeed, want X > 5000 to do 64.32, before we do universal Substitution.

.04 After defining a subtree, it can be inserted into cands & be immune from destruction by crossover — Rep Rep the old subtrees that gave rise to the definition can be/still by crossover, (since we more uncertain of Rep persistence). → (68.18) → 68.18

64.13

.09 :63.21 : The idea in 63.08 – .21 is that the f(·) choice & tuning → use ended w. a SGA stochastic hill-, & the pc of a cand, is meant to be an approx. of its G (after G x from $f^{-1}(\cdot)$). The "definition" of f(·) is to be compared w. def. of f(·) at 63.35 – 63.13 : Perhaps main (not the "data") is one class of ways to approximate the desired SGA.

I think SGA is equiv. to using "C = G" on 64.13

("Mean G" on 63.39 and ~ thing for 64.02)

So, a way to "calibrate f(·)":

Get a bunch of cands: a "Prob" from into a Root Bag & Branch Bag.

.15 Generate to complete cand produce w. wts. = product of ? Case counts of Bags.

Of the corpus of Cands: Get G/Sof all of them.

For each $G_0 \pm \Delta$ we have a sub corpus of Prob cands. Obtain mean "wt" (of .15) for each G_0 : = $W(G_0)$; Probably not Expected value

This means that for each wt, we obtain a corresponding "expected" G_0 — And we have for each G, a corresp. mean wt.

Now, the relationship of this "wt" to pc, is unknown not apparent.

What I want for each G_0 value is a pc_0 ($pc_0 = f(G_0)$) so that when I use this pc_0 as a wt. for the root & branch Bags of a cand, then the card product of the wt's root & branch Bags, gives pc's that are as much as possible, in exact "true G" order (If f is monotonic, G order ≡ pc order)

Say I use G^{ms} itself for wts of root & branch components of a cand, Multiplying G by a constant, would be lower order invariant, but adding a constant, G → G+A would change things a lot!

(I think ? — G → G+A is equiv to multiply by $\frac{G+A}{G} = 1+\frac{A}{G}$.

.31: 69.07 A Gone for f(·) : We use this f to assign wts to cands on basis of their G's, We use those wts in the Root & Branch Bags for the entire corpus (for all encountered G values)

From the 2 Bags, we obtain pc on a fairly large bunch of cands. These cands have pc's for each cand, given by the stack grammar. Also, encountered G for each cand.

In this set of cands are ... For 10% of cands w. highest pc's will contain a certain fraction of, of the cands having the top 10% of all G values.

We want to adjust f(·) so as is as large as poss.

The Threshold T_G at which 10% of all cands have $G > T_G$, is ... of f(·), so we adjust f(·) so that we have max no of cands as

GA : Holland fixed length, finite Genome. ? repeat & crossover & mutation.

GP : more Functional/ long (Mess) Smoothness of Mutation Much crossover. ADFs (added later) Prog. w/ args, crossover fewer G, but crossover

.01 w. pc in top 10%, must have $G \to T_G$. We could get a nitrassmata by
.02 Sampling. This sampling can be optimized so one gets as fast convergence as poss'l. → .12
Mainly, what we want is that by looking at cands of highest pc, we
end up w. as many hy G cands as poss'l. This is impt., because we
.05 have an easy way to enumerate cands in pc order (highest pc's first). .25
Re: Hy frequency subtrees: The idea was to use this pseudo pc ($= pc_0 = f(G_0)$)
as a wt., equivalent as a multiplyor for SSZ, to obtain a stoch grammar using the "fixed data"
as the "corpus Bag". Trouble is: large SSZ has 2 aspects: (1) independent
hy pc (2) Getting rid of noise! Unfortly, we are only getting (1) when we do this pc₀
.10 wting — noise will be a function of true SSZ. (16)

.12 (02) Re: optimum convergence strategy to obtain f(·). Say f(·) is a (poss'l)
function of — 1 param is α. We get empirical noisy f for 3 values of α.
From this, we guess at best value of α. We then try 3 α values
around this peak, w. larger SSZ. The SSZ of these will climbing steps
can probably be optimized so as to get back poss'l α per se.
.16: (10)? Well, it may be that we are not much interested in noise level! —
Say we have this cands, consisting of a hy G root & a hy G Branch! Both of
SSZ ≥ 1, hvr. The main expected G of this cands is hy, but it has enormous
variance. We are interested in the hyexpected G, hvr, & much less interested
in its variance. As a result, applying ideas of 2161 on hy frequency
subtrees of hy G; We are mainly interested in the (yeast-x. — (But also note
64.31–65.07 — on "misperamy" and "immunity from crossover breakage" ("immunization
destruction by crossover").
.25: (02) Re: (curve fitting a f(·) function). It may be easier to do this:
Select out a large bunch of cands of highest G from the population. Then adjust
the params of f(·) so that this set of cands has highest pc.
This may be an Elitism, but the Q is — to what extent does it & diversity
(Re: Elitism). Does this really give the suspected badness of it of diversity?
.30 Woops! This would give "Baseball", but only → 67.01 → 67.01
.33 .34: 63.28 More on Elitism: I graduately delete (63.25–.33) a method of
using successive runs of less Elitism. An alternative: I run of hy Elitism;
as the ratio of climb slows down, one becomes more & more
"desperate". (Woops! Maybe all wrong! This would, I think, give "Baseball" 67.01
Out. pc's of Branches: For a given cand, all of its roots may have
about same pc; But it would seem that the Branches could have highly varying sizes,
& ∴ much diff pc's (Actually Roots can sometimes differ much in size).
However, I was thinking of simple: The system gives only one (mean) fraction pc. —
For a SSZ of a few, this is perhaps o.k.

.01: 66.30 if we give pc=1 for cands of ~~best~~ best G, & pc=0 for all others!

Hm, 66.2→30 may have to Gaussian a good deal more; Namely that was already in borderland on a "good fit" for Pi. by G sub-corpus of cands.

Persistence: 5+ → 1 w. diffy δ → 0 very slowly

USN Koza, in Perl paperon Fitness Landscapes & Design has 2 G that goes from 0 to very large. (its = "Badness" criterion). In that 1990 report, he uses to form $p_i = \frac{1}{1+B_i}$; Boltzmann normalizes w. $\frac{p_i}{\sum p_i}$.

Note that in spite of the fact that GS seems to assume that $p_i = f(\sum G_i)$ is Normalizable (i.e. $\sum G_i < \infty$; $G_i \geq 0$), this is not nearly true of all Genes used. $G_i \leq 0$ can usually be fixed, but $\sum G_i \to \infty$ can be difficult.

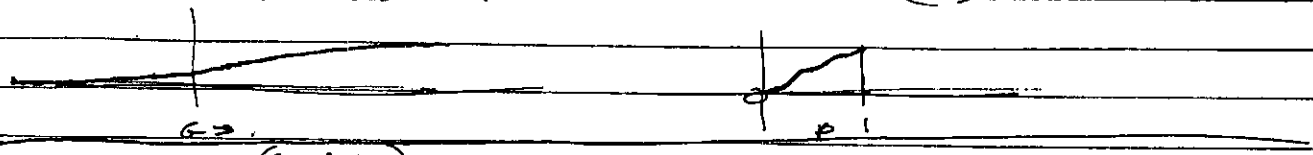
USN SGA can be thought as a General model for a large part of GA research. In the whole area, they are using a great variety of techniques, & SGA seems to cover most methods to vast majority of them. SGA is not really the only way to do GA .... (or the best way to solve 0 ≥ probs), but it gives us an understanding of much of GA & it tells how to improve it, & suggests where one method may be better than another.

On the $G \to p_{Gi}$ mapping: One simplest way to optimize $f(G_i)$:

top 50 percentile of G & of P: How many can in common?

:20 → Is Kolmogorov-Smirnov statistic Relevant.

Old for $\left(\frac{P}{G}\right)$ plot fraction of population $< \left(\frac{G}{P}\right)$ as a function of $\left(\frac{P}{G}\right)$:



:36 In fact, the (fraction percentile) of $G < G_0$ would map directly into probability in a ~~one~~ unique way! Also, easy to do: Just make a table! Maybe have logarithmic scale for p_{G_0} in table, so linear gradation for numbers p_{G_0} near zero.

T. table can be constructed very rapidly by sampling.

A Quick & Dirty way to make table "steady-state updated":

Pick highest & lowest G that has : divide into 10 = quants. $G_{i,}$

For first 100 cands, put case count (total) in each G_i box, (call).

At end of 100 cases, Make 2 functions w. 10 inputs: $F_1(G_i)$ = total case count up to G_i ; $F_2(G_i) = \frac{F_1(G_i)}{\sum F_1(G_i)}$ = Normalized pc's.

So do this & update for next then every 100 trials. This is much faster than updating every trial

In fact, one might update every time for first 10 trials, then every other trial for next 10 trials, etc (or some other scheme that more finely focused on minimum error per cc expended) — We might also want to change quantization spacing, etc.

See 92.32 for some Continuity of p_{G_0}

So that takes care of the pc assignment problem for a while.

① Another major Q is how ADF's work/should work! How ADF's Arbitrages work.

② How Koza wants us to do a three dimensional Network via a tree structure. This seems like an impl. way to deal. Rings.

③ It may be best to not bother w. crossover at all! Crossover is just a way to obtain Good sub-nets.

So a way to do it: Generate a bunch of functions in pc order. Those functs are G's. Evaluate their G's & assign pc's to them. View 67.25 ff.

Suppose we had a corpus of functions (cands) all of same G (so format C for a while. We could use something like zrt(to assign pc's to small sub trees. ◀ So we could get a X = no. of occurrences of subtree / " " " " expected via frequency parts. (Also renumber OSL, which zrt(can deal w. but it is tricky to apply to real problems)

When we do .13 w. a corpus of many different G-values we convert G's to pc's, and in both numerator & denominator of .13, we use wtd case counts, — in the denominator — simply sum of wts. used in numerator.

.18; 65.07 One (perhaps) way to get around the "Mississippi" problem in sub-nets (see 64.31 — 65.07 for discn. of the problem — a possibly an adequate soln.!) Another way: When one makes a definition of a sub-tree in a bunch of cands: One duplicate splits each cand. in which the subtree occurred in 2: One in which the sub tree remains as before & one in which it is replaced by a single function call (Rings not "suitable by crossover"). One effect of Ring is to double the size of cands w. duplicated subtrees in them: One would have to double all other cands as well, to keep nodes OK's

Actually the idea of 64.31 — 65.07 isn't so good either! Say a certain subtree occurs 10 times in the corpus. Its X is, say 5. —

Using auxiliary criteria, we decide that this is a good thing to define, we put it in the set of functions along with the pc.

Using standard ALP, we would now start from the beginning & generation a new corpus, using the new function & its pc. — The way to do this would be to try to find several useful functions per generation. Here, we don't use the definitions until the next generation.

This starts each generation a "bit" seeming very wasteful! — The advanced generations have new/larger useful functions to use.

A way to modify a cand. in a given generation, using a new function: Take in turn each input wire of the cand, & cut off its feed; connect the output of the new funct, instead. Connect the inputs of the new

.01 function to all possl. combinations of 10 other vrs (that were above it) in t. heap.

[Note: t. no. of combinations involving only 1 new function in t. next one enormous — So t. space of funds is, incredibly, enormous!

If a soln. can be found, it is only because there are many, many solns! —

→ Or because t. GP algm is much more powerful than suspected!

At any rate, every problem will have no solns of size ≤ a certain K, & it will be quite large for many problems. The superhuman large Lcost would have to be reduced by "going along a proper path" of "somewhat smaller ↑n G."

— This I would be to obtain a lowest Lcost soln. There may be smaller Lcost solns, that has what look like "Appendixes" — parts that are now in trees, but were necessary parts of t. path of "somewhat monotonic ↑ in G".

.12 For GA to work at all, the G funct. must be adequate. [I'm thinking of INV probs, that are solvable by GA because a Good G was used.] —

But what about OE problems, — in which the G function is given as part of the definition of the problem? — In those cases, one may have to modify t. G function to obtain a good soln! E.g. for early parts of search, / low G use a different G than for later parts.

.13 — An example: "Curing" Cancer! Say we don't realize that we may not be able to cure it (an INV problem perhaps), so we settle on Herbal, or "cures for most people." To get better, we probly want to first get to understand human biology & how cancer works: There are OE problems, that have a different G funct than "Curing Cancer". → Unless one has a G funct that — Somehow realizes that progress in these subjects is also progress in t. top goal. Say t. top goal is by % of Cancer cured. In getting to understand Human Biology, we haven't done anything to advance t. top goal! was lacked. — So t. top goal G would give us no feedback at all!

Consider early Alchemy! (t. part that wanted to change things into Gold.) There early work really did help progress in Chemistry, which helped Physics, which finally gave Transmutation. — But could t early Goal of changing things into Gold — could that sustaining goal really help/get Physics & Chemistry for most of their "PATH"?

Could this "modifn. of t. G funct" be an Essential Part of TM₂? Well TM₂ has an inh'l problem of Getting t. system "ON T. HILL" — where By. slope to right of G is decidedly >0. — But also TM₂ must not cease if on an area where flat local Maximum may be very low w.r.t. final goal. Anr. in (if) (curing cancer) "TM₂" pursuing given t. system some subgoals may do not directly advance t. top goal. T. relationship of t. subgoals to t. top goal is inductive, not direct. → 70.01

.01: 69.10: IN INV probs, it's clear that TM2 [illegible] lot of work to do to construct a useful G function for a stated INV problem. For OZ problems, we are given a G function, but this G function may not be adequate for solving the problem. (As in 69.18 — "Curing Cancer") The given G that may be mean [illegible] only for final framing of the soln. sequences. In general, we are only interested in the original Very hy G [illegible] portion

Ideally TM2 should be watching the progress of a TM's "hill climbing"

.07 & possibly "giving" it goals if necy. (WON problem?)

→ Perhaps [illegible] to MCT treatment to get better understanding of the walk.

Given int. Problem Definition — But in many cases we will refine & augment G in the course of making better G

.10 Another way to look at it: that all probs have either "Hard Goals" (INV) or Soft Goals (OZ). That TM2 treats them both in similar ways — that TM2 recognizes that the G of a "Soft Goal" is only relevant to "final tuning", but it can be used heuristically to get both subgoals, or, in many cases, it can be used to guide the entire search.

.15 (21.01)

Some Books to Review &/or Buy:

(1) Genetic Prog III Darwinian Invention & Prob Solving. March 1999 1184pp $79 Amazon. This is a set of new applications: chapters on Architecture Altering Operations Analog Ckt design. Discovery of Cellular Automata rules. GPgms in Molec. Bio. Parallelization and compns.

.20 → (2a) Advances in GP (Spector, Langdon, U-M. O'Reilly, Angeline, eds. Aug 1999 500 pp $75. Some Recent New Applications: 59

a) Robot Language Interpreters; Timeseries Modeling; Quantum Computing!

b) Theory: Fitness Distributions; Analysis of Single-Node Building Blocks; on "crossover". Revised Price Schema. In general, looks interesting.

c) Extensions of GP

Solo 2PM four stars.

→ (3) Prob'ly Get Koza II (on ADF's): $65

→ (4) Angeline Peter J., Kinnear Kenneth E. eds Advances in GP $62 538pp Oct 1996 Very Good Buy

.28 RE: FITNESS Functions: Koza starts w. a fitness that is zero for perfect hits, & → 1000 for bad hit (Like MS error). Then Koza does $\frac{1}{F+1}$ which puts it below 0.1 & then he normalizes so $\sum G_i = 1$.

This may be fine for the kinds of problems Koza solves; in particular one has a vector Gene, (as in GPS); one can add the [illegible] errors to obtain his first fitness function. This makes it possible to adjust by relative importance of the different Gene Components. The shifting of the function into 0-1 & Normalizing is perhaps not the most important part — yet it is what 67.25 does well.

Re: 67.25 ff: I'm not completely certain about what this Gen + PGA foundation does & whether is "best" or whatever

5/18/00 Bulg.

Bookmark 1.06

.01 : 70.15 : For problem of 69.12 - 70.15 : A reasonable way for TM to solve both ENV & OZ problems: In both cases, TM2 looks at the problem & designs a suitable G function. [70.10 - 15] As the working of the problem continues,
.04 TM2 can change the G function if it seems [redacted] expedient.

.06 A summary of the present state of the (MC GP) TM:

1) The form of $G_i \xrightarrow{CO} PC_0$ 67.25 - ff seems adequate for the while:
also Note 70.25 - .90 on the shape of G, the original fitness function.

2) The problem in both MCT & GPTM is divided up into a TM1 part, which does search in MCTM; or GP in GPTM. II and a TM2 part, which furnishes the original OMC (or PD) in MCTM, & which furnishes the early cands, & ADFs & designs the G function for both OZ & ENV probs in GPTM
(See 71.01 - .04 on Reisler)

.19 3) For GPTM, I'm not sure that Koza's crossover + ADF's is anywhere near as good as it could be. I have a good formalization of crossover, but I don't yet understand ADF's very well.

In Koza, 1994 discusses Koza's ADF's
1994 compares Koza's ADF's w/ Angeline & Pollack's "Modules":
No explanation why ADF's create such much better, — the superficially they are very similar. The difference may be that Koza used better (hyper) mutation. For his ADFs than A & P. But I must understand this & see if ALP can propose a much better way to do "common subtrees".

Also, I must be sure that my Model is able to do **OSL** (recursive size = 2)
→ Koza Has expressed the successive Modifications of an Electrical network as a trace of functions: Also, he may have devised other creative languages for other problems. I should understand these techniques, so I can get TM2 to Learn them. See his paper on "Evolutionary Computation" July 97, for some of the stuff. Book Koza II (on ADF's) may have much more this Koza III has more advanced applications & relevant languages.
In General, I should read many different applications (languages) so I can teach them to TM2. So Advances in GP (70.20) for More unusual Applications of GP (e.g. "A Robot language interpreter")

Probly the Most immediately critical problem re .19 : Getting a good version of a GP design.

Some "End Notes" on GP TM

If these 2 "improvements" don't speed up GP much, then it's because (& in Direct Sys (2) probably reduced attributes are very poor!

1) In searching via probabilities; P_i is not a good value to use. Expected time to discovery of N ($\subset$ N = no. of cands). $\sqrt{P_i}$ is better, but does tradeoff P_i order is best. Try $G_i = \frac{\sqrt{P_i}}{\sum \sqrt{P_i}}$ to compare w. $G_i = \frac{P_i}{\sum P_i}$ & possibly an easy G_i order.

2) How Good is the Cart product of Root Bag & Branch Bag as a Grammar for the Corpus? This could be decided via ALP. The fact data is unordered, it is probably possible to do an analog of the "Exact" codes used in Z141.

.08 PC of a particular cand would be $\sum_j N_j^R N_j^B$: $N_j^R N_j^B$ are t. case numbers for the particular Roots & Branches of that Cand. The PC of the Corpus would be the product of all of these for all of the cands.

This analysis suggests that the corpus must be a min SSZ before this Grammar can be expected to be of much value. It would be good if I could estimate this SSZ. — However, the effect of different "wts" for different G values is certainly important, but I don't yet know how to deal w. it.

WOOPS! The Summation Formula of .08 makes it clear that the Method I'd planned to use for (1) finding the cands of the next generation in PC order, would have many duplications! I guess it would have a "ratio no" of dups, for finding the corpus, but it would be very unpleasant to have to recalculate the G of each of these cands!

Actually, I had planned to only include $N_j^R \cdot N_j^B$ products of ≥ 2 certain threshold: This would not be, however, true for "most likely" cands, but may be a good approx of that "Most Likely Bag of cands"] — [But not a Bag of cands, since repeats are usually not necessary & Maybe "Gonbag" (Generalized Bag) .208 → 73.08

(3) Could I use a Monte Carlo Sampling Method to Estimate the "Efficiency" of a Grammar) — or the PC of a corpus wrt that Grammar?

For TM2! When working on the $2^n + k$ bit multiplier problem. After doing $k = 2, 3, 4$, say, TM2 should be able to "Notice the Pattern", & do $k = 5, 6, 7$ very quickly!

The Crossover systems of GP used by Koza, seems to be a very primitive way to Evolve! It should be possible for TM2 to try to find better systems. However, if it used the usual "trial & error" routine, the feedback will be very slow & very noisy!

— But it could do experiments on small problems (Say it took ~ 1 second to solve this "small problem" — it could do M trials in $\frac{M \cdot k}{3600}$ hours — 3 solves/day — not at all unreasonable. — If it found something that looked good, it could keep it on (longer problems). — An important Meta problem is to devise a language that describes crossover-mutation schemes that gives by PC to present methods & variations, yet is Turing complete.

.01 We have to make good models of TM_2 behavior available to TM_2 for free. This seems about as serious as the "TM_2" problem in MCTM: We look at a bunch of OT's & make a (very fast expression from Complexity Theory try to make the "Turing Complete" Theorem we do Learning in this space of OT's Copyright techniques). — Hmm, a pt. that I hadn't considered many yrs. ago when I was first thinking about TM_2's improving OT's: That I have to include models that decide which OT methods (or a probability ordering of OT's) for each problem domain.

.08 : 72 24 : A possible approximate fix to be: Instead of using the $\sum_{\sigma} \overset{R}{N}_{\sigma} \overset{B}{N}_{\sigma}$ of .05, use the single j for which $\overset{R}{N}_j \overset{B}{D}_j$ is Max. I could just do $\overset{R}{N}_j \overset{B}{N}_j$ pairs in order of size of product. Many such pairs will be of the same rank, but the set of all $\overset{R}{N}_j \overset{B}{N}_j$ products > some threshold, T, will include many pairs from the same (usually big G) rank. It would perhaps be desirable to pull in 2 distinct $\overset{R}{D}_j \overset{B}{D}_j$'s of the same rank.

5/21/00 AH! Just re-read Wolff's latest dated of ~2/9/00, Very Good! I do have a "parsing problem" in grammars of trees. Wolff's work, Hmm, suggests how to deal w. it. I just make up sub-theories: a larger set (well maybe not so large) of theories, on bases of expected Δ pc; Then using these expected pc's of the subtrees I try to parse the whole tree for that pc.

I then try theories w. the "optimum" parsed tree that both up on the old corpus. A make a bunch per "round", then I reparse the whole new corpus, using the new plus old set of theories. So every time I make some new theories, I reparse the corpus, This "reparsing" is/forms A Great BREAKTHRU!

Hmm, I Also want to discuss ASSISTS (& counterexample,) & I should do it now to show why I'd been planning to do it. Re a corpus of sequential symbols.

This seems very much If it really has to do w. higher order regularities than simply common sub-trees. If properly done, this should put us far beyond capabilities of ordinary GP!

Possibly a better way to look at to ALL our speech processes! There is this true stochastic larger over all possible sounds. We create samples of it, & we get a corpus, which is related both actual pc of each sound, but in a "local" way ("local" w.r.t. the total sample of the corpus known thus far). The way we proceed is to make "local grammars" over the known samples of the ensemble.

Perhaps study effects of crossover, mutations, etc, on linear (or other than trees) Genome: (We understand that functions are never "broken".)

To what extent does a "crossover-crossover" approximate a real GP? Crossover genomes do tend to clump together, synergistic functions, but to a lesser extent. This is regular organic circuits, in which the functions work much in ||, somewhat indep. IN TOZZS GP, functions are very much sequentially activated. → 75:01

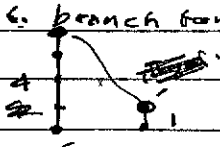
One key Quality of the crossover idea is that it seems to bias toward ↑ in size ("Bloat") — But I'm not sure, Crossover can produce trees of very small size also.

→ Actually, in absence of selection pressure, crossover keeps mean size of Genome invariant! The argument is that I replace an average sized branch by an average sized branch. → There is against this: that a longer branch has more targets. ∴ is ↑ more likely to be chosen, than a short branch. This fact is the clincher! Rate of Growth maybe ∝ size ∴ exponential Growth

← (But this has to be checked). $dl \propto (\bar{l})^2$; if so: $\frac{dx}{dt} = ax^2$ $\frac{dy}{y^2} = a\,dt$ $-\frac{1}{x} = at + C$ $x = \frac{-1}{at+c}$ $x = \frac{1}{c - at}$

Say long Rx are x_1 w. prob p_1 ... x_i w. prob p_i.

On the other hand, the root tends to ↓ in size just as much as a branch tends to ↑ in length! So mean stays constant!



Say we only have 1 branch of its of length l.

So it looks like mean length is always ↑. — No! If Seems so by the same!

→ If we add branch at at random pt & add zero length to it. The mean and length of the result is $l/2$. If we add an average of $l/2$ to each broken branch, the mean total is $l/2 + l/2 = l$.

I still don't feel confident about this — overbrag on the bias; the only thing that seems clear, is that mean length doesn't ↓!

0 1 2 3 4 : length 4: mean root length = 2 ; Mean branch length = 2. mean sum = 4.

Why Say mean size is constant the successive generations: Longest length card in each generation doubles in size each generation, so the frequency of these very longer cards must be very low. (Note that this w.o. any selection)

5/21/00 Bulg.

~~5/19/00~~ On Utility of crossover in ~~Genetic~~ w linear Genome:

Say we have 2 genomes: α: A B C , β: D E F (A, B, C of α are ~~genes~~ substrings)

to get the "good" E from β into α, between B & C:

ABC → AB .. C DEF → D .. EF

AB, EF → ABEF. Then we have to exchange Y. terminal F w. C

DBE, F ; → ABF, C → ABEC.

This is a lot of crossing over. 2 crossovers needed to do ABC → ABEC

But actually, this may not be so bad! If E is "good" we also want to retain as much of its environment as possible. E for "Branch EF or the "Root" DE would do this.

So MAYBE crossover is not so weak!

Thought quick mention of ADF ; Asp Modules: (AGP #1 K. Kinnear)

But first: I'd like a form of GP so that any derivable form of learning or functional form, is learnable by the system. I think that my NL lang (Vistlisp) functional lang should be able to express all prim rec functs ... but can it simulate any & prim rec funct. w. a finite # of Root functions?

What about "Distance" measure in simulation instructions" space? One function is "close" to another if we don't have to modify one much to become another. Unfortunately, 2 functions that are "close" in this sense need not have I/O properties that are at all similar! So how can GP work if this is true??

It would seem that suitable Lisp would be much better in finding "New-bf functions".

On Hunting for common small subtrees: This might be done by the same way we find common words in a linear text: Every time a function occurs, we have a unique code for it & use it to store a unique pt. in Memory for its (cargo bounds cset.)

Def

Alternatively, use B trees to store cset 'info'. This takes longer, but will (perhaps) use much less RAM. Another way might be hash coding (Also uses ("invertex") much ram — but there maybe efficient Hash codes that use Ram. More efficiently & still have few "collisions".

Also, if there is a symmetry or (commutativity) on the inputs, we will want to either reflect this in a common storage point, or be able to pool the several cset's after we've (analyzed or bin sorted) the corpus.

Note that the # of new functions will be ↑ as we expand the corpus — so we need an efficient cset storage method that can deal w. this.

I think I solved the "+2" mirror problem (starting w function "IP" only) by Lisp's finding common substrings: Prog last was which become ADF's for next generation.

5/22/00 Bulg:

In ∃ Alg "ep": See if it can find a proof of ∃ ∀ G or hyPC.
(i.e. short long Pr.).

Prf of hyPC means hyPC from one "generation" to Next. — conditional PC.

Also, Consider || pgRs. — This looks like Metabens, hmm. !

Is it meaningful to have conditional pc. from (x,y,z) to w?

I think so! it ≡ $\frac{pc(x,y,z,w)}{pc(x,y,z)}$ or ≡ $\frac{pc(\text{population}_2, w)}{pc(\text{population}_1)}$.

→ Maybe V.G Idea!

This may be ∃ correspdn. via Stoch Grammar construction! ☺!

or So t. Q is, can t make much better Grammars (perCC) Prod t.
"Crossover Grammar" (+ t. ADR Grammar)?

"Cloning" a TM: For cloning a TM for a new installation or on a Faster computer or a new kind of computer: It might be O.K. for cloning only the TM_2 part: So TM_1 would not need direct access (or maybe any access at all) to the original TSP's of TM_1.

Also TM_2 could "work on itself" to adapt itself to the new situation.
(TM_2 is the stochastic [Generator/Grammar]) that has a problem
darn as input & has as output plans on how to solve it, ['hints'],
Cands, Refinements of Cands or Partial Solns, / new G-Productions for INV
problems over new G for OZ problems.

We might store TSP's that various clones have used, &
later distribute them to other clones.

L search seems to solve INV problems only.
GP Solves " " OZ problems only

These appearances are deceptive: GP really solves INV problems by ∃ L-srch:
It makes a local model of Condition vs. Gr vs. pc & searches that region
of hy G hy pc for more cands of hy G.

L srch in INV problems would work for simple problems, where
f: CJS was small enough — i.e. TM was close enough to the soln. to start off.
For problems in which this is not true, (Most INV problems)
∃ G function for the INV problem is constructed, and we work much like in GP.
The details of this are not clear, however.

5/29/00 Bugs:

A critical Note about TM_2 (T. horizontal GP).

Even each initial problem Dern, TM_2 Devises a set of good candidates and a set of functions, — essentially a "metric" that tells what candidates are close to what other candidates. As the solution proceeds, TM_2 may modify the metric, in view of TM_2's "previous experience." Part of the modification is in the stoch. grammar that derives the corpus, but TM_2 may also add and/or delete functions used in the stoch. Gram.

Actually, TM_2 will be constructing functions of its own, so TM_2 doesn't merely have to do Reit.

Presumably, as a population gets better & better, TM_2 has a better & better idea of what the Metric should be.

One view of this TM is a SGA that really doesn't use GA at all! It just updates the Grammar periodically, & then produces new candidates, until it has an adequate no. of new cands of hy & then feeds the grammar in a new direction.

There is the problem of just how to wt. the data (= cands). We do have the pc associated w. G, but we also are concerned w. size: I head do Reich that a complex pc with cands will not generalize — but I'm not sure — it may work fine!

So a dominant problem now is PSG-discy. I had a good idea on how to define some syntax (= contexts). Try to find it.

Perh I think the idea is just for each word, terminal or the nomb (symbol) that follows it. Then when size is low, we can add these nomb's so that predictions what follows those words, is more accurate & hyer pc. Words that are followed by the same nomb form a nounset. Say $[A_i]$ are the set of words that are followed by the nomb $[B_j \cdot w_j]$ (w_j is the coeff of B_j)

So the cart product, $[A_i] \times [B_j \cdot w_j]$ is an nomb.

Note that we can do this cart mult back. to a nounset & an nomb, or back to a noset & nomb's but not back to nomb's, unless we first normalize one of them (maybe) or both of them!

This idea of product of nomb's — each of which represents a sub-category. The case would seem to (, since they are pc's. As such, we can take cart products & get a new sub-lang (after renormalization). Actually by "cart. product" I meant the other of concat of 2 sublangs (= POS's = part of speech's)

$\Sigma \to \alpha \cdot \beta$: β & β are sub-langs = normed bags of strings.

Anyway, the "cart product" of .27 is an nomb which we can normalize; it is a kind of "pos." (= sub-lang).

We can do similar mult nounsets like $[A_i]$ & nomb's like $[B_j \cdot w_j]$ in .27 via the defns .25, .26.

5/24/00 [redacted] Bulg.

F:\Nonsense.olg Downld\RJS.001
ℓ 224 General on early
ℓ 430 evolutionary Computation; & DIMACS workshop
list of papers:

After we have developed a lot of nguists & nbags, & we find a few multi rules per Rcon, we might try to see if certain of Rcons approximately sum to others.

Grammar-wise, this would be like:

[redacted] $A \rightarrow C \cdot D$
$\rightarrow E \cdot F$

A is Rcon to sum of the 2 nbags C·D & E·F
C·D is cross product of nbags C & D.

Maybe this will work!

nbags can also be defined by other kinds of "contexts" than $\Sigma \rightarrow \Sigma \cdot A$

A context can be a pair of words surrounding a selection. [I may (need/want) Base more complex "contexts" in analysis of trees.

Well: I did want nbags (≡ upon bags) to be defined so I could see if they were used "contexts" for other nbags defns.:

But: can I get recursive? e.g. ($\Sigma \equiv$ Null)

$\Sigma \rightarrow A$ [redacted] (where A is previously defined nbag.)
$\rightarrow \Sigma \cdot A$ [redacted] or $\Sigma \rightarrow \Lambda$; $\Sigma \rightarrow \Sigma \cdot A$.

For finite nbags, $\Sigma \rightarrow \Sigma \cdot A$ can't be true exactly, but it might be true for the hypothesized part of Σ. — With this, if we have the true original Σ, we can see if SSZ warrants the extrapolation to $\Sigma \rightarrow \Sigma A$.

One (perhaps most important) Q is: is $\Sigma \rightarrow \Sigma A$ a cheaper defn. than (an alternative defn. of Σ) we could write $\Sigma \rightarrow \Sigma A$ and, as "errors" add other nbags to Σ. Having Σ too small will have corpus derivations $pc = \phi$; having Σ too large, can have pc of observed corpus be > 0.

DEFS: New set of defns: BAG ≡ bag of strings: nbag ≡ normed bag of strings ≡ stochastic sub-language = SPOS (stochastic "part of speech")
nbag = SSL = SPOS = Pattern [Non-Terminal OR Terminal] [usually Non-Terminal]

SN It may be that OSL (≡ {R=2}) is the most common kind of human (or any other kind) learning. It seems very close to CASE BASE LRNG CBR

SN My recent analyses of Wolff's MK10 Re parsing after every 4th (or after several) new definitions would be also very irrelevant to coding the corpus of Cruds in MC TM.

Two deficiencies (perhaps) of that approach to defining sub-trees:

1) It may not generate recursive rules well, & so it may not do all of the things Koza's ADF's do.

2) It may not implement OSL.

Wrt OSL: $\exists N \forall Z (\ldots)$, the way I implemented (prior OSL [redacted] w. $\Leftrightarrow > 2$), I looked at all of the suffixes of the (sequential) corpus, & put counts & used each suffix as a code for the corpus, & got all procedures from all those locations.

Could I do anything like that in CM GA TM's subtrees?

A possl. way to do this (equivalently): Precisely any subtree of G covers; (particularly large sub-trees) has a reasonable probs of occurring again (as a "OSL") in a new Con
This means that if we are running a System Mt Carlo-wise, we can pick any (usually) large sub-tree in the current & build a [redacted] new Cand around it.
.05 —— Hm, we mainly want to use very large subtrees, so we don't have to make too much C.V. (ppc) to complete them as [redacted] (complete) Cands (i.e. having prior inputs & outputs)
Well, it may well be that any Branch of a Cand, can be used as a Cand, [redacted] — Since it has legal inputs & one output. — Yet, since it (usually) doesn't use all of the inputs, it would seem to be a "bad bet".
.11 It suggests that small modifns of any big G cand, would be good Cands to try. "Small modifns" means all G. some but some functions different.
— Some small sub-trees or [redacted] small branches different.
Actually .11 is what "Case Based Comp" is!
Another way of looking at it: We can do various xfmns on the [redacted] of Cands:
These xfmns don't change the Cands Much (in an Info sense). They can be analogy xfmns or Substitution/addition of (a Branch or Subtree).
We want TMs (or TM[redacted] proposal) to be able to learn just what good xfmns to use in each "Problem Domain".
In Koza's designing of Electrical Networks, Essentially what he has, is a way of Modifying a Network. His form of a Net is a kind of "Modifns". So he have invented a powerful set of xfmns for a large domain.
—— SN So 2 Points I need to understand well about Koza's moMods (1) Just how his "Net modifn" loops work (2) Just how his ADF's work; Why they are [redacted] way of use than Rip's "Modules".
Re: Xfmns, Modifns: I will end up w. several "modified" versions of a <Cand or Big G>: Some of them have big G, I will find more modifns using cheap; I will continue in this direction until new modifns become expensive.
So: I start out by generating a very large initial corpus by combining G. basic functions in a random way — until I get "on the hill"; I then begin to "Modify" the big G cands (Please see link "Modifns")
→ "Crossover": I'd like to be sure I look at a bunch of big G cands; suppose promising offsprings: → 80.01
35

5/22/00 Bdg:

Re: Optzn problems in GA: TM_2 will suggest several [redacted] OTs — (just as in [redacted] such). Hvr. in GA, an OE is defined by [redacted] an initial corpus of candidates & methods to go to & next generation of candy. (or a steady-state method). This will include mutation types, methods of "crossover" (or exchange of sub-trees), sets of functions to use & ADF methods (?).

→ Do details of OZ soln in cold MCT analysis. To what extent can it be worked into a practical soln?

Note that TM_2 has more time to do things than TM_1 — Also I can help it.

.08

Re GA analysis: Try listing OSL contents of a [redacted] sequential corpus in pc order Then try doing this for trees:

A possible trick: Put pc's into Huffman codes, then list all strings of length K. — these represent the Huffman codes of the corpus of $pc > 2^{-k}$.

Re TM_2 to "jump start" look at it's problems (Say 5 sort of probs to be solved) & Pascal soln (Macos). Try to solve TM's pgm by hand & factor to some so TM can use to factor as language for search or maybe "Optran search"

For [redacted] "Tree" OSL: Switching Branches would seem to be one way to do OSL. $R_1 B_1$ & $R_2 B_2$! → [redacted] B_2 do MX $\frac{1}{2}$ sec.

→ $R_1 B_1$, $R_1 B_2$, $R_2 B_1$, $R_2 B_2$ enough since Pascal knows symmetry. 20 M

$\begin{pmatrix} R_2 \\ R_1 \end{pmatrix} \begin{pmatrix} B_2 \\ B_1 \end{pmatrix}$. SSZ=2 for [redacted] R_1, R_2, B_1, B_2. 800 MM

10^{15} cycles

In universal, Pascal, it would seem that some kind of "internal" subsitution or mutation of a subtree would also be a kind of reasonable "OSL". 10^9 x 1000

10^{12}/sec

In general, GA() want to look at kinds of operations & read in "Crossover prob [redacted] solving — see where I can span fast space w. a certain set of Xpons.

.30

Some General Ways of Prob Solving: (1) a universal or almost TM (universal set of instrs (things combinable to make pgm-strands)

(2) Use of Lisch or maybe (as in Eliza) search method.

(3) Learning or Redundancy TM_2: Modeling: [redacted] improvements in ordering by noting regularities in previous soln traces, TM_2 need not [redacted] have a complete set of instrs for evolution. It could use a very permissive stock (say, & yet improve TM_1 tremendously. If TM_2 is macro-general & uses [redacted] the complete logic of TM_1, it can go v.g. (if TM_2 is a piece of [redacted] TM_1, it can be even better (

5/31/00 Bulf

Hermann Helmholtz ; Ludwig Boltzman } They were a different people. I always get them confused!

Were they contemporaries? did they know each other?

.01: So a basic area of research is to find canonical kinds of TM's? Can we computationally put all types of GA's, Ev. Algs, (Neural nets (Recurrent ANN's)) into Bias Models?

Re Jürgen Schm: He doesn't really address the Horizon problem: His system ("Lrng How to Lrn") simply uses as short a horizon as possible. It can't do much in the way of "experimentation" (i.e. doing things that don't directly but may over long term ... This "long term" is a kind of Horizon).

To "trouble shoot" a proposed Lrng System, look at various things a human could try (including the human's learning method). Could this be found by the system? Could it devote various human hours to use Beam Rider its own problems? Could it use Gen methods at all levels of hierarchical problem solving (i.e. $TM_2 = TM_1$).

I've been taking a rather E1 view of TM_2 — that it was simply trying to improve OZ methods or ENV methods:
A basic E1 TM_2 looks at the past solving process (& successful solns) & tries to find some ways to get this goals faster: i.e. it looks at original problem domain & (semantically, if would decide whether it was an OZ or ENV problem; Non-el, if would not make such a decision). It would look for Gen. Pgm that would try to get good solns as fast as possible. This is in the spirit of OZ solns by Lsearch: The system looks for V.G OT's & uses them in attempt to speed up future successes when using the current Gen.

— Just as the OZ problem does, TM_2 will use one main Model for TM_1 (at any particular time), but will occasionally also try a fair amt. of time on various other possible TM_1 models.

Q: Could such a system look at the whole TM_2 - TM_1 system & try to optimize it? Maybe unnec: TM_2 has all options available to it in optimizing TM_1! If the TM_1, TM_2 structure is necessary for TM_1, then TM_2 should be able to find it!

However, this gives us the problem of "Just what is TM_2's Goal?" — a classic problem that I've written much about! Conclusions? — I think I solved it. — Part of soln: that user must give TM_2 its "Horizon" & perhaps other parameters. TM_2's Goal is to try to do so expected total Goal in future (say up to H in future is longer exponentially. our preferences decided by User)

Some random Notes:

1) To code a corpus, instead of trying to get 100% accuracy upto the bit before trying for 100% accuracy, am I right? Just try coding as many bits of corpus as poss'l, using as few bits as poss'l. As t corpus being considered long Runs, one codes more & more of the corpus (not nec'ly a byte @ a time, hrr) ..... As t. corpus long Runs, certain coding models that means doesn't actually compress, will compress. A simple example of a corpus in which 1) is applicable: We have 3 SM stock sequences. T. data is known sequentially as a single sequence $S_1(t), S_2(t), S_3(t), S_4(t+1), S_5(T+2), S_6(T+2), S_1(T+3)\ldots$

2) This could be important in coding a 2 (or more) dimensional image or a Bwah dim. into.

3)

In TM we want it to be poss'l. for TM to become a "Summary Machine" of arby kind. It should be able to have arby functional forms as declared objects, that can have "arby by" pc. (Easy as good as original primitive functs), Those may have this, but don't. Funct's get into PC's & are Risky considered "fast (wrt. SIZE) enuf".

T. way Koza's ADF's may get OSL! (?)

On the amount of "SMARTNESS" of Koza's pgms:
The problems of Analog IC & filter design — After Koza put in the initial info about kinds of functions, the task of the Machine was not so diff'lt. Multidimensional optimization w. some constraints, (Sometimes discrete) parameters. However, it thinks So, I should be able to design a System to solve these problems much faster (if not, Ray can (Comes) "Hard Problems" (.27) (.33)
I should be able to have the System start w. a large no. of primitive functs. From after a reasonable training period on simple problems, it should be able to assign reasonable pc's so those (& newly defined) functions so that the PC's of soln's are reasonable.

(.33) (.27) Some "Hard problems" Breaking down a big problem into smaller sub-problems for which unexpected cc's are small.
Note that when TM estimates cc's of soln. of a problem, it also can make suggestions on How to solve the problems: Both These are tasks are closely related.

6/1/00 Buie

Kinnear uses Press (p120)

No: Koza uses Press

{ In progression is an (integer) ≥ 2 (2, 3, 4, ...)

It may be easy for Mathematica Koza 1992 ! Also chapters 20, 21

on Modulas: pp110, 601

T. main diff. arises in my analysis of GA: I the more or less understand the effects of Root/Branch crossover & how it relates to OSL: I do not understand ADF & how it relates to frequencies of occurrence of Subtrees in Grads.

I don't understand difference between ADF & Alan's "Modules" (= MA)

K. Kinnear's book: p 225: Table: p127 describes of features - Tot how Basic (w.o. ADF or MA); MA, & ADF differ.

I think I understand G. "features" but I don't understand the quantitative aspects ---- Probby of each feature being invoked

Ibid p129 4 suggests Koza performs optimization in: (This is known class A.I. expert systems: "I. more info put in, the better it works (for a specific problem).

One very direct part (to/of) TM! Just write Pug TSP of dolas & problems in Algebra. I do know how to get pc's of combined conics. T. methods of 2nd #1 should be adequate. Particularly w. "Ropersy". Then try harder problems: See how I work them, try to discover heuristics & then devise TSP for the needed heuristics.

A possible approach: Look at hard problems that have been solved by GP (or other Machine Learning methods). (Actually, Case Based Reasoning can occasionally solve hard problems — This it is usually crazy from an AI P. pt. of view). One trouble is that Koza's Linear EC's & Linear-filter problems, (as stated) would require ~.5" for each Gen/Evaluation for Sin (way, maybe .05"), still even 1K trials is 50". Unless my own methods are ~ (4 times better than his, 800 trials → 80 trials → 40 seconds etc.

Even Koza's pretty problem of the "Broom balancing" took enormous time for each "G" evaluation.

One way, would be to take a problem w. an expensive G & try to make a model of it that is very inexpensive (like Irving Fox ridge bots). In the case of Koza's problems (Broom Balancing, Linear EC/filter design) This would be an interesting approach! For the early parts of the Search: use fast not-approximations to G: — As it runs get better, we eventually use the final expensive G — but the total cc is

<<< Using the expensive G every time!

Trouble is, I have no idea how hard the Spice approx problem is.

For Koza, This is clearly the best route, since the Spice approx could be used to speed up all of the network problems.

For approach 2 single Spice problem, probably the best way would be ~ to reasoning behind this: first Gather some data to make rough model of "Spice", — Basically, one first guess & bout some

Insights for the early attempts on a particular network problem (say IC design)?

SN It might be possl. to get a general soln. to solving most works of LCR's: (actually, those used only LC nets, essentially). Perhaps series, || forwards plus ~~not~~ xfmrs would solve all LCR nets. — It may take a while, but not ~~[illegible]~~ .5" = 40M clocks per G. early!

Could t. same Remy be done w. N.L. nets? Maybe w. 2 terminal nets. — static (rather than dynamic).

→ At first, I test my Spice model & network model on true Spice. I want it same answers on my Spice model as true spice, for me sort of initial network problem trials. ∝ Using my Spice models I use GA to rapidly devise some networks that seem v.g. wrt it. I then test t. v.g. models on true spice & try to get my model of spice so work well on those models. loop to ∝

T. main point of the exercise would be to find out if my basic method was better than Koza's — whether he had some ideas that I was missing.

Module Aquisition

One possl reason that MA did much poorer than ADF: That in MA, the modules being defined were not very good per unit size — i.e. lots of introns. That in ADF's had many fewer introns. Is there any reason to believe this is true? Well I don't see this making any difference! But ADF's could be larger than MA's After t. system has run for time T, t. average ADF will be the size ≈ size of the Result producing Branch ("size" ≈ "depth")

ADF's, however, will be chosen Modules ~~that~~ are from the "Result producing Branch" that are perhaps usually of depth < t. full depth of R.P. Res. Prod. Branch (i.e. they will contain some (genes)

A kind of OZ problem that I have not studied: The G function is "open" to TM. This could be useful in many of Koza's "hy G cost" problems. — e.g. Filter design, Linear IC design.

→ From reading Keynes's chapter, I get the impression that when Cowan did a crossover that caused a node to a node from Gen. to Gen., that the performance to ADFs was from the ADFs in Gen. 2 this seems insane!

6/2/00 Bulg Jun 5,00 | Blue Gene (IBM) | 5uo x speed of previous fastest computer = ASCI Red at Sandia 85
so ASCI's one only ten times faster than Deep Blue!
1k x fpc of Deep Blue ! expect to take 3 yrs to build.
2M x fastest desk top comp now available
#1 soon reach projects.
10^{15} ops/sec ——— 1 petaflop ; one quadrillion ops/sec.
4x total fpc of 90 fastest machines in the world
ASCI = 2 Teraflops 2 trillion ops/sec
T2 of Petaflops. = 2×10^{12} each
Normally x1000 takes M 5 15yrs via Moore's law
T 12-13
SMASH = many, simpler Self Healing.
1M proc. each at 10^9 Hz.
32 µproc per chip.
64 chips/board,
∴ 2k µp/BG.
= 1 pc of ASCI machine
8M simult-fpms
∴ 8 simult ops per µproc.
RISC machine
Orall will be on chips.
each chip is 2cm x 2cm

In Kinnear's Chapter: His treatment is mainly empirical: Mx centered on 2 problems. p 125 (ibid): § 6.5.1: A sorting problem: In which ADF degrades performance but MA produced no change in performance.
The even 4 parity problem: Even when MA produces no change in performance, but ADF much enhances performance.

.07 SN [redacted] Does my SGA system suffer from "self-conforming hyperplanes-ness"? What I do: I have this corpus of conds. I then augment it corpus w. some conds that are "similar" to some of the hy G conds. If any of these have hy G, then this region of cond space gets more pc. ("G" = empirical G; pc = pc assigned by stoch Grammar) ← So I make even more trials in that region of cond space. So I may end up w. [redacted] 2 very large no. of very similar conds, that have all about the same hy G. —— looks like a local Max! A Big Trouble, is that the more trials I make in that region of hy G, the hy pc I put in that region — So dramatically, I may get "locked in"!

Even if I did the more general SGA: i.e. simply trying to make model of G as a function of the params of a cond, I'll run into a similar problem. The Q is,

.20 [How much more about the peak do I want, in my initial Conds?] These are "rule oriented trials" not "G oriented trials"

Perhaps this: Make more trials in region of (cond space) in which the parity below, is predicted G is (true). — But: To what extent .20 is like "what Temperature to Use" in "Simulated Annealing."

.24 One feature I (may) want in SGA: If several regions of hy G in cond space are found, TM2 should find a way to Generalize their locality of these hy G occurs, so it is able to suggest more regions of hy G.

So this → is the SGA Problem: [redacted] ① How does it map onto MCTM? ② How Does MCTM Deal w. it?

.31 Perhaps all one needs is an "Adequate" OZ soln. at the lowest level. This "OZ soln" would find better solns to TM2's problems & get better solns. to OZ probs. We have to be sure the search space is complete — Certainly Turing complete, but ideally, closed "under composition."

Do we want to Model the Entire [redacted] Space? → 36.01

A related Q: To what extent is ordinary GP a "universal system"? Well, if the primitive functions set is complete, then all functions are reachable within a "distance" dependency on how "fit" complete the basis is wrt the set of primitives. → 36, 28

→ 39.02

6/4/00 Bolg. IBM : "Blue Gene" : 500 times speed of fastest present machine. 2×10^6 × speed of fastest desktop machine.

Blue Gene may not be currently for General research, but for Cryptography or other Military uses.

BLUE GENE REF Details (some) SPECTRUM July, 2000 @ ff ff 10^{15} flops

Sounds like great breakthrough in Bytes/dollar.

.01 : 85.24 / GP may be distant from most OT's in pop. We investigate how many regions of cand. space in U. If our [region of cand space] search at a local max, we should put more cc into exploration. Perhaps "more search" could be "rate of ↑ of G_{max} in that region." (Mindful of Jurgen's "Lying how to learn" (Model paper))

[120 d. = 10^7 sec. to be sec for Blue Gene.]

The pc as a cand. need not be the [probability of a new trial cand. in that region]. This last could be a funct of prob of ↑ of G_{max} in that region.

[A+IBM Paul Horn Ambuj Goyal]

.08 A May Unpleasant Objection: to ● say we have 2 regions of cand. space. In the first, G_0 is the present max G so far. In the second, G_1 is the best so far. $G_1 << G_0$, hyp. Say our trials in the first region give occasional ↑ in G_{max}, but only around $10^{\pm}$ trials. On the other hand, in the second region, G_{max} has been ↑ every trial, due by "less" a given w.r.t. G_1 — it approaches an asymptote — its pc ↑ by about the same amount each time, because ● each trial is given a by ordering of its G in the Ensemble. So while region 1 would probably be doing quite well, region 2 would seem to be more promising, in terms of rate of ↑ of

.17 the pc assoc. w. its G_{max}.

Possl. ways out of .08 – .17 : (1) Have the G function increase with Utility. (2) Make the original G function be in some sense "smooth". Then, after the first 100 trials, we make a rough map of G to pc based on ordering of the empirical G's. We use more than G → pc map "smooth". W. this new pc, rate of ↑ of pc_{max} becomes more comparable in different regions of cand. space — ● the objection of .08 – .17 would not hold. In (2) our supposed "first 100 pts" might be random so may cover the space more "uniformly". Otherwise, the G function becomes disturbed by the "self complexity" effect of 85.07 — "automatically zero" "X" for the G function

.256 : 85.38 A GA system that would probably work: 85.31ff: We want a system in which TM_2 can optimize all aspects of TM, TM_1. Then, TM_2 works on TM_1's problem. Instead. Various trials for TM_1's "mods operandi" (≡ m.o.) at first use TM_2 makes those mods & ways Darwinian early, simple TM_1 problems. Then, if some mods seem good, they & their mods are tried on a larger, more diff'n corpus. By the "stop sort" TM_2 spends about ½ of its cc on this problem.

State of TM project as of now:

1) My understanding of GA's how to improve it, and ~~not so easy by level~~ — Maybe so: I'm proud by reading more chapters in Kinnear's book, & maybe after of lot of "ironing" — but how ~~Eq~~ Eq

2) Z141 seems OK. (w. parsing bottom-up, etc) Also, maybe Rao "no SC" approach using loop nests. But the extension to CFG's is not in such good shape — (Rao & NGM Bye ideas may work well). I'M not sure how impl. CFG Bisect is now. I had planned to have TM learn English a diff way (i.e. not by discovering grammars in English text, but by learning Algebra, then learning nuances of English formalization about Algebra). So this is a way may not be central to TM. [I might get some ideas by reading Wolff, but that would be very difficult & very slow]

3) My understanding of MCTM is OK. — The I'm a bit fuzzy on how the probs. are worked & how to info about the solns. gets into the PD.

4) I probably should write (essentially) papers on the impl. Points I've come to understand — so I will have easy access to the ideas in the future. Ideally, the "papers" should have refs to my work notes

5) Perhaps I'm ready to start the Algebra TSQ.

6) Perhaps make list (or more organized form) of the very important ideas in TM, & where they are written up.

7) Perhaps Best Approach to GA: 85.31; 86.28 : A TM_1/TM_2 pair in which TM_2 is ~~quite~~ huge universal (or has a huge universal lang. to describe true solns. (= cands). A TM_2 is one of TM_1's problems.

TSQ is "Enhanced AE:" Miller, Lisa!

It starts out in terms of sort, number of sorts, etc. : I could just start out on this stuff, doing integers at first. — Some how have to teach TM concepts inductively. The concs. taught will not really have to be useful later.

Or don't use textbook! Just teach various impt concepts like, number, sort, Beg, frequency, relative frequency, sum, difference, product, ratio, (add, subtract, mult, div = synonyms). I might even do this all in a subset of English — and also in Math notation. (redundant, synonyms ...).

I could also come up w/ examples, too.

I could try tsq's on Grace, but she would have verbal associations as cues, that TM would not have → [is there a graceful way that TM could acquire associations of this sort?

(if > 50% are bad, was doing pgm for them!)

.15 — Could I use 50% buggy chips? (by 50% I mean that any one pgm has only 50% prob. of being 100% good) — for large searches, maybe yes! Already

• After each search, we test the results on many M procs.

Then we "cut out" M procs that make errors. Soon we will have >99% correct M procs! Even when M procs are highly reliable we will cross-test impt. results w/ 2/3 results of many searches. This can be relatively inexpensive.

One (BIG?) trouble: A Buggy cpu could miss a soln. This means it may get to soln. later. (How much later?)

— We could have a long period of "shakedown" calcs done by all cpu's in "lockstep"! Every 1000 steps we check for identity. Calcs are such that any single error will tend to propagate to all future states.

.27 (if Pr is true, we don't have to check for identity very often!)

Perhaps TBM is counting on .15–.27 themselves to be core bit of the system!

Some Computers are on all the time you access overnite, say. Could I run time on them in a useful way?

Re: T. TSQ: Work out as many details as poss. to try to pursue poss. diffs to Prob a bit.

.35 → If we use $T \leftarrow 2T$ (such), we can randomize choices of cpu's each time around. Say we have 60 cpu's numbered (0 to 59 if 0 is eliminated) we assign each one to $\frac{1}{32}$ of the search space. They don't all finish at same time, so as soon as one finishes (w.o. success) we assign it to the next section of space that hasn't been (searched-through), so each cpu is assigned by

essentially random, sector to search each time.

95.28 On "universality": My very simple functional Lang. "NL" may be able to express all prim rec. funcs, but it is probably not "universal" in the sense that say functionals. "UMC" seems to be. It is more like the universality of AND, OR, NOT gates.

But, NL is able to express any prim rec. func. w. similar # of symbols as any PR prog. Also, (Maybe) if there is another formalism that can express all prim rec. funcs, then NL can express any of them, w. at most a constant longer term than F_i can.

→ This is perhaps true, even tho NL doesn't have a facility to define functions! (?). "(cc) universality" is important also! Since I want TM to be able to improve its cc as well as pc!

— I had this idea of a cc universal machine, in which a pgm started out w. a hardware defn. of the machine, followed by the pgm. At one level, it needs an interpreter to interpret the pgm, with "state of the present art" HW. An engineer using this idea has a pgm to do physics & HW research to improve H.W., then constructs the machine, runs the pgm. For such machines, the time to run a pgm is always the time of the optimum machine + the CONSTANT time to do the research & construct the machine. I think this may be a Less EI view of the $\frac{cc}{\text{usual time}} \to \infty$ in each finite, time.

LG — But we can't assume "Moore's law" ☺. → 93.05

Another random pt: In **koza's ADF's**: When the ADP part of a cand is modified, the RP (reducing) Branch (or as it) of that cand, refers to the modified ADF. If the ADP is grossly modified, it would seem unlikely that the cand would work better (or even nearly as good as) before. — But small modifications would seem like reasonable things to try.

Also, when a chunk of a cand (Branch) is crossed over to another cand, it carries refs to its old ADF's! but in the new cand, it refers to the ADF's in the new cand! That this should work, seems very unlikely! In many cases, the ADF will have the wrong no. of args!

30 A Generalized "Mutation/crossover" scheme that would seem much better: We have cand defns, ∝. We then try cands in order of distance from ∝. Distance from ∝ to β is $\pm \frac{pc(\alpha,\beta)}{pc(\alpha)}$. Furthermore, this "distance function" should be a function of the original problem & the region of cand space we are in, i.e. dist from ∝ to β & the condl. pc of β, given ∝, but there are also other conditions — like the original problem & "how things look in this region of cand space". One of the things MCTM learns, is to improve this condl. pd., as MCTM "Matures" ☺.

So: How does mutation/crossover in normal GA/GP achieve (≈30) #? Well, mutation assumes a fairly simple, ≈ constant (i.e. non-long) distance function. Crossover would seem to have more learning in its ie. the kinds of "Mutations"

Xor: $x \cdot \bar{y}$ or $\bar{x} \cdot y$. If $(x\ y\ z)$ If = $x \cdot y$ or $\bar{x} \cdot z$

Would improve vs. t. work on t. problem preceded:

89.30 Considers (TM universal) methods of derog. modifns of α. Since we have a deriv of α (not nearly a demon V.g. one, but perhaps usually not bad, (if we manage not to have "intens"!), this may not be so diff'l.

Hvr. Koza (& prob'ly almost all GP workers) considers only a small set of functions — usually not "Turing complete". RootBias (X) BranchBias crossover scheme is one way to pack a kind of distance funct. for conds: T. cond/ created has pc = (pc Root)•(pc branch).

Another way to improve t. distance function would be to define subtrees that occurr often. — And "often" is as much as 1 time, if the subtree is large enuf and we do OSL. (I suspect t. subtrees doesn't have to be very large, for this to work.) (85.07)+ discusses possibility w. Hy frag. mod'fns c/o hy frag. modifiers. "self conforming hypoth-ists"

.14 One thing we want: TMs should look at "modifn operators" that have been successful in either mapping α to a cond of $\geq$ G or better yet, .16 or t. mapping is of about same G on average as α, but is > by 6! .29

So main approaches that are promising:

(1) Straight MCTM. (2) GP w. ideas of 89.30–90.16 (A Koza approach) better method of Mutation/crossover. (3) Emphasis, in both (1) & (2) of TMs's > improving t. system & having a complete (Turing? Rice?) set of instrs. to do this.

Looks very good!

- I want to read that chapter in Reinmear, about ↑ α of G. (Altenberg, pp 47 ff) Also 5.18; 85.9.4
- Koza's system sounds very much like Sew. Animal in a sense that has starts w. a bunch of randomly chosen pgms in Cand. Space & continues from there. If he used mutation only, t. correspondence to us would be very strong, but crossover would seem to make it distant, & potentially, much stronger. — T. idea being that we have a larger size for organizing t. "Modifn operators" .29 of .14–.16

Def: In evaluating mcops (mcops = Mutation Crossover ops): maybe $\frac{\Delta G_{max}}{\text{no. of mutations}}$, This is no. of trials to get that ΔG_{max} (?). After a large ΔG_{max}, we tend to move to a new "Center of Exploration". — and t. successful mcop is called told about the set of all other "centers of exploration". This would seem to make crossover much superior to pure mutation! We could start out w. pure mutation, but as t. evolution process got smarter & smarter, it would give more & more weight to t. mcops that had been historically developed & found to be v.g. a cross many "centers of exploration"

Q: Boolean net v.s. TMess: To realize certain functions, a Boolean net must be very large. Turing can realize such functions w. short d cons, but they exchange d cons for ↑ space for Tape. So a Boolean net can be very fast, but will be very large. A Turing could do same thing w. much less hardware, but will usually take much longer. Consider a Boolean function of a certain "depth". I think it can always be expressed as a 2 or 3 deep boolean function of (most "w.r.t." (many gates).

Consider many (n binary input) boolean funcs. There are 2^n inputs 2^n ways of mapping the 2^n inputs into 0,1. If we have K input (OR AND) gates that we could do in 1 layer, then we could realize any function in 2 or 3 layers, but w. very many (2^n) parallel AND & OR gates.

What is expensive is num. of H.W. K time.

Not quite; the computer has long term storage

A Boolean net, has temporary storage.

Thus a F.B. boolean net can store in loops!

A digital computer is usually a not-often kind of "status"

We could have Boolean nets w. F.B., that could have "internal functions" that could be used by different parts of system. This would save "space", possibly lose time, because small use of the "states" by various parts of sys. pgm, over in part. → So Any (synchronous dig.tal) computer can be directly simulated by a synchronous Boolean net w. F.B. — The storage function is obtainable w. a simple F.B. loop!

X or fed back to 1 input: If other input is low, memory is retained. If other input is hi (for 1 clock, say) memory is reversed.

Another xor gate can be used to control state of ↑ derived new state, i.e. decide whether to reverse or not. So 2 xor's will store 1 bit in useful way. Probably a J-K flip-flop will store 1 bit in nice way.

So, if Koza's system can evolve create any network, it can create any synch Boolean Net & ∴ any dig.tal computer — which means it can also have states!

OR: Just have arbitrary Boolean functions — maybe w.i.a F.B., but w. memory

(SIN) R.E. Rev recurrent ANN — what's input, what's output — could Reasser 1-20-00 to 1-25-00 simulate all Prim. rec. funcs ≥ → see TM: New Approaches

I think I examined recurrent ANN's in another note: In particular, ↑ Qof I/O.

If I want to GA a bee to simulate any computer, I could use either the NL formalism (+ RAM) or have F.B. recursion. In either case, I'll probably want constraints on ∞ choices, to ↑ likely hood of ≥ universality of function.

→ 73, or on "Universality"

8/8/00 Bldg. | Sleep/make coarse vs. insane. mind. Self/ego. awareness of "outside world" (can be acquired by wavers) "Large Problems".

"can cross egg"

NB For a GA/MCT: Redefine G as a cond pc: abandon pc(envt, x, y)/pc(envt, x) = pc_envt,x(y).

on [redacted] $PC_{envt,[redacted]}(y, G_y) := \frac{pc(envt, x, [x_y; G_y], y, G_y)}{pc(envt, x, G_x[x_y; G_y])}$ (108.16) Needs WORK!

.04 can be evaluated by random input to a UMC (which is kind of MC-Carlo technique) We try to find y of high expected G values. This is done via evaluating G(y) using G.proc. Naively (which is very expensive: As in Koza's human use of SPICE prim or solve of a DE to evaluate G) — so we claim. This exchange, replacing it w. another (Mighty cheaper!). → on cond pc →

.10 The pc(G) one of, can be evaluated w. G(x) as a "Black box" or Box w. varying suite of "Transparency". (Mighty Box) — fast kitsune evaluation (A "clouded Crystal" Box). (Knowing Methods of Guts, can make chooser Go approximate it) →

.04 eq. .04 may give a much better idea of what's going on in GA/MCT — HW, I do have to refine to eq. — make clear assumptions, types, beds, etc.

((What I want from ALP pc. of new: I want a cond dens, y, ⇒

.23 $\sum_{G(y)>G_0}$ pc(envt, y, G(y)), is as large as possl. [G_0 is present max (evolved) Guts.] "envt" includes any data we want to include in this Approach.

ie. ∃ y that has most probly of having a G ≥ G_0.

.23 can be Rec'd in Trmc terms ⇒ ∃ pc of all codes for (envt, y), having G(y) > G_0.

Now, in fact, we rarely eat pc using ALP directly: We usually use Models like Stochastic Grammars, or linear/nonlinear regression models, act. (Assembly Kitsune approach).

Suppose we use a CFG (stochastic), in .23.

SN I think the G→pc function I described in 67.25 ff is way wrong!

.28 This just puts (ranks in G order, & uses $(1 - \frac{\text{confusing no. 2's}}{\text{no. of cards}}) = pc_2$ ⇒ No! The pc's are btw 0 & 1, but they are not normalized to all 1!

My Procedure to use: I did use .28 first two times, but the f. pc's were not normalized

.32i → Hmm: Some criticisms of 67.25: If we have a bunch of non N cards, & we find a new card w. G ≥ max of the N cards, the old pack pc was $\frac{N}{\sum_1^N i} = \frac{N \cdot 2}{N(N+1)} = \frac{2}{N+1}$! If we find a new pack that's >> 1-fold up, ⇒ pc will be only $\frac{2}{N+2}$ — i.e. worse than f. previous overall pack pc! It will be only slightly better than the present pc of the previous pack: by which is even: $\frac{N}{(N+1)} \frac{2}{(N+2)}$ v. 3. $\frac{2}{N+2}$

T. only Thus above 67.25 just preserves order btw G & pc: But any normalized monotonic function of G will do this as well! So it may be that 67.25 does essentially nothing!

Hyp., pc's $\frac{c_i}{\sigma_i}$ is very bad if we do $c_i \to c_i + K$ w. large K. It ends up w. same Gr ordering, but all pc's are about same: 67.25 ends up w. smallest p.c. having $\frac{2}{N(N+1)}$ & largest having $\frac{2}{N(N+1)} N$: A factor of N difference.

89.12

OT: 91.01 # On Universality: Say we have a set of primitives that are "Turing complete" so that any function can be arrayed as a tree of these functions. If we can do all of these primitives have equal prior, we get an a priori on all of functionspace.

I have some Q about $\sum p_i < 1$ in Prefix groups. ---- I think we know a function ends when it has all of its inputs taken care of as dummy variables ....

.10 Now, we define subtrees ($\equiv$ functions) and put them into dictionary w. different pc's. We still have a prior on the space of all functions, but it is, presumably, crucial to have pc for Turing.

Since any function is definable by the system and can be given by, say, a program to code to any computer def. by suitably defined functions.

.84 So: Why make more complex languages when the Prog. Size (which has definition of subtrees as the early regularity discovery method), is, in the sense .10 — .14 adequate?

Only that more complex langs may be able to detect regularities "faster" (i.e. less seq & less cc).

Would not the System of .10 - .14 be "adequate" for a system w. TM$_2$. Might be able to devise more complex languages and assoc. method for Grammar discovery? Hrr., I do want to be able to put as much heuristic info into the original TM design as possible, so it will take minimal bits ($\equiv$ TM Seq.) to "get off the ground".

Another Impt. Point: The set of funcs should be able to be recursive — so full Turing Complete. Lisp is, but it may be somewhat awkward.

PROBABLY NOT good idea to design (suppose one succ. — Just run TSP "in English", giving [redacted] rules to be observed by TM. From the needed rules, I can construct a "Language".

I guess that I feel that I can solve induction problems o.k. (diff. Pd. is crucial one for me.) & similarly, I can solve time probs. (diff. Pd. is crucial one for Y. also. What I'm most uncertain about, is OZ problems. The MCT is supposed to solve them, & 92.23 is about how it's done.

We want a function that maps from the OZ prob dom. to the marked of solving OZ problems. It sounds like a "TM$_2$" type problem. For time problems, TM$_2$ looks at the prob. dom. & produces a Pd. to be used.

~~For OZ probs~~ " " " " " " " " a O.T. (optim. technique)

Another, More vacuous "impt idea": That time probs are usually solved by XFing them into OZ probs — the solving is guided by the (closeness) fitness function.

6/12/00 Bulg.

T. approach of 92.23 is ~~e~~ e-l. (Tho it does use only order of G).

T. Strat 02 soln. is Non-El: I think it automatically included tech. improv. for cc (e.g. Quicksort)

Hyr. often Non El soln. are too diff. at first & a good elem. is preferable to start off w.!

SN ISQ: doors 1, 2, 3, 4 ··· = 1, 2, 3, 4, 5 ··· ?

1, 2, 3, 4 ··· = 1, 2, 3, 4, n ··· for any n ! ?

No! naturally 1,2,3,4 ··· uses minimal complexity; 1, 2, 3, 4, 6 uses much higher min complexity combin.

1, 2, 3, 4, 1234 for 1234 4224 1234 ··· 1234, 12345, 123456 ···

1234, 123, 12, 1

1234, 6789 11 12 13 16 17 18 19 21 ·····

Goal. Discr. of Path to TM! Most problems are 02 problems.

Strat (in most recent calm moment I know) used w/ set of OT's: GP is about as basic OT as I know & it will get max wt. & so I'd like to know how to do it GP "very well".

I also have to work on problem of how G P.D. is modified by TM's (T. MCT problem) — be sure & know exactly how to do this!

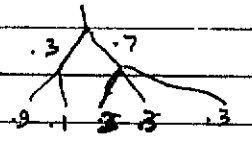
92.23 is perhaps a step in right direction for 02 probs. (& perhaps GP).

Best immediate approach ① Read Alterberg ② develops 92.23 first, for ~~some~~ cases of one G value. ← "Study Program" Store new evts, & all previous in RO entries.

③ Expand to cases w. > 1 G value.

.23 SN On selecting cands of max (exp) pc from z ← ; track CPG! If one picks e. most likely choice each time in the Generation Tree, Gace will often do well.

Occasionally, Gace will have to "look ahead" & be (less greedy): "Look ahead Depth" means one looks at branches at depth k from where Gace is, to determine next decision. k = 2 is minimally non-Greedy way;

.28 

for k = 1, backpushes .7 x .3 = .21

k = 2 " " " .3 x .9 = .27.

one could go thru (initial tree) & label each node w. product of its pc & the highest pc it branches to.

We can then recursively repeat this process until k nodes down (relative to where Gace is). Then would give new pc for ~~the~~ this branch.

.35 Another way: starting at the leaves, go 1 node up and propagate pc w. (i.e. x max local pc). To go up 1 level is tedious. Continue to top level ← .35 if taking long time to get back pc, but .25 if willing to get poorer approximation at less cc.

(This amounts to MCC, to accurately all branchings, use 1 + the ~~branching~~ factor; No branch ··· score 95:01)

(Another way to get truncation)

→ 95:01

.01 94.60: A very Good trick! Using fairly greedy methods, find some complete parses of the fairly good by total pc. Then go back to next but don't bother examining branches that have total p.c. < the ones one has found. Then can be made approximately by only considering depths k from the GOP as a "complete trace" & finding best path" & not examining paths w. ppc < max found

.07 On another approximating a G function of a string by a pc:

The way it's done: Say $\vec{x}$ is a document of a cond. & $G(x)$ is the observed G of that cond. Then, if we want a stoch lang ($\equiv$ pc) $pc(x) \tilde{\propto} G(x)$, the mean of pc ($G_c$) is the true pc of that x. Then for large $\leq\leq z$, we will have about $k\,G(x)$ occurrences of x in the corpus. ($k \propto \leq\leq z$). So, when x_i has $G_i = G(x_i)$, we just use k_i w. a $\leq\leq z$ of $k\,G_i$ as data to construct our stoch lang.

Here for a finite sample of $\{x_i\}$, there always exists a k so large, that the ad hoc grammar (i.e. a listing of the x_i) is the best grammar.

The grammar would be $\xrightarrow{\theta_i} x_i$ [$\theta_i \equiv \theta_i / \sum \theta_i \equiv$ normed pc].

Now k to be finite, perhaps small.

Say p_c is the pc of a stoch grammar. Then for $\leq\leq z = k$, the pc of the corp becomes $p_c \cdot \prod g_i^{k\theta_i}$ g_i is pc assigned to x_i by Grammar.

.20 $= p_c \cdot \left(\prod g_i^{\theta_i}\right)^k \rightarrow \ln p_c + k \sum \theta_i \ln g_i$

$\sum \theta_i \ln g_i$ "$\tilde{\propto}$" $\sum (g_i - \theta_i)^2$ ("$\tilde{\propto}$" is actually "$<$ but $\sim$")

There is some reason to feel $k=1$ would be reasonable; k can be regarded as the wt of $\ln p_c$. A trouble w. the "error function" $\sum \theta_i \ln g_i$ is that when $\theta_i = g_i$, the error is not zero! It is minimal, however. ([Rank!]) for all.

$\frac{\partial}{\partial g_i}\left(\sum \theta_i \ln g_i - \lambda \sum g_i\right) = \frac{\theta_i}{g_i} - \lambda = 0 \therefore \boxed{g_i = \theta_i}$ for maximum.

.27 So the $\sum \theta_i \ln g_i$ error criterion need not be $\propto \sum (\theta_i - g_i)^2$ — all are convex & it is min when $\vec{\theta} = \vec{g}$. Other things, however, another possibility that k or n, the no. of cands.

Actually, what we want to know is, how good the g_i are in determining the θ_i. — So for this reason, we'd like the error (measure criterion) to be zero (no correction needed) when $\vec{\theta} = \vec{g}$.

Using $\sum g_i \ln \frac{g_i}{g_i}$ or $\sum \theta_i \ln \frac{g_i}{\theta_i}$ would have zero error when $\vec{g} = \vec{\theta}$. $\sum \theta_i \ln \frac{\theta_i}{g_i}$ looks better, because θ_i corresponds to the "true pc" & g_i is the approx.

.36 $\sum \theta_i \ln \frac{\theta_i}{g_i}$ (is used in Sol 78) > 0 $\therefore \sum (g_i - \theta_i)^2 > 0$

However We really want to compare the pc of the dcrsn of $\{\theta_i\}$ using $\{g_i\}$ as approx. Is .36 a measure of how much info we have gained than $\{\rho_i\}$ to get $\{\theta_i\}$? $\sum (g_i - \theta_i)^2$ is a measurable measure

6/13/00 Bulg.

G_i' is Normed true G

β_i is output of approx. Grammar

of the amt. of info we need to correct β_i, if the error is Gaussian, is perhaps 95.36 is close enuf to $\sum (G_i' - \beta_i')^2$. Hmm, Both $\sum G_i']$ & $\sum \beta_i']$ are normed to 1, so they wouldn't differ by a Gaussian D.F. unless they were very close.

I.e. the info. distribution is 1, "constant", independent β_i.

.05 Ea $\sum G_i' (\ln G_i' - \ln \beta_i')$ we expand to $\sum G_i' \ln G_i' - \sum G_i' \ln \beta_i'$

$\sum G_i' \ln \beta_i' = 95.27$.] so, if .05 is the info needed for error correction.

we do want to maximize

and we want to maximize $\ln P_\alpha + \sum G_i' \ln \beta_i'$. which is 95.32 w. k=1 !

Look at amt of info for error correction; say it's $\chi = \sum (G_i' - \beta_i')^2$

So, O.K. we do want to Min χ $\sum (G_i' - \beta_i')^2$, but how much to weight it with the rest of Grammar decision, i.e. $k - \ln P_\alpha$. Sounds ~ to linear regression: decide how many coins!

Unique Parsing Grammars:

Nothing Gödel or a well-known example.

Every impt problem is a "solve conforming hypoth" problem in disguise. Examples from many areas.

One way to look at it is a N model, possibly a n-dimensional quadratic shell with negatives. We decide how large a hypersphere to distribute over best data pts.

— Actually $\ln P_\alpha + \sum (G_i' - \beta_i')^2$ is actually an accurate curve fit to data. All we are doing is non-linear regression. (closer to curvefitting: but data is onordered). The analysis of H.M.C. (How Many Coins) is probably very relevant.

It is n dimensional n-linear Curve fitting. If we assume we local linearity & that we are close to a soln, then n dim curvefitting is relevant.

Another approach. If we assume only 1 parse, & that Str. Grammars will each be characterized by a set of pc's. Each β_i will be the product of a certain subset of pc's. Certain subsets of these probs (in the model) will have to sum to 1. No other constraints. The product = β_i suggests using logs but use Lagrange Mults doing the "sum=1" constraint.

.27 $\frac{\partial}{\partial p_i} \left(\left(\sum (\ln p_i - \ln \beta_i')^2 \text{ constraints} - \sum \lambda_k (p_{ik}) \right) = 0 \right)$ → 98.01

We can minimize this (w. constraints) or minimize $\sum (\beta_i' - \pi p_i)^2$ without constraints, w. constraints when we feed in all the models, we get the pc by interpolating the formula over all values of the prob params. This is easy to do because probs usually on (0,1). (maybe sometimes trouble around $p_i = 0$ (between or $p_i = 1$, but none at $p_i = 0$. !

SN in non-linear regression with local linearity,

I. $\left[\text{linear model of references is } \sum_j \left(\sum_{i=1}^{k \text{ dim}} a_i C_i x_{ij} - y_j \right)^2 = \min \right.$.

It may be that if one knows G_i' accurately then one integrates over a smaller space to estimate Y_i, so these 2 effects always cancel / — which seems correct. Since the final reason/ would depends only on the "no of params" in P_α is would be (I should think) a "soft" constraint rather than a discrete one. (analog)

→ 98.04

If a certain structure is possibly correlated w. hy G, good! Hyp. a certain str. could occur in both hy & low G cands, & be essential for hy G. A str. could occur in both hy & low G cands & be unnecessary for each.)

So correlation is inconclusive. Lo. correlation, tells nothing!

I'm not so sure!

On ~~Some~~ Comparison of "A.I." to humans:

Deep Blue ~ ~~7 x 10^9~~ ops/sec (Lewchuk's rate) gave ~ human performance.

Koza's 64 x 200MHz = ~~5 x 10^9~~ ops. 2 days x 2 hrs = 100 hrs.

Compared to 1 year's work by human zoologists. So ~ twice as fast as Human

So Koza is ~~1000 x as good as Human~~ ~ Deep Blue ~~as Deep Blue~~

Deep Blue used very little A.I., mainly search speed.

Koza used much A.I., but perhaps ~~some~~ inefficient search: 2 areas of much improvement obvious: (1) Faster ~~hardware~~ approx simulation of "SPICE"

(2) Learning ~~procedure~~ transfer of info from problem to problem.

Some GROSS SPICE simulation improvements:

(1) Quick Abort! to realize cand ~~is~~ is meaningless. Output is always zero, a/o is indep of input. (2) Energy version of Amplifier: Restriction D.C. input

Very simple models of str.

Early Space ~~vehicle~~ models would ~~have~~ try to get max no. of rejections

& cands for cands in minimal CC.

Same Modelsat SPICE could be used on Mostow (linear DC's)

" linear ~~Kalman~~ Filters,

In "Book III" Koza was less "hyper" about his pgms being comparable to humans, Peter had me in earlier papers: Why was this?

.26 I may be able to improve Koza's General search schema a lot, or maybe not: It may be that any apparent improvements in search strategy amount to a kind of "Criterion" & usually have effect of discarding "Rare" trials that are potentially very "Creative".

Anyway, via .10 ~~2.5 x 10^9~~ 7 Mflops is ~ human speed: — which is ~ (3) times speed of present-day desk top machine: Say $1K. So $3K would buy human capacity!

Nvr., a - Comparison of Koza's Pgm. w. Human Performance is a poor comparison. In his "1 yr" a human would not be constantly working 8 hrs/day on a problem. On a PC Koza's pgm could be much improved via (.15 ; .16) at least & possibly .26.

In Koza's 3rd Book in his paper in ~~Adv.~~ AI: Go Journ: , he lists ~30 cases of Human Competitive work by GP's. It might be possible to get info on ~~IPC~~ of ~~these~~ PGMs; But comparing to Human time spent, would be diffic. ~ 110.18

Re: Deep Blue in 1997 (what a breakthrough)

~ 200 M positions examined

3 min. 6 M positions in 1 sec.

I wonder know speed of "deep Blue"?

1 sec clock/position on a PC. ~ 7 x 10^9 clocks/sec

6/14/00 Bulp

To show that if they domain sizes not proportional then Lattices Plans could overlap.

Same length & distant: they could overlap.

96.27 spec

101 : 96.40 : $\Sigma (\ln g_j - \ln g_i)^2$ may not be so good. Would much more interested in good fits for larger g_i's.

Note: 96.27 is what we got if we used error criterion $\Sigma (\ln G_i - \ln g_i)^2$

(I guess: What I'm complaining about now, is

However, maybe it would be good for Model to be able to just get good fits for low G 's walks by! — it would get rid of a lot of "undercounts" — which is perhaps as important as predicting by G values!

Anyway, once I get in general region of Best Fit (After having chosen discrete params), I can use method of 96.14-.55 (possibly 2 & dim quadratic form Newton deals) to get true & precise peak.

It may be possible to solve 96.27 — exactly. Also, it may be that I only have to solve for low g_i values in the grammar! (maybe only 1!) — For the most part, I will only need to know the changes in pc. assoc. w/ certain small changes in grammar.

How it will often be for a SSZ ≈ 2 (for OSL).

Dad MSL .17 .18

So, I want to get the d.f. for g_i: Making sure that I take OSL into Account

Consider 2 cases: ① "Regular" induction w/o OSL .30 ← MSL (Multishot Imp.) ≠ OSL

② Pure OSL. Pure OSL

SN On implementing TM; TM>1's problem is very difficult but it can be viewed as "slowly". One way to do this would be use of "spare time" on Home Computers. Thought Co. that "Entropia" Arrays this: Said that cost of doing this is ~1% of cost of Supercomputers: "Computer" June 2000 : p16:

It might be possible to get much IPC illegally by "Hacking" into HomePC's & perhaps even Supercomputers! .25

① For simplicity, he used my 2×423's: use : for TM1, for TM2

Mobility Electronics Inc: "plx Bridge to Chronographic Sensor"

"Slowly" means that BW from computer doing work, to place where results are used is very small

Computers share common PCI BUS — can allocate a common PCI card.

Q: Is the SETI prg simply looking for regularities in data streams? — use recursion (Rabin)

[6/15/00] Ah ≠ "regular" induction! Say we have this context ends & we express it as a subset of cart. product of Roots & Branches.

Think Perhaps for OSL!

Trouble is, Sequitur had n cands, it adds 2 nodes/ branch/ pair. So ~ n nodes, n branches.

2nk-2 pcs (params) to adjust, but only n data pts (equations)

Perhaps for only branches w. cScnt ≥ 2 would be of interest. (This is not OSL).

— Consider the sub branches of a branch: — do they have any "meaning" relative to their "mother branch"

Well, just consider a corpus w. all same G. Then factoring into roots & branches at .30 would not be reasonable. — I might be useful for OSL, perhaps. .39

With constant G, we simply want to write a good stochastic grammar for corpus.

Say we have the grammar of 98.[illegible].39. It will have various "p." params. If it is a PCFG (stochastic). If these params are functions of G, then we could have one form of stochastic grammar of the type we want.

.00 OK. Say we optimize the [illegible] for the 39 G. cands. Then we optimize a few p's for a few G sets of cands: [illegible]. So for each p. we have a 39 G v. a few G values, and we do linear interpolation for other G values. If there are 4 choices of model, we have to give (4-1)·2 params (& accuracy

.09 of harmony).

This → model is much different (& potentially more accurate) from

.11 the previous model, in which the "Grammar" keys to approximate G of each cand (w.o. giving a p.d. or measure of certainty of p./G values). → Same 100.01 for design of theorem

.13 — presumably with a smaller error for the known corpus & comparable.

The .11–.13 model is more like fitting a grammar (non-stochastic) in which one uses negative cases! Say all G > G₁ are positive cases, all G < G₀ are negative cases. → 100.01

The "Roots & Branches Cards product" of 98.17ff is a Pink, pretty much OSC for MSC, I need known Cards of Sub Grades. Cst is Cst > 1

→ So, using, say, 39 G sub corpus, we derive a grammar — (then we may want

→ to generate some more "critical cards" to improve G grammar ← like the w.e "teacher"!

Then we try to optimize the params to get better prodn. of G.

— including rejection of many low G cards.

(N.B. | Certain structural features/params of the model may not be able, (on the average) to distinguish better 39 v low G cards, but certain various other params/features may give a 39 Var, so we get a fair no of 39 G cards! | So what we really want is a model that gets a high ex fraction of 39 G cards. Such a model would be a stoch grammar

How we could construct such a grammar, use various values, generate some cards get more true G's; Vary params, Generate more cards, get more G's, etc: Then pass a 4 param quadratic from the 4 data & get a new maximal best point.

A — apparently distant approach may find one useful in (1) input Muxer — Generalized by "conditional nearness" search. Perhaps look at this in some detail.

Make list of approaches to problem (1) "S G-L" (2) OK (1) input Muxer in conditional nearness (3) Straight OZ/DNV Lorch (4) Maybe look at Hubbard & Alternberg

.01 99.16 There are 2 kinds of models of "stochastic grammar" we're trying to distinguish!

99.04–.09 get Pc as a function of the conditions, i.e.

.03 99.11–.13 gives a "best bet" [illegible] of G, for a conditions; Then the entropy

Model has a certain σ_G^2 of expected error in the prediction. This gives a dist.

for G, just like Pcon 99.04–.09 Models.

Both kinds of Models are necessarily common. 99-11-13 is common, however.

For the "Best bet" Model of .03 (99.11–.13): The corpus of 96.27

depends on the interpretation: A more tractable error criterion is

$\sum G_i' (\ln G_i' - \ln g_i)$ (w. Lagrange Mults). The $\sum G_i' \ln G_i'$ is

a constant & is irrelevant: $\sum G_i' \ln g_i$ is certainly familiar! ☺

It looks like it will be very simple mathematically! g_i will be of total no. of

times it occurs in the Corpus for the entire Corpus. Not not quite!

.13 Say $\prod_{j=1}^{k_i} P_{ij} = g_i$ Here we assume only one best parsing for Cond$_i$.

(not "w" ?)

.15 So $\sum (G_i \ln g_i$ terms $+ \lambda_i$ (constraints) : PCG constraints are normalization constraints

on the P_{ij}'s.

Every time P_{ij} occurs in the product of .13, we have one normalization

Constraint of the form $P_{ij} + P_{i,j+1} + \dots + P_{i,j+n} = 1$. (commonly $P_{ij} + P_{i,j+1}$)

so taking $\frac{\partial(\text{LS})}{\partial P_{ij}}$, we get $\sum_i \left(\frac{G_i}{P_{ij}} + \lambda_i n_{ij} \right) = 0$ n_{ij} is the no of times the

Constraint on P_{ij}

.21 So $\frac{\sum G_i'}{P_{ij}} + \lambda \sum \dots = 0$ occurs in the term of Cond$_i$

→ I really have to work out an example to get this Algebra straight! ←

& Which terms are n_{ij} λ's in the [illegible] eqns.

.25 involving Cond$_i$. So $\frac{\sum G_i'}{P_{ij}} + \sum_{j=1}^{n_{ij}} \lambda_{ij} = 0$.

So we solve each such eqn. for $\frac{\sum G_i'}{P_{ij}}$; A bunch of linear

eqns. w. the additional linear eqns, that $\sum_{i=1}^{n_{ij}} P_{ij} = 1$. (Normalization Bds).

Best by analysing some specific cases: Say we have Rules A, B, C.

We have various kinds of data that have been assigned. G varies in the corpus

We could use the previous Lists of 2-[4] to find subsets (new operations)

w.o. considering G values. Then we can have a dictionary in A, B, C & Pr.

"tree-in-BG" [illegible]: Which one or interpretation of BG. If BC dictionary

(Pr in is some distance from 2-[4] in that the corpus is ambiguous

Conds ---- but disregard this time which).

In the early part of the search, the Conds will be random combinations of

A, B, C, So we should find many "intermediate" subtrees (called 1) – 10201

[redacted] TM (H.W)

6/16/00

.01 : 98.25 MOBility Electronics Inc: Split Bridge Technology.

H/W/ mobilityelectronics.com/ index2.htm Goo Bunches files on Pro

.02 in F:\SS\(6-16-00) : Split bridge can be used in Desktops. (Using Simple Split Bridge extension slots in the form of an external Box) ?

The it may be adequate to connect 2 computers via hub link or other faster (1 cable transfer), Since updating from TM1 & TM2 is infrequent.

It may be possible to connect PCI buses of 2 computers together; also, use common clock (w. variable delay to adjust "skew".

The USB is rather fast (it will be in speed in future). I may want to use this to connect 2 computers: One of the papers I +

.02 discusses Briffys speed of USB.

This charge #200 for Naive Split Bridge Docking station for laptop.

They say 1.25 Gb/s so ~156 MB/sec.

Also, I could use standard ethernet stuff (but I don't know how.)

I do have to know Pci Cut: It works w. W95 & also LINUX.

6/16/00a Bulg.

cst case-with count

= c_{ij} ← cst's after Pe paralyze

.01 Later, subtrees of unusually big freq. will appear.

Anyway, w. our "Augmented dictionary" à case counts, we can produce & ranks, & get a second approx to case counts for augmented dict'n. From these, we can perhaps try to find p_j's to approximate these G_i's.

Using equs like 100.15 - .21

In generating the codes, say I have choices of any dictionary entry, at each point. There are k dict entrys.

I want $\sum_i$ (all codes) $G_i' \ln p_i$ (each code) to be min, w. constraint $\sum p_i = 1$.

($\sum a_i' = 1$ by definition)

$\ln p_i = \sum_j \ln p_j$

The i^{th} code has c_{ij} cases of the j^{th} dictionary: ($j = 1 \ldots k$)

$\sum_i$ — So $\left(\sum_i a_i' \sum_{j=1}^{k} c_{ij} \ln p_j \right)$ $\sum p_j = 1$

.15 So $\frac{\partial}{\partial p_j} \left(\sum_i G_i' \left(\sum_{j=1}^{k} c_{ij} \ln p_j \right) + \lambda \sum p_j \right) = 0$

$\sum_i G_i' \frac{c_{ij}}{p_j} + \lambda = 0 \qquad \frac{\sum G_i' c_{ij}}{p_j} = -\lambda$

.19 So $p_j = \sum_i G_i' c_{ij}$ / normalizing factor (normed $\sum_j \sum_i G_i' c_{ij}$)

So p_j = normed : no. of times p_j occurs in the codes, w.t'd by G_i' each time it occurs. If we wanted p_i to approximate G_i, we'd No! would want the occurrences of p_j by G_i rather than G_i'.

This would seem false, yet, in eq (.1E) if we use $G_i' \to G_i$, we will be trying to get p_i to approximate G_i ! Say $\frac{p_i}{G_i'} = 10$. Then all of the p_i's for G_i will be 10 times as large as the p_j for G_j'.

Since p_i is the product of several p_j's this would indeed work out!

.26 $\prod_i \left(\prod_j p_j^{c_{ij}} / G_i' \right) = \min$. Not! would Can't! do constraint $\sum p_i = 1$

.28 $\sum_i \sum_j c_{ij} \ln p_j - \ln G_i' = \min$

$\frac{\partial .28}{\partial p_j} = \left(\sum_i \frac{c_{ij}}{p_j} - \lambda \right) = 0$ Indep of G_i'. In .26 $\prod_i \frac{1}{G_i'}$ is a constant factor

.24 In 100.15 I had $\sum_i c_i' \ln p_i = \min$ (w. constraint $\sum p_i = 1$)

.25 $\sum_i G_i' \sum_j c_{ij} \ln p_j$ which is 102.15

.28 $\sum_i \frac{G_i' c_{ij}}{p_j} + \lambda = 0 \quad \therefore \quad p_j = -\frac{1}{\lambda} \sum_i G_i' c_{ij}$

$g_i' = \prod_j p_j^{c_{ij}}$

$\ln g_i' = \sum_j c_{ij} \ln p_j$

I think this derivation (.24 - .26) was obtained by assuming both G_i' & $\overline{p_i}$ were normed. If we don't make this assumption — I.e. Q is — where does

.01: ↑ derivation breakdown?

Look at $\sum G_i' (\ln p_i - \ln G_i')$: Suppose here $G_i' = \vec{p}_i$ were normed to α. Then expression is result, by α (no.)!

~~Then constraining $\sum p_i = \alpha$ + Hol~~ ↑ constraint is $\sum p_i = \alpha$.

So $\sum p_i$ need not = 1. So it may well be that p_i (if $\alpha \neq 1$) ↑ expectation to be changed to look!

See how well 102.26 works:

Say $G_i' = \frac{i'}{\sum (\text{n chosen})}$: $i' = 1/n$. $\left(G_i' = \frac{i'}{55} = i' \cdot \alpha \quad \left(\alpha = \frac{2}{n(n+1)}\right) \right) \quad \alpha = \frac{1}{55}$

$G_{ij} = i$

$c_{i1} = i$

$c_{i2} = i+1$

$c_{i3} = i+2$

$p_j = \sum G_i' \cdot c_{ij}$ · β ← normal.

$p_1 = \alpha \sum i \cdot i \cdot \beta^i \cdot \beta$

$p_2 = \alpha \sum i (i+1) \beta^i \cdot \beta$

$p_3 = \alpha \sum i (i+2) \beta^i \beta$

Say $n = 10$

$n(n+1)$ normed

$n(n+1)+$

$\sum_1^{10} i^2 = 385$

$\sum_1^{10} i = \frac{10 \cdot 11}{2} = 55$

$p_1 = 385 \cdot \beta$

$p_2 = (385 + 55)\beta$

$p_3 = (385 + 110)\beta$

$\beta = (1320)^{-1}$

385

So $p_1 = p_1 \cdot p_2^2 \cdot p_3^3 = 385 \times 440^2 \times 495^3 / 1320^6$

$G_i' = \frac{1}{55}$

$p_{init} = 1/1320$

$p_1 = 385 * ba$

$p_2 = 440 * bG$

$p_3 = 495 * bE$

Print. $p_1 * p_2 \wedge 2 * p_3 \wedge 3$, $1./55$

$1.7 E-3$ $1.8 E-2$.

pretty far off.

Bolg103.BS For $i = 1$ to 10 : $C(I,1) = i$; $C(I,2) = i+1$, $C(I,3) = I+2$.

$G(I) = I/55$

$p1 = p1 + G(I) * C(I,1)$

$p2 = p2 + G(I) * C(I,2)$

$p3 = p3 + G(I) * C(I,3)$

Next

$N = p1 + p2 + p3$

$p1 = p1/N$: $p2 = p2/N$, $p3 = p3/N$.

For $I = 1$ to 10 :

print $I, G(I)$, $p_1 \wedge C(I,1) * p_2 \wedge C(I,2) * p_3 \wedge C(I,3) = g(i)$

Use Excel

$G(I)$: $g(i)$ ← Same classes for $\Sigma = 1$ (w. Dim factor $= 10$)

For $\Sigma > 1$ to de-pendency increased: $G(I)$ ↑ the linearly, $g(i)$ ↑ exponentially

Also, approximating $G(I)$ by $\bar{G} = G(5\frac{1}{2})$ would be a much better approx from $g(i)$ — for all values of i).

Well, $p_1 \cdot p_2 \cdot p_3$ can't all be close to 1, so if we take products by powers of each of them, this product will have to be very small!

— No matter what $G(\vec{x})$ is.

Maybe g_i should be normed. — This sounds more reasonable, but

In the race car example, if $g_i = \prod_j p_j^{c_{ji}}$, small differences of c_{ji} will cause enormous differences for g_i.

(Ever, this may be OK. We will have to keep the c_{ij} small anyway (low complexity).

c_{ij} may be small, but $\sum_i c_{ij}$ could be appreciable.

If the p_j are normed, Probly G_i shouldn't be normalized.

We only have a small part of the language in our corpus.

We may want to normalize both $\vec{p}_i$ & $\vec{g}$ if we normalize G_i'.

$\sum G_i$ the $c_{ij} \ln p_j + \lambda_1 \sum$

$$\sum g_i = \sum_i \prod_j p_j^{c_{ji}} = 1 \qquad \sum p_j = 1$$

Thus $\sum g_i = 1$ constraint is a drag!

$$\frac{\partial \sum g_i}{\partial p_j} = \sum \frac{g_i c_{ij}}{p_j} \quad \| \quad \sum \frac{G_i c_{ij}}{p_j}$$

$$\sum \frac{c_{ij}(\lambda_2 g_i - G_i)}{p_j} = \lambda_1 \qquad \sum_i \frac{c_{ij}(\lambda_2 g_i - G_i)}{p_j} = \lambda_1 p_j$$

Maybe solve by successive approx?

Is it worth to Norm the p_i? — They are simply parameters. —

But they are normed if we assume the corpus is a stock feed.

Modeled by our stock Grammar!

Do consider $\hat{p}$ not normed! ?

If $\hat{p}$ is not normed, Platts heuristic by which the scalings were determined, is invalid. — Well, it may be any way — since it was independent of $\vec{G}$!

.27 If N is no of cands Pres for, use $N G_i'$ as "ssz" of cand$_i$.

.28 ($\sum N G_i' = N$)

.29 .27–.28 is quite reasonable! If the corpus was generated

.31 → so as to have $g_i = G_i'$, then Pres would be the ssz's observed ← IMPT!

.32 would G_i' N be often ≥ 1! If it were ≥ 1 Pres might

.33 be enough justification to declare Cond$_i$ entirely as a scheme!

So our Q is: If G_i' is rather highly determined, .32 would seem to be false, & G_i' may be low by pts than the data of .33 is perhaps warranted!

Try Pres (.29 ff) now on the 11 input corpus to see if its quantization reasonable.

So: Each Cond$_i$ has a ssz of $N G_i'$.

6/17/00 Bulg.

This is a manual 4/23 40
6/17 105

.01 [Remember that our corpus was not obtained in an "honest" way! It is not a "Random Sample" by any means; (It tends to be highly biased towards by G cond's). → Sok. hight of 10t.29 — .3 is "incorrect"

[Optzn Techniques]

SN1 Re: MCTM: Just how did it get a good P.D. for OT's? Well, maybe it looked at past OT problems; It looked at OT problems, OT(S) used, [part of by G-values obtained w. assoc CC's]

From this, given a new problem & a CCs; It induces a P.d on OT's & their expansions & w. CC = CCs.

How does it get expected G for new, untested OT's, that it is about to invent? More exactly, how does it induce new OT's w. hi early utility"?

SN2 In Cases These combined functions: The functional parts form a tree, but the "side effects" of the fcns (which are often the main thing of (interest)) do not form a tree. The whole computational sequence may be one. form a tree?

SN3 INDUCTION problems: While these are OS probs & can be solved using OS algs, they can also be solved by a direct search, using the current "Best P.D."

Other point re mention of .01, it would seem that this weighting of the data in accord w. G' would be a v.g. idea: As better $\vec{P}$ is normed $\vec{G}$ is not. Here, what it seems to do is, instead of $P_j = \left(\sum_i (G_i')^2 c_{ij}\right)$ / normzn factor, $\left\| \sum_i (G_i' c_{ij}\right.$ / normzn would give same result.

Which is rather weird! because any monotonic xfmn of G should not change results much (?) Hm, in general, changing $\vec{G}$ in any way will affect $\vec{P}$ a lot.

On the other hand we use "pseudo sample size" of $\alpha N G_i$ to hunt for good subtrees. Do we also use it in calculating p_i's as well?

SO: Present Problem: Does the system work? Is it able to detect useful sub-trees &, predict them? / Multiple Shot Lrng.

.31 [Note that this is MSL (not OSL), so it doesn't cover ~ ½ of lrng. Also, the # of many consts is always almost always < that only a few cases — Since p_i's are always < 1. (Unless we allow many passes, parsings.)

.31 suggests that we may have a very poor model of G! — But it is a reasonable model of Nat language!

We might get Conds of reasonable screen by Φ_G, by expressing them as Modifns of other Conds of known by G.

So one Major present problem: ① to get a good model for G as a function of Cand. descr. or ② a fct for cand descr/G pairs. or ③ a stoch operator in which one inputs cand descr & can get fct. of G.. or ④ One inputs G & gets a stoch thing. That "has" that G

More General: General Soln of the OZ problem: or General Hillclimbing.

SN When one gets stuck in G, look at the sources of the ↑ & try to make more modifns of it.

SN Some impl. Components of Prob. Solving Systems: ① Find macros that are useful in a variety of problems, Combine these macros as cands for Solving new (& old) problems. ② Look for obs that guide creation of macros to solve problems.

Apparently Alternative Approach: Just do TSQ's for MCTM. In general, I will get the machine to solve probs the best way I do — & I'll use Lsrch — & Modify the p.d. for Lsrch as TM interfaces.

A trouble is/was that Most impl. probs are OZ problems, and usually Env problems are solved by finding a Suitable G & solving them as OZ problems. Many OZ probs can be solved by devising a local p.d. to do Lsrch or Hillclimbing for "Blurry" trials.

.19 Do such problems really exist?

SN A possible objection to always using Lsrch: Certain problems may have no "G.F.B." for low hcost trials: Yet, while larger hcost trials would have much lower pc for each trial, there are many soln of hy hcost ∴ Σ pc >> 1 So doing Random trials of hy hcost (low pc) does yield solns — Non-random Lsrch for low hcost solns gets nothing. How can one recognize such situations? — If one cannot recognize them, then hcost plummets! → 107.17

(No I'm not so sure of this!)

Anyway, I could just come back for it & do it w. regular (low hcost Lsrch.

Well, it would seem that for certain problems, Only hy hcost trials would have any G at all & that no info would be obtained from low hcost trials, Trying hy hcost trials at random would seem to be a bad approach — but if the standard hcost is not too large, it could be feasible.

.31 I was considering GA because the standard Lsrch Soln of OZ problems involves considering various (usually v.s.) OT's — & GA is a v.s. OT. However, I don't normally use it (I think)

.35 in my own OZ problem solns. I probably do use something

.36 like conditional pc, Lisabeth. But usually, I have plan, & it feels me when I'm doing better (≡ a locally discovered/created "G")

Anyway: The reason I began investigating GA's, was that it was a OZ solver (& all probs are reprocessable as OZ probs), & I think

Another Rmk I like about [redacted] Koza's GP) Since it seems to work have problems (abs. value), I would be able to give it a TSQW w. larger CJS: Such seqs are easier to write than TSQ's w small CJS.

.01 that I'd be able to emulate it w. SGA. by

So far, this has not occurred.

Also, there is a good correspondance betw MCTM & GP systems, so I think that I could work on both & learn/look at problems from both pts. of view.

Even I may be able to emulate GA by 106.35–.36: Search over local P.D. for conditional p.c. (conditions are ① locality in cond space. ② nature of Goal/problem).

(SN) I had this idea of Synergistic Conc A & B, so $G(A,B) > G(A)+G(B)$: This idea of adding G's is somewhat different from the idea of 202.15 — in which I sort of Multiplied G's: (The P_i's were G's of concs). Well, I did add & loss, but — But the logs were always negative, so Σ concs together were always worse than even one conc along!

Because Koza seemed able to work large problems, I that that I could use Koza's system & a TSQW w. [redacted] large CJS. Probably misconception & Search is as fast as one can go! (If one updates P.D. properly, (Timothy)

.17 (106.19–.31): If there are many solns. at hy Kcost, then the pc's of these solns. could sum to >1 ∴ Not Normalizable. ∴ This situation arose in a (perhaps) different context ···· in the idea of "Real" p.d.'s — which I that to be common. The "Soln." to this problem is that p.d.'s are always normalizable (i.e. $\Sigma pc < 1$) — If they are not, then I've not been using a real pc. — That one could always fix problem was formulated improperly — That one could always find a "proper" i.e. normalizable. P.D. to Guide [redacted] fr. Search.

Consider a population of men where probably no one will die before 73 yrs? ~.5; certainly $p(1)$

In this case, an exact would be: what probability man would die after y years (Cumulative $y = 0(\infty)$. Or, die in his z^{th} year ($z = 0(\infty)$ (non-cumulative: $\Sigma p(z) = 1$ Kcosts)

Or, consider cands with [redacted] between 0 & 16 bits: What probability any one will have $G \geq$ [redacted] Go? I.e. pc. summed over all such cands (There are [redacted] $(2^{16}-2^{10} \approx 63\%$ of them), should not sum to 1.

.32 When we do this "conditional pc" search for (x) ∈ A, (or in an OT), we have to be clear

.34 about just what this PD is! I think part of the proof of the MCT deals with this.

(.32–.34 is Tricky!) T. P.D. has to be for mut. exclusive "events" (or whatever) what RA (cond'l) Rd. do we use in OZ (or OTs).

In Early work on this I considered [redacted] just for OZ probs & mistakenly thought that probably that a given cand had prob. (Mut. exclusiveness).

I'm probably sure I didn't use that approach in the proof of MCT.

.01: In t. present context, I want t. condl. pc of dcrn, x, given it's known G value. NO! (GG)
T: results are different, depending on whether G is "open" to t. ume, or not.
Narrowly defined, condl. pc of y wrt. cmde x, is pc of (x,y) / pc (x)
[pc of x,y is variously defined: accuracy, t. probability that, in random input, ume will print x,y & stop. (Or perhaps y, x & stop). We might find a way to restrict def of pc of "y" so cond pc of Λ given x is 1. & also cond. pc of x given x is 1. (?). I don't immediately know how to engineer Plaisted properties of condl. pc.

.09 (.01 is wrong: I want t. pc of y, G, given x, G. If G is
.10 "open" to ume, reference Run a "G" is automatically ("free") & assigned to all possbl. y values. — so I guess, pc of y, G, given x, G should be = pc of y given x.
So t. "Open" condition of .09 - .10 is not so important in formalizing / using Kol.
However, we can have various amounts of openness: In one, we have a pgmmer as before but w. a CB limit — so for only certain y values is G "frozen".
.15 Another penalty is that t. ume (given) has to "freeze" a probabilistic realization of y.
.16 : 92.04 Some general. of condl pc: Given an unordered finite set $[x_i, G_i]$, $i=1(n)$
.17 what is pc of x_{i+1}, G_{i+1}? Given sequence α; what's pc of α^β? {Usenet ALP}
NB ≠ $[x_i, G_i]$ can (often) be Rep of pc's "BAG" — even if the x_i are not all positive integers.
.16 can be used to define in doctrine an unordered set for Kolmogorov complexity. Say, given corpus $[x_i]$ $i=1/n$ — woops, we want a "bag" corpus, not a set!
So t. or functional .16 to apply finite data set corpus would only work if cpcty were all 1 and 0 only.
.24 Some confusion in .16! What is t. pc of a set $[x_i, G_i]$ or even just $[x_i]$? Several possbl. doubts: T. pc of (a random input) ume will print $[x_i]$ & stop (any order is accepted). — we can easily frame this to Kolmogorov, so it has to print all x_i, any times (in any order) before it stops.
Another doubt involves in schemes Plaisted defines t. set $[x_i]$.
SN Just what was t. defn. of "Join" for "reasons"? Was it an "AND" or an "OR"? or ....?
.31 Anyway, given a single x_i, G, this implies a Def. x_i, G_i for all x_i, G_i.
.32 Also, given a (set) $[x_i, G_i]$ this implies D.D. over all x_i, G_i.
So, in G-N terms, .32 is generalized crossover?)

.33 " " crossover
well, re: .31 is "Mutation"; We shortcut via a fixed (a priori) $x_i \to$ P.D. over x_i
.32 "crossover" " " " " " $x_i, y_i \to$ P.D. over x_i', y_i'.
Mutation is then a kind of crossover of x_i, A (null) $\to x_i', y_i'$, but we discard y_i'.
HWR: .31 & .32 are confusing (both better than / normal) mutation or crossover — or even named unnatural crossover.

Normal Mut.'s Crossover do not use G much - apart from selecting X's to mutate/crossover that have hy G (probabilistically ∑ - Monte Carlo).

108.01 ff. (& 108.16-40 in particular) have some new ways to look at induction as special cases of Condl. pc. Also Gonzns of Condl. pc (108.16-17 in particular)

.06 Consider "Mutation" = given X, G → P.D. on all $[X_i, G_i]$

.07 = $\frac{pc[X, G, x_i, G_i]}{pc(x_i, G)}$ May be also regarded as stoch operator X, G in $\xrightarrow{P} X_i, G_i$ out

Clearly, these condl. pc's are critically dependent on what prime is used. The way we begin to fix the prime, is to make this condl. pc, also conditional on a previous TSQ.T:

When people design Mutation operators, they are designing the stoch operator (.07 ff.), based on their own "experience" (≡ TSQ). In general, this stoch operator should depend much on the problem domain, the problem itself (& perhaps on X, G?). (often the mut. operator will depend on G via mechanics of G. If we seem to be at a local max, we will run my longer mutations to get out of it.

For mutation types, we should keep score of them. Ones w. hy scores should then be given more wt. (Just how to do this Quantitatively speaking, is unclear ∑ — say we had a M.S.D. for the ΔG assoc. w. each mutation type. (perhaps M is really neg. exp., so we may be merely interested in hy of mutations.) — Probly Law Of Iterated Analysis is relevant.

Anyway, In generalized Crossovers we simply use a ≤ SSZ of Z (rather than SSZ = 1) of the pure mutations. Actually crossover gets Z cands from Z "parent" Condl. pc gets a continuum general d.f. on all X_i, G_i from 1, 2 or more cands.

((Would have much better P.D. than using only 1 parent

However, people devising mutations may have been more "creative" than people devising Crossover types (probably except Koza ∑ who has devised Adf's, loops in ADFs, recursive ADFs.)

.33

So, I would be good idea to try to characterize the set of (hy) G's first & second moments of various substrings (Walsh) would be of interest.

.36

.35 For recent method of H.C. using pc's obtained by "large samples of cands": Can choosing (Boltz) OT's used in Normal OZ (Sorkin) be regarded as a kind of "coding" in connection w. method of (35)?

This idea has occurred to me before. Anyway?

110,26 spec.

MCE quotes = "Holistic" or "Integrated" BSC
BROAD SPECTRUM LEARNING".
"Wholistic Learning"

0 - 2:30 No duc
2:30 on | DDCon.
6' 40"

.oc When Many runs are done in GA (or SimAnneal), using different sets of initial random seeds: I don't think they remember from 1 run to next! Very Unsophisticated!

.04 I had Brainidea that if G evaln. was Very Expensive, then TM should try to devise a low cc Model of G to be used in early BOF search in GA (or any other OT method). Hvr, finding such a model seems close to the OT goal — so perhaps they should be worked on together.

Another idea about -0+1 If the original hy cc G function is "closed" to TM, its own Model of G is "open" & so it could study the structure of its own model as kind of OT (approximate OT)

.11 A common Method of "Model Construction": "Fuzzy Controllers"; to get an adequate set of representative G points, then use linear (or other) interpolation. The literature on "Fuzzy Reasoning" may have Good Suggestions in this Area. This technique might work well in "Broom Balancing / inverted pendulum".

Normal GA matches (w. crossover) could be part of a kind of "Model" for (.11) in which the seeds of known G are given wts and may all scream (Pandemonium) to be have their sub branches (Made part of accepted as far) the new Cand. (or some similar method of "interpolation".

.19 : 97.40 Re: T. "Creativity Level" of Koza's gens: At first glance, it looks like the problems were inherently easy — in the sense that only ~ 80 M evals were needed to solve an "Analog ckt design" problem. Hvr, this seems may have been much more "creative" than usual because of the Grossly Limited Pop Size & the very non-elite search. He really considered many "far out" possys. Whether this would work in other, more diff/R problems, is only somewhat unclear. ↳ 10" hz desktop.

$2^{10} = 10^3$
$2^{20} = 100$
$2^{200} = 100$
10 yrs = 100 !

.26 : 109.36 .40, .33 So, using the ideas of 10816ff on Metabaseline Crossover as being Chaotic approxns to the $\Sigma[\text{Cands}, G_i]$ d.f., and having the most General possl. Ways of approximating Rest D.P.F. I should be able to Crit Eval/improve normal GA methods a lot. Each approxn can be regarded as a coding of data (in Mutation & Crossover: very little data). I can then measure just how Good these codes are & suggest improvements!

↳ In the case of pure mutation, the $SSZ = 1$, & no info is mainly in the approx.

This is essentially info on what the User thinks are reasonable schemas to get Cands of about the same G, but in (hopefully) hy cc. This could also be done by studying how good each mutation has been in the recent past, or on the Best 100 Cands Rest from (which is a Xwindo or Rwindo on the ΔG of the mutation ... The G of the parent ?? can be (like one Time dim Xwindo (?). One could reconstruct (& Xwindo) a mutation as many times as used on a Cand of a newly G(?) Perhaps a better way to think! Consider how one does Xwindo when the time data is not spaced uniformly.

6/21/00 Bugs. Hardware (F3)

Perhaps this exponential wt. wrt. Q is close to Simulated Annealing!

I did analysis of this in SM notes. There was a 'normalization' update in documentation.

Perhaps t. wt. of each new data pt was $\propto e^{-\frac{\text{elapsed time}}{\tau}}$.

So I updated numerator using data; updated denom using 1.

This assumes a finite, not too large, no. of Mutation types.

As G gets bigger & bigger t. wt. of new data thrown by ΔG's $\uparrow$ exponentially w. G.

.06 G of Parent & wt.; ΔG of Card(child) is what is recorded. (.22)

"Crossover" is more like small ssz induction: ssz is 2.

.08 On Crossover: When 2 properties are synergistic, we want to define Resolver "AND" as a new property. "Synergistic" means $\Delta_{1,2} > \Delta_1 + \Delta_2$ or some similar relation. Δ_i could be the ΔG assoc. w. 'Operation' (by Parent) → Card. $\Delta_{i,j}$ means Both operations on Parent. (They may not commute!) — The m

.12 Biology, the operations are all one-way, so they do "commute".

.13 ~~Hardware~~ It would be well if TM was written in ~~C~~ so it could be run on Various Machines. In Particular, it would be good if f. "Learned" parts & f. "ESQ" parts could be easily transferred to a different Machine. — This will be easy when the "New Machine" is periodically "Upgraded".

"Forth" can run on many Machines (I think) & TM seems to be a lot like "Forth"; T. "library" would consist of a Dictionary: word of words, perhaps.

Perhaps TM psme should be written in ~~C~~ "Forth" (The ~~Super~~ Super, "Factor machine code" version.

.22 (.06) Re: Mutations: They are x-terms like in .08-.12; As such we can learn to combine synergistic Mutation Operations to make beneficial Mutations. This sort of thing could ~~also~~ improve "Mutation" considerably!

OSL may not work on Mutation: to have ssz=2 f. child would have to be 2 parents! For "crossover": 2 parents, OSL could be meaningful: ssz=2. for distant parts of child:

.27 Say parents = AB & CD; if child is AB then BC

.28 A & D have ssz=2. Similarly BC could be child w. BzC having ssz=2.

.27-.28 would deal nicely w. knots crossover. Hmm I think it might be better to deal w. other things like sub-treaks. — i.e. in .27-.28 we can interpret B & D to be sub-treaks with same no. of terminals.

— also can then exchange them, → AD & BC.

.28 If it's said AB & CD both have same G value. If not, it's much more complex: I don't know just how to deal w. this. The G of child can be anywhere between G_{AB} & G_{CD}! What we'd like to make be easier to guess — maybe closer to better — or

.38 maybe use best & P.D. over all G values. → 11.12 (Space)

.01 If I have a set of trees w. their G's, & G is completely "open" to TM, then for any tree dom, the assignment of G is unique, so if I have a population of trees, C only have to MAKE such gen'n for tree dreams, not the G's. However, as is usually the case, either G is expensive "closed" to TM or "notoriously inevertible" — i.e. we can't easily get all tree doms of Max G. Then we have a problem; (G is (2nd is almost always true (common non-easy invertibility) — In Pearl Pic "inversion" is what typically (or) problem is about.

So, the problem is the SIGA problem; to get an easily invertible, approx to $P(G, \text{tree dom})$. We want this function to be such that we can easily find tree doms of by (P, G) (whatever that means!)

.12 ~~11:35~~ One may be able to crossover G def.: G of child = mean of G's of parents, but there is a QV that we obtain by averaging over past cases. [I] maybe that we can empirically obtain a G, z parents → G child distribn: so then we need not use mean of G d.f. of child = mean of G's of parents.

Good! Another possibility is that a child could be more like 1 parent than the other parent. So G distribn would not be mean of parent's G's. [degree of likeness" can depend on how much

Hvr, a much harder problem is when one has ≥ 2 parents: * How .20 G, P is obtained for children. The "genes" of the child would have larger sizes, but they come from parents of DIFFERENT G's, — which is about where I was at the Beginning!!

info child gets from each parent. ~.35

There was this idea of Pearl's second Moments in G for a particular subtree I think I'm assuming G's of Pearlness (= subtrees) are ≥ additive. .24 ~.35 Not as necessary In mapping from input to output (or even w. no input, between outputs); the function tree does not have loops, Pearl is certainly temporal/cap computability in the action of G that. After each function has acted, it will have a certain output, so at that point at each ΔT, we will have several "outputs". If we like, we can suitably "trim the tree" so any one of those outputs is the "final output". (it's like specifying the location of the output string in the cost comput'n)

→ The fancy is the same as picking an arbitrary point in the tree as "output". If a "tree" has feedback, (recurrent) then when we pick an output point, we also have to decide when we will accept the output, because the values at the particular pt. in that can change with time & perhaps never "settle down".

.35 (208) One large population of [trees, G's] Then G of a child depends on the G's of its parents. The G's of parents are obtained by distribution funct. over parents' Gene Pool features. Also, each feature has a certain contrib "info" assoc. w. it. The G values of features w. more info, get more weight in determining the G of the child.

"phi", anthropy & very similar. Like anthropy

only 3% of all genes derb. pro teins: DE wash to reach one "Control Genes".

Into

Note that ↑ log in a feature need not be related to its "size" It should mainly depend on its frequency in ↑ corpus — but this is also mixed w. G values!

Well, one way to derb ↑ corpus w. it G's is 3 grammars, each w a different μ_G, σ_G^2. T. grammars can be all ev, have common parts, w. small differences. Th. Grammar can be ↑ from Grammar, w. ↑ first pc. for each "fraction". Or, ↑ 2nd derb. reads w. top 10% of G values, as a sub corpus w. a μ_G & σ_G^2.

Or have ∃ 1 tree representing ↑ grammar of ↑ whole corpus, but certain branches have certain μ_G, σ_G^2.

Actually, ↑ idea of a stoch Grammar is not really what is wanted; what is wanted is an easily invertable G(cond,dem). Mainly invertable is by G region. So, if ↑ conds are to be represented by "cracks" sub trees, — how do we get a G estimate?

If we ∃ seperate a G (+ or −) w. each "part", I think we also want to have by G w. small no. of parts. If G is mean of ↑ of parts (which is reasonable), we may also want to weight each part; so it has 2 parms,

.16 w_i, μ_i : total G appears of a cond is $(\sum w_i G_i)/(\sum w_i)$.

Maybe also have σ^2 for each part.

——— Nvr. ↑ main immediate goal is to get a bunch of parts (& assoc parms) into which ↑ conds can be linearly fit give good G approx

.20 Eq. 16 w_i & G correspond, in statistics to $\frac{1}{\sigma^2}$ & μ — so .16 should give

.21 $\frac{1}{\sum w_i}$ as ↑ σ^2 of ↑ resultant G! If we could actually try $w_i = \frac{1}{\sigma_i^2}$

.22 Easy to calculate.

.23 Hm: It would seem to work out badly: If A & B were parts of hyrdra,

.24 A & B would be best conds! Also A would have More σ_G^2 than A∪B (A desirable)

Well suppose we cond on part of ↑ grammar & ↑ G contain parts are seperate.

I/E could make stoch Grammar that ↑ Cond dems, then assign approximate G's using

(.16) (.20) (.21) ↑. We could compute ∃ pc of ↑ corpus using ↑ formalism. If it did much compression, it would be good for prediction.

- ——— There seem to be 2 pc's for each cond! ↑ pc abt ↑ cond dem.; & ↑ pc assoc. w. its G dif.! What does this mean? I would guess, simply take product of ↑ 2 pc's. Each "part" has a pc, a μ_G & a μ_{G^2}. = wt. —— But this system goes by pc to conds w. few parts. — So objection #3 — .24 would hold.

.33 How well ↑ Model w/ .22 seems [redacted] reasonable! It should be able to check a corpus & make corrections.

One possibility is that one will simply have to try A & B as conds. (#.23–.24) Using ↑ Huffman Coding trick, it should be easy to list all [redacted] pc order. Can I list them in "p" order? [redacted] "Expected, or most likely" Probs.

If evaln. of G is very expensive, it will "pay" to do-rather careful evaln. of relatively complex models of ↑ as a fnct of cond dem. In such a case, it

Might be OK to list cands in PC order (highest first), then get $\hat{f}$ of each, — after evaluation of say 100, put them in $\hat{f}$ order & do GA evaluation.
Or, ~~as new evaluates "f" values, put them in "$\hat{f}$" order, [illegible]~~.
This can be done by having "Bins" for each small range of $\hat{f}$.
~~T. seeds a generator~~ T. ranges & bin widths of $\hat{f}$ bins can be estimated by
Studies of range of G values of interest.

Perhaps $\hat{f}$ should be an additive, rather than an average of ~~G~~ "parts"

Go thru eqns for "unsatisfactory soln" ! ~~Some that it's~~
All correct! Count no. of cands, no. of eqns, etc.
T. Grammar needs another symbol, that tells it when to stop---.
When the cand. is finished! Otherwise, cands are all of 20 length! [I don't think this helps; it affects to ↑ cost of all cands by same amount.
Also, look up Sears work on Fanchon Grammars. Let told how many functs one could have ~~with~~ in N dim. funct, I've forgotten Grammar
and how to function Grammar works!

.16 OK! Lower model ~~minimum~~ of 113.20 ff 2 pts, 113.33 ff is prob by 0.4.
We do get the "dirty" of 113.23-24, but it's not a "dirty" 1: it's a reasonable ordering of cands, considering the available info. Still, I can use either average M_G or additive M_G for estimate of $\hat{f}$ of Cand. — Perhaps an empirical Q: ~~which~~ cf. which methods gives $\hat{f}$ closest to G?
Using weighted mean → 113.21-.22 is correct, since M_G & σ_G^2 are easy to estimate. — T. Q is - are they any good? $\hat{f}$ = estimates. "Old cands"
3 Operations of Interest: (1) Deciding on "Parts" of ~~random~~ cands of known G - & old cands; "Parents"
(2) Assigning $\hat{f}$'s to parts (3) Combining parts to get $\hat{f}$ of Cand kids

T. problem is More like: Que is given a Pool (X_1, Y_1) points & asked to try to reconstruct a function by asking for as few new X_2, Y_2 pts as possible. The known set $\{X_1, Y_1\}$ is clearly not a "random choice of pts".
The initial population is random, but not subsequent populations!
So our wife want to try to take advantage of this initial "Random" def. (i.e. take advantage of fact that it is Random).
One trouble: If one has a set of Pers, it would be reasonable to try to get the $\hat{f}$'s of new pops by evaluating each of them as [entire kids]! I should think that each would get low $\hat{f}$, because only in combination do we get many (any?) $\hat{f}$.
→ Best thing to Do: Decide T. problem in detail: Recon last pop. cands. & their Weaknesses.

6/22/00 Bulg.

Def prim

.01 An Approach: Say we have the first several primes, A, B, C, D.
We start out by listing all strings of them in lex order (= alpha for A, B, C, D)
Since each household size goes same as uniform random d.f.; we can
begin to evaluate β of each prim. (using addressing model of β)
But if its address is "A" is "Bingo", Regn 1. Bogus guesses start by β consists AA'A····.
Now whatever worked w. using 113.20 = z1 (μ_z & σ_z^2)?, 113.23-24 says short strings
of by β of B would be of lowest β. If we instead strings in order of length, for ℓ (length) n
A(n) would always be first.
If A, B, C, D was order of β values, trials of strings of length n, in order
of β, would seem to be "perfect guess!" Proof would be fairly based forward
t. by β prims.:

For finding a single soln: Using Mt carlo pc wtd search has same expected soln.
time as uniform search.

.14 ① pc based Mt carlo: $\sum_{i=1}^{n} p_i \cdot \frac{1}{p_i} = n$ = expected find time. or $\frac{1}{2p_i}$?

.15 ② uniform pc = " " : $\sum p_i \cdot \frac{n}{2} = \frac{n}{2}$ ← This looks wrong (Prob'ly should be n)

$\sum_{n=1}^{\infty} n(1-p_c)^{n-1} p_c \qquad \sum_{i=1}^{\infty} i\,(1-p_c)^{i-1} p_c = p(1-p) \sum_{i=1}^{\infty} i\,((1-p)^{i-1} = p(1-p) - \frac{d}{dp} \sum_{i=1}^{\infty} (1-p)^i$

$p(1-p) \frac{d}{dp} \frac{1-(1-p)^{n+1}}{p}$ $= p(1-p) - \frac{d}{dp}\left(\frac{1}{p}\right) \cdot 1 = -p(1-p)\ln p$

$\sum_{i=1}^{n} x^i = \frac{1-x^{n+1}}{1-x}$

Sounds wrong: $p \sum_{i=1}^{\infty} i\,(1-p)^{i-1} = p \frac{d}{dq} \sum_{i=2}^{\infty} q^i = p \cdot \frac{d}{dq} \frac{1}{q} = p \ln q$

$= p \ln(1-p)$

$p + 2(1-p)p + 3(1-p)^2 p \qquad p[1 + 2q + 3q^2) \cdots p\left(\frac{d}{dq}(q + q^2 + q^3) \cdots\right.$

$\left(\sum_{i=0}^{\infty} q^i\right) = q + q^2 \cdots \qquad (1 + q + q^2) - 1 = \frac{1}{q} - 1$

$= p \frac{d}{dq}(1 + q + q^2 + \cdots) = p \frac{d}{dq}\left(\frac{1}{1-q}\right) = p\left(\frac{1}{(1-q)^2}\right) = \frac{1}{p}$.

$= -p \ln p \cdot \frac{1}{p}$

so average $= -\sum p_i \ln p_i \cdot \sum \frac{1}{p} = n$

Bogus 15. Bogus

p = .3, say N = N+1 M=0, M=0, Z=0
If RND < p then M = M+N : Z = Z+1 : N = 0
Print Z, M/Z, -p*log p
Goto 10
p = .3 ; 3.32 .361

p = .3
10 N = N+1
If RND < p then M = M+N : Z = Z+1 : N = 0 : print Z, M/Z, -p*log(p)
goto 10. Answer — 3.32, .361

Try p = .01 101 .046

If we do trials in pc order, then expected time is fixed number of
$\sum_{i=1}^{n} i\, p_i$ = I worst case is uniform pc : $p_i = \frac{1}{n}$.

$\frac{1}{n} \sum_{i=1}^{n} i = \frac{1}{n} \cdot \frac{n(n+1)}{2} = \frac{n+1}{2}$ (so slightly worse than Montecarlo uniform, is still better than n.)

So Prob results would seem to make fitness proportional to probability of Mutation/crossover look very bad!

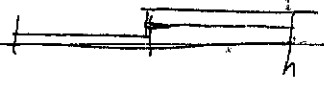
The results are different if assumed pc ≠ the real pc.

In 115.14: Say p_i^* is true pc; p_i' is assumed probs.

$\frac{1}{p_i^*}$ is expected time for search; $\sum \frac{p_i'}{p_i^*}$ is expected total search time.

If p_i' is usually $>> p_i^*$. no: $\sum p_i = \sum p_i' = 1$.

Still $\sum \frac{p_i'}{p_i^*}$ can differ from n (+ right)

Eg  $\frac{1}{n} \times 1.5$; $\times \frac{1}{2}, \frac{1}{2}$

3 .33 $\frac{3.33}{2} = 1.67$

In P.S one case, where $\frac{p_i}{p_i'}$ was 3 for $i < \frac{n}{2}$, .332 for $i > \frac{n}{2}$,

The worst search time was $n \times 1.6$ rather than n.

.13 It may be that $p_i = p_i'$ give min. search time ← Maybe not same .25

Here, use by searching in p_i' order w.i. $p_i \neq p_i'$ it could be very bad.

Say $p_n = .1$ but $p_0' = .9$ s.t. p_0' is smallest.

So it takes me n hits each time. — which is = to 115.14 (Pc based dist (Darwin))

So why does GA work at all?

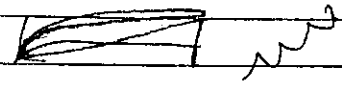
p_i = $\frac{p_i'}{p_i'}$ s min. $\sum p_i' = 1$

.19 $\frac{\partial}{\partial p_i'} \sum \frac{p_i'}{p_i'^2} = \lambda$

$-\sum \frac{p_i'}{p_i'^2} = \lambda$ so $p_i' \propto \sqrt{p_i}$ which flattens the d.f.

I guess not surprisingly because $p_i' = \text{flat}(\frac{1}{n} p_i^{\frac{1}{n}})$ however, does better than $p_i' = p_i$.

$\sum \frac{p_i}{k\sqrt{p_i}} = \sum \frac{\sqrt{p_i}}{n}$ $\geq \left(\sum \sqrt{p_i}\right)^2 k = \frac{1}{\sum \sqrt{p_i}}$



.25 This seems to contradict (.13)!

Small n GA can actual useful (sub-trees/structures/parts) —

but in general, doesn't (directly) enhance search strategy for big GA at all!

→ In .19 it looks like second derivative is positive so $\sqrt{p_i'}$ is at a Minimum! ← Check this (Numerically, Hm — should be by minimum!)

My — search (if $p_i' \propto p_i^{\frac{1}{n}}$) the log N seems to give $p_i' = \frac{1}{n}$, which .31 is better than N = 1 —

Bulging Bags

For i = 1 to 10: $P(i) = \prod^{\beta}$: P(J) = P(J) / sum(P(J)) Next. ∝ output

" " " " " S = S + P(J) : S1 = S1 + P(J). Next. .1 9.66

" " each M = M + P(J)/P1(J)/S*S1 .5 9.1786

From M results 9.1786 .0 10 " .4 9.2098

This suggests (Hmm. .6 9.2094

With $\beta = .5$, still sums about $\alpha = .5$. So its sums a bit $\alpha = .5$ .01 9.965

But results = 10 for M = 0 or 1 ! not 5 ! .001 9.996

−.01 10.03493

1.1 10.38

.3 9.3809

.7 11.111

Bulg 117.BAS

For X# = 1 to 10 : A = X/10

" Y = 1 to 10 : B = Y/10 Bulg 116.BAS inside

Print using "##.### "; M;

Next : Print.

Next.

Saver 118.01 - 40 for output:

Book

I vaguely remember in "Adv. Gen. Pgms I": Rest of concept in a pgm, Koza replaced fitness dependent selection w. more random selection & it made no difference!

Since the choice of the fitness function is arbitrary. They could choose one that is equiv. to $x\sqrt{\rho_2}$ y in which case, if the P.d. were of form x^5, say hyperconcentration at hy p.s, then the $\sqrt{\rho_2}$ would be somewhat useful. So the success of a run could hinge centrally on choice of fitness function.

However, the idea of mut/crossover being a small size perturbation of a bag of finite objects, seems basically sound — that it looks like a very correct approach. 108.16 ff

— An argument against doing trials in order of Expected (or some closer): That using strict order maximizes that trials that are adjacently bound to be of similar structure, & hence about same G. So to find a hy G and must rapidly, one should space trials so that adj. trials do not have correlated G.

.21 Previous Argts were against GA working at all! — Another on why they may work: One starts in a region of cand space: One that picks region of hyps G. Then centers search on that region. As more hy G pts are found, the center of search moves toward those pts. Ideally, this can be a branching search, but in fact, it doesn't, because branches of search are combined by "crossover".

So how does .21 A. deal w. the argt vs. fitnessly chosen crossover/mutation?

Well in the argt of 115.14, low pc "wins" and rmpt — in GA, they may not be. One tends to find that hyper pc "um" rmpt first, & not speak so much when low pc "wins": → 138.01

138.01 it's not a good picture of how GA probably actually works & compares it to SGA

.31 Right now, I could just go in TSO's: Arrange so that ALL of TM's problems are solved & then I solve them — this includes both OZ probs & EXU probs.

However, I'm still interested in GA, because even if it seems like a very unrealistic Prob solving Method, & it may be able to improve it tremendously & use it as a common OT. Also, it maps into MCTM, in an apparently useful manner.

119.01

6/23/00 Bulg: output is F:\SM\Bulg117.Bas

B → / A ↓	.1	.2	.3	.4	.5	.6	.7	.8	.9	1.0	A ↓
	9.996	9.984	9.964	9.939	9.908	9.872	9.833	9.789	9.743	9.695	.1
	9.993	9.971	9.937	9.892	9.837	9.774	9.705	9.629	9.549	9.465	.2
	9.990	9.962	9.917	9.858	9.787	9.705	9.614	9.516	9.413	9.304	.3
	9.989	9.957	9.906	9.838	9.757	9.664	9.561	9.450	9.332	9.210	.4
	9.988	9.955	9.902	9.832	9.747	9.650	9.543	9.427	9.306	9.179	.5
	9.989	9.957	9.906	9.838	9.757	9.664	9.561	9.450	9.332	9.210	.6
	9.990	9.962	9.917	9.858	9.787	9.705	9.614	9.516	9.413	9.304	.7
	9.993	9.971	9.937	9.892	9.837	9.774	9.705	9.629	9.549	9.465	.8
	9.996	9.984	9.964	9.939	9.908	9.872	9.833	9.789	9.743	9.695	.9
	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	1.0

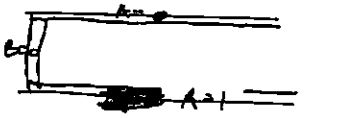
smallest

```
F:\SM>
F:\SM>
F:\SM>
F:\SM>
F:\SM>
F:\SM>
F:\SM>
F:\SM>
F:\SM>
F:\SM>
F:\SM>
F:\SM>
```

Table is sym w.r.t. $\alpha \to 1-\alpha$.

output is monotone ↓ in β.

$A=0$, $A=1$ & $B=0$



$p(J) = J^B$ / Norman Factor.

$p_1(J) = p(J)^A$ / Norman Factor.

B practically 1 will [illegible] & output

are all lines w. trails = 10

for $B=2$, $A=\frac{1}{2}$; output = 7.930 trials.

$B=5$, $A=\frac{1}{2}$ " = 5.167 trials.

$B=$ [illegible] 10, $A=\frac{1}{2}$ " = 3.27 trials.

So, for $B \gg 1$, ↑ defines bdry. $A=\frac{1}{2}$ (optimum) & $A=(1$ or $\phi)$ (trying) means 10 trials is maximum. Also, if $B \gg 1$, Rec optimum method is of [illegible] crude approx, is very good.

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

Non-EL

.01 .01 .01 So: The Main Result is 1. TM_1, TM_2 combination implied by 2a (Non-EL) MC Thm. soln. of Gen. TM Problem. If last property Pres w. 2 minimal, intelligent start, & a reasonable TSQ, it will eventually become arbly "intelligent"

.04 I want to start off as by a (as little as possl, how, a GP seems like a possl. start for a good part of TM_2.

So .04 is partly why I'm interested in GP. Also, I want to understand OT's better, & GP has important features to study that are common to many OT's.

—— A summary of Main "Progress" in GP understanding:

1) Mutations are most easily understood. They are size=1 induction — A "conditional pc". The condition being "mutate" (output=Mutant). Also, other "conds" may be dern of problem's entire TSQ corpus

Ideally, TM_2 would watch how effective a certain Mutation & chromosome & the mutation & change (& Crossover operator) work to co-evolve in the problem solving system ($TM_2 = TM_1$)

.15 2) Generalized "crossover" is viewed as induction w. size ≥ 1. From 2 (or more) (Cond: G_i) pairs, one induces a pd on all (cond: G_i) pairs. ~~This is a real~~ The pd. is a cond. d.p. as before, & ideally, it accounts of (1)

Some problems involved in (15) I want the p.d. to be based on a recognition of the Parents. This can be an expression of parents as subtrees of operators or functions or whatever. How to get Pd. on kids isn't clear. Kids can be formed by compositions of operators in parents. So Given parents & kids (& G's of parents) all expressed as operators, how to get G d.f.s of kids? Also how to get def. of G of kids in a form such that we can easily find kids of by expected G.

[Another Q is: How does "soln" of .1c (i.e. 20 ff.) accord w. Koza's crossovers, ADF's, ADF loops, Recursive ADF's? Can I improve, generalize his techniques?

.28 Two models proposed for .20: If we somehow assoc. w. each (function operator) a μ_G, σ^2_G & we express G appropriateness of kids as either (1) Sum of μ_G & components & (2) $\pm \frac{1}{\sigma^2_G}$ wtd sum of μ_G & components.

.32 Th. Diffs: (1) suggested very long kids w/ repeated operator of Max μ_G.

.37 (2) suggested short kids w. " " " " "

Obs. 2, (2) seems (easy bad. The kids would be listed in expected G order. —— Short kids usually first.

In 119.32 ff: It seems that there's something wrong, yet 119.15 is a "complete soln. of the problem in terms of ALP's. Given a set Σ cond$_i$·G_i] to get pc of an arby cond G_j. Here conds are stored in terms of known operators. So the terms are strings of those symbols that represent functions. [It is expected that substrings will later be deemed to improve systems.] If I don't use ordering of symbols, each cond is a bag of symbols.

.07 → [Perhaps I can list all pairs, triples, models (combinations as above).] → 121.12

Each symbol has 1 or more params. We can add on Mult params to get approxn. An advantage of few symbols is ease of prodn.

Bub Suppose that we're only interested in predicting (= pdf) G.

Consider .07: Consider all possible functions of a (small) bag of symbols. Say the function is indep of the ordering of the symbols. One way is to assoc (one or more) params w. each symbol: So we have all symmetric functions of K symbols! The i^{th} symbol will have its escort of (each) a (usually different) param. So the set is sym. functions of K, 2 vectors. At first glance, monotonic in each of the 2nd params (say escort = w_i, the other param is g_i, $K \geq i \geq 1$ (k).

.17 Say all $w_i = 1$; A symm linear function would be $c \Sigma g_i$. | 1. only possy. → 121.12

In fact, if the ordering of the symbols (= functions) is ranked, the actual G changes. So the best we can do is get the G approximation to be a mean of all $k!$ orderings of the k symbols (and many more if some of the symbols are repeated.

Perhaps the 1. approached G, if there are an enormous no. of the k symbol objects (the Rhs of their pc, I wasn't really thinking about that)

Perhaps merit. by pc is reasonable!

So consider: ① $(\Sigma g_i) \cdot (\text{pc of string})$

② $\left(\Sigma \left(\frac{g_i}{\sigma_i^2}\right) / \left(\Sigma \frac{1}{\sigma_i^2}\right)\right) \cdot (\text{pc of string})$

(SN) Earlier stuff on 105 is preceding; I tried to get $\prod_{i}^{n} g_i$ as an approx to G.

Longer conds would have daily getting G_i G_i's — I think I had the g_i's normalized so $\Sigma G_i = 1$.

Actually, I don't really expect ΣG_i to be bndd. There could be an unbounded conds w. $G_i > \varepsilon$ certain amount. Those conds would be "rare", but possibly only by addition of "Infrons". More reasonable: Σ pc conds G_i is bndd. since Σ pc cond < 1

$G_i \cdot pc_i$ is a more reasonable object to try to normz. When we get a set of G_i values for K conds, & we normalize them, those normed values could change a lot when we add 555 The $G_i \cdot pc_i$ values for those k conds, when summed, are/are likely to change w. 555. ?? well No! say we normed them unto Σpc_i!

I.e.

.38 given → $\left(\frac{G_i \cdot pc_i}{pc_i}\right)$ | Hvr if we did $G_i \cdot pc_i \to \tilde{G}_i = \frac{G_i \cdot pc_i}{\Sigma G_i \cdot pc_i}$

Then $\Sigma \tilde{G}_i = 1$.

→ USING "BTREES" to sort wrt Real Size (.35)
(Ri.3) it's no "exact breakpoint" : correspondence below. Numerics to look at terms
→ Hz Gcorn "well known")

Hvr. $\sum \tilde{G}_i = 1$ is not desired, because then $\tilde{G}_i$ will [illegible] SSZ.
Would (20.38 R) tend to stay ≈ same as corpus grows?
No. $\sum_{Pcs}$ would ↑ w/ corpus size.
So consider either G_i or G_i pcs. Well, suppose we want to get G_i a "normalized" sorts like a pc. & all G_i are on (0,1)?

.06 $\tilde{G}_i \equiv \frac{G_i}{\sum G_i}$ would change w/ SSZ, but would tend to be ≈ constant.
No! it would ↓ as corpus ↑ & $\sum G_i$ wd → ∞.
SSz we only put top 100 G_i's in denominator ⌈ In Ri's case, $\tilde{G}_i$ would ↓ slowly
as one melted toward 0 by 6 soln.
If Ri's top 100 were essential corpus that one was interested in, then
maybe Ri's kind of "Normalized G" is not bad"

.12 120.17 (20.07 ', Anyway: ALP in some useful for a k param svgt: would list all symmetries,
k param functions in order of complexity (≡ Pc). Hvr, I'd like to have greater
notions of just what characterizes (& constitutes) a what those predictive
functs (Cs) to have. Perhaps just look at some natural ways
that G is (typically ⊕) a function of. Could draw.
Think of a Fourier more in terms of OT such in MCT, rather than
in G& terms (if Ri's is, indeed, distrib). (No GA is usually
based on induction from [illegible] 2 & sometimes (multiple) cands.)
While in MCT the prodn. uses a much larger corpus — but may use a smaller
corpus to consequence.

A possl Method: generate a random set of N cands.
From top 10% of G, generate 2 new ∴ N of cands — passes both Ri.
Nearest pc in the Grammar defined by former top $\frac{N}{10}$. From Rs
[illegible] set of N, take top 10% of Nearest Gs & repeat.
This avoids the problem of [illegible] predicting G's of cands.
It may be better than normal GA, but it isn't really the "Best Way" —
The Best way should try to predict G values

.29 (Laafoo) On My Objection to Monte Carlo pc or Fitness sampling GA:
Not really relevant; A more relevant Q: Say one Does interbreeding
2 corpus and does 1000 cands; Look at expected results (mean G)
of cand corpus using ⊕ 2 ways to select 2 Mates:
① ⊕ pc or G ② pc uniform. In My problem, E finds the clear
.34 best ① wins: But goal is distinct from mean no. of trials to "success" → 122.01

.35 "Sorting w.r.t. Real size": put real no. in binary notation & use Btree to → 139.01
put it in Canonical (≤ size) order!

.01 Re 128 & 29–35 I say we have a population/w. a certain G range: [Gmx to Gmn] of Genes
A certain set of Mutation operators will produce on Average a certain
→ ΔG ± δ, so in operating on a Gen of a given G
we will expect a certain fraction of Mutants to have G > Gmx.
ΔG ± δ will be about the same for Gen w. Gmn < Range (Gmin, Gmax).
Hvr, if we do crossovers (using a certain Crossover set) The G of
child will be some func of G1,2 of parents. In General, I expect
that δ for mutation will be < " " for crossover, so in doing crossover, one
can use a larger range of G in Parents, than one does w. Mutation.
I.G may be wants to keep at least 1 crossover parent w. hy G
region, but that other parent can have much lower G than a Mutant parent. → 138.01

I have been having diffy in getting Mutation model for G distribn
on kids as a function of Parents G, dcrns. Perhaps drop this for a
while & try to get ΔG d.f. as a function of parent Dcrn & types of
Mutation / crossover. Perhaps Read Lee Altenberg on this.

.17 On (Grain/Mesh) size: (There is previous ref. in last wk or so.)
In doing exhaustive (tedious) search over children, — this may
be inefficient if "close" parents tend to have "close" children.
I must be ↑ G of kids! Best to not do/complete exhaustive search but do exhaustive
search w. a certain "mesh size" so that consecutive kids are less correlated.
Hvr, since search is not exhaustive, I will be less likely to have "minimum
k cost" solns to problems — so may will prob. have "INTRONS"
So one may have to select Grain size to get "TRADE OFF" on 2 Desirata.

(SN) Incidentally "Grain Size" for A–D conversion for ALP to be used, is
always a somewhat "unsolvable" problem! I don't know if this is relevant to .17, hvr. ☺

ABCDE
I want to "Review", characterize, just how far I can w. k. position,
& indicate most likely paths of Soln ABcdo ABcdo. ABCDE ABcd.

.32 T. model for Crossover can be w. model for Mutation: i.e.
We have Conds w. G1, G2 and Crossover type i. The child is of $G_i = f(G_1, G_2)$
± $\sigma_i^2(G_1, G_2)$ — or use same σ^2 for all models. The functions to be
determined Empirically. Some possl. forms: $a_i := A + \frac{G_1+G_2}{2}$; $G_i := A + \binom{\min}{\max}(G_1, G_2)$.
Hvr, .32 is a rather GROSS Average type of "Prediction";
we should be able to get better results using all info in the 2
parents.

.21 While 122.32 is perhaps OK as an effective way to do crossover, w. periodic revision of Pmr (parameters of the) G_i & G_i^2 functions, it doesn't tell us much how to improve / optimize Prog Functions.

In most of GA's evolution, we are mainly trying to discover good "Parts" (≡ Subtrees). Actually, we may need to evaluate not individual parts but "subsets of parts".

I Mike got a better idea of how G varies w. mixtures of Parts, by looking at Koza's detailed crossover procedures for ADF, ADF loop, ADF recursive.

So far, I seem to have 3 methods of combining G's of "parts"

1) Pure linear, additive.

2) Linear but multiplicative 3) linear but w/ M_1, M_2 cross-product

3) Multiplicative. (4) possibly w/ multiplm. $G \to \ln G$; we do additive w/ additive term $\exp(\cdot)$

.12 Remember, this is not X. same as Bio evolution: Here, we can have clear(.20) separation of 2 parts. In Bio Evolu, the reality is all so complicated has 6 point (or nc) meaning!

A (maybe) New Tack: T. problem (extrapolating to $\sum$ rands, G_i] seems is a problem in induction, ∴ solvable by Lisrch. Is not, of course, a complete soln. to the OZ problem. [A more complete soln. would suggest what Conds. that would be desirable to check in a model or distinguish better models. ("experiments")]

.20 .12 T. Mult idea of .12 may not be so bad: In general, long strings of symbols are unlikely to be obs by G; This is because we would probly never find such seqns (except those by chance), & because we only find long seqns by expressing them as shorter sequences of "chunks."

So: 2 possys. in .12: (3) or (4) This seems w to 100.13 & .t. stuff leading to it.

We want $\prod_{j=1}^{k_i} p_{ij} \approx \beta_i$ Hm, I'm not sure I want to normalize that I asked for around 100.13

Anyway: maximize: $\sum \phi_i \ln(\prod p_{ij}/p_i)$ ∴ min $\sum \left(\beta_i \ln \prod_{j=1}^{k_i} p_{ij}\right)$

.24 $\text{Min} = \sum_{i=1}^{N} \beta_i \left(\sum_{j=1}^{k_i} \ln p_{ij}\right)$ k_i is usually small (<10); N is large, 100, 1000, 10000.

w.o. constraint, .24 min: p_{ij} all very small. Say $\sum p_k = 1$ (p_{ij} = one of p_k).

W. this constraint, min p_{ij} →

constraint

$\frac{\partial \cdot 24}{\partial p_k} = \sum \beta_i \frac{c_{ik}}{p_k} - \lambda = 0$ C_{ik} is # of times p_k occurs in $\prod_{j=1}^{k_i} p_{ij} \approx \beta_i$

So $p_j = \sum \beta_i c_{ij} \Big/ \sum_j \sum_i \beta_i c_{ij}$ i.e. 102.19 well! $\sum p_k = 1$ makes all β_i estimates < 1, which would be very poor estimates!

The earlier work started w 95:20 AM For normzn of $[\gamma_k]$

Say our constraint is that $\beta_i \equiv \prod_{j=1}^{k_i} p_{ij}$ and $\sum \beta_i = \sum c_i$ and all $p_{ij} > 0$

So a Q is: Can we get reasonable $\prod_{j=1}^{k} p_j^{c_{ij}} \approx g_i$ if we suitably scale g_i; i.e. we allow $\sum_{i=1}^{k} p_i$ = constant then? I.e. constant $\sum g_i \cdot c_i = \sum G_i$ may be what we want — But it's mathematically d/f/t (or seems to be: Maybe a few successive approxns?).

.05 (T≈SN) on 95.07 ff "k" is ssz; for large k, we want $\sum G_i' = \sum pc(x_i)$; which is usually $<<1$, so $G_i' = \frac{G_i}{\sum pc(x_i)}$.

If G_i' is "what we are aiming at", then use $k \cdot G_i'$ as (ssz cscnt) of K_i.

(SN) suggests idea: try to get for each symbol something like t. G_i "per pc"; Using a symbol (≈ "part") has 2 effects: ↑ G of cand & ↓ pc of cand. The pc assoc. w. a symbol is obtained from its occurance in t. Corpus of Cands. (Indip of "G" info).

So if A & B are 2 Symbols $P_{A,B}$ are their pc; $g_{A,B}$ are their multiplicative Incremental g's., then we are interested in $\frac{g_A}{p_A}$ v.s. $\frac{g_B}{p_B}$ (?)

So, I'm assuming a d.f.nt "true" pc d.f. than one observed? Possibly, Since t. Observd Corpus of Cands has More to do w. my Mut/Crossover Scheme than it does w. any Inherent "PC" d.f.

(.27) Crossover Thy Pics: Each symbol has its own "g" use: t. expected g of a cand, is like a Sum or product of t. g_i's (if contains; multiplied by its expo!) t. approp has nothing to do w. t. g_i's of its conds, but is mainly its "length" or pc of construction.

So, in t. make of (.27) for t. Corpus of data, we divide G observd of each cand, by it's pc, so we can then measure Rate effects of pc. g's of t. symbols, w/o. t. pc's effect.

.34 The model of (27) seems very reasonable, but I'd like to have a good rationale for it!

Anyway, w. (.27) we end up listing conds in G order — which is mainly by length. Say r_i is product of g_i to P_i (pc) of symbol i. so $\prod_i r_i$ would be the expected g of a cand w. a particular set of symbols. To maximize this we would just have the proportion of t. symbol of max r_i! Unless we used sum of g_i's times pc. In which case, short dense ones are better, since pc is exponential w. length, g_i is linearly in length.

Using this additive model, the estimate of t. g_i of Symbol i, is obtained by

.33 well: $pc_i \left(\sum_{j=1}^{k} g_j c_{ij}\right) \approx G_i$ so $\sum_{j=1}^{k} g_j c_{ij} \approx \frac{G_i}{pc_i}$ $(\equiv G_i')$ (35.c)

Of these are k symbol types: In the i^{th} cand of t. Corpus, there are c_{ij} symbols of type j. g_j is (j = 1 | k) is t. g value of the j^{th} symbol type.

NO! from .33 $\sum_i \sum_j g_j c_{ij} = \sum_i G_i$.

$\sum_j g_j \sum_i c_{ij} = \sum_i pc_i G_i$.

$\sum_j g_j$ (some number) $= \sum_i pc_i G_i$.

Well so much for this work; have .33 be an approx (we would as little have as poss!)

.01 (!!24.33!) say $G_i' \equiv G_i / pc_i$

.02 we want $\rho_0(\beta) \to \sum_j \left(\sum_q \beta_q c_{jq} - G_j' \right)^2 = \min.$

$\frac{\partial \ldots 02}{\partial \beta_j} =$ [illegible]

$\sum_j c_{ij} \left(\sum_q \beta_q c_{jq} - G_j' \right) = 0$ is true for $i = 1/k$.

So [illegible]

.06 So [illegible] $\sum_q \beta_q \left(\sum_j c_{ij} c_{jq} \right) = \sum_j c_{ij} G_j'$ (k eqns: one for each $i = 1/k$)

Uncertain as to what order of k

a Method difficulty: will k be large? Even if k is as small as 10, solving this set of eqns would be time consuming. Just getting the coeffs $\sum_j c_{ij} c_{jq}$ takes k^3 time. — And solving it takes k^3 time also.

Well, we just update the β_q values not too frequently!

.12 I still have the problem of poor Rsquared for Rij Model! [illegible]

Th. Main Justification would be if it did give G_i approxns w. some accuracy — enough to justify the k coeffs — But usually for ≤ 52 for which summation is $\gg k$ — so one wouldn't need much goodness to be legit. Anyway, output of the curve fitting is the expected σ^2 of Error — or simply σ^2

Re: (12) A way to look at it: If pc_i is small, Rij's because there are a very large no. of reads of the "size" L, and as $L \uparrow$ (à pc_i) the fraction of reads w. hy C_i the. So, for reads of length L we expect maybe, on the average, one read to have $G \approx \sum_{i=1}^{L} \beta_{q_i}$: If we have c_{ij} (of .02) then there are very roughly k^L reads of length L and the no. in contigs C_i: $= (j = 1/n)$ can be computed: Maybe $k^L / (\prod_j c_{ij}!)$ It is quite large.

While we may, indeed, have one or 2 reads of length L w. $G \approx \sum_j c_{ij} \beta_j$, (for a specific $\vec{c}_i$) on the average for reads $\vec{c}_{ij}$, the β_q will be more like $\left(\sum_i c_{ij} \beta_j \right) / k^L$

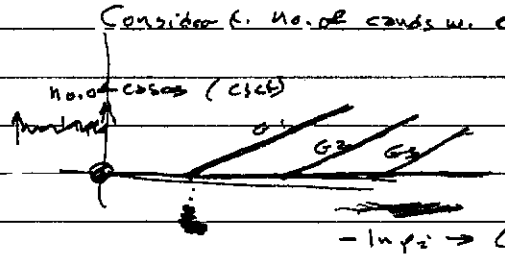
In making the estimate $G_i \approx \left(\sum_{q=1}^{k} \beta_q c_{iq} \right)$ [illegible] $\cdot pc_i$ (.02)

.29 → I write by fitting a $pc_i \to \boxed{pc_i^{\gamma}}$ — try to find a $\gamma \neq$ that fits better! Ideally, I'd like to predict just which reads had hits by $\sum_q \beta_q c_{iq}$, but the model won't that good! The only info it has is the $\vec{c}_i$ vectors.

↳ in .29 finding the best γ is computationally easy. Normally we obtain $\vec{\beta}$ by solving the [illegible] (from Matrix eq. .06. We can do this by inverting the matrix on the LHS, & multiplying it by the vector on the RHS. Changing γ means changing only the RHS of .06, which is easy to do, so we try various values of γ on the RHS & multiply each trial by the same inverted Matrix.

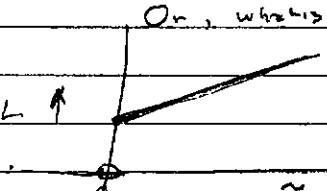
→ By "pc_i" I mean $p'c_i = \prod_x p_x^{c_{ix}}$; where p_x is obtained from counts w.o. G_i info. — perhaps $p_j = \sum_i c_{ij} / \left(\sum_j \sum_i c_{ij} \right)$

.01 Consider # no. of cands w. $\sigma > \sigma_0$ as a function of distance $-\ln p_i$ from origin in string Λ space. This seems like the right way to look at the problem!



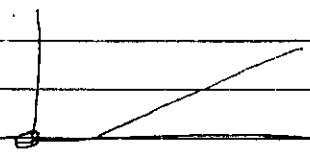
.04 For binary devices: An ℓ bit device is Euclidean distance $\sqrt{\ell}$ from origin. I want a picture of how, as Γ ↑ [redacted] looks for (binary) strings, # no. of strings N w. $\sigma > $ a certain $\tilde{\sigma}$ ↑, i.e. N as a function L, $\tilde{\sigma}$

Or, what is the length L of the shortest string w. $G \geq \tilde{G}$.



← Since G is arbitrary w. string + monotonic function, the form of this curve does not have much info.!

or as we ↑ L, the highest G [redacted] of cands $\leq$ length L!



While this "Threshold effect" is of some interest I'm mainly interested in Density (CSCT) of cands of length $< L$ w. $G = \tilde{G}$: see fig (.01-.04)L

Also interested in derivative = density at length $= L$

How many cands of length L w. $G = \tilde{G}$.?

→ For given L what is the D. R. of $\tilde{G}$ values? In abs units ($\equiv$ CSCT) or $\frac{\text{CSCT}}{2^L}$,

This last may be ≈ indep of L. It is only as we ↑ L that we get overdensity 2^L is (conjectured to be) the expected no. of [redacted] by G ("Soln") to be $\tilde{\geq} 1$.

Nice hypoth. for a model! Seems like a reasonably model! — Suppose that solns are ≈ uniformly distributed in Cand. space.

→ If all pts have ≈ same Hamm. Dist. large L is no distance from small L!

Perhaps it's something like ALP, in which short codes are mechanically to be correct, but longer codes often consist of short codes plus "patches",

Or that density is roughly uniform (w. a certain d.f. for G) — That the G value is (somewhat) (slightly) better modeled by the relative nos of each symbol in the Cand.

.30 So [redacted] $g_i = \sum_j c_{ij} p_j / \sum_j c_{ij}$ p_j is "goodness" of symbol j

We examine short L first, because (usually) they take less time to evaluate.

From .30 we should be able to get a simple formula for p_j in terms of c_{ij}, G_i.

Use eq.s 125.02 ff, but have $G_i = G_i \cdot [\quad] \left(\leq \sum_j c_{ij}\right)$

Since we want $G_i \cdot \sum_k c_{ik} \approx \sum_{k=1}^{K} c_{ik} p_k$ instead of g_i

Note Discussion of CC: Matrix inversion: cc $\propto K^3$ Also discussed in 125.29 ff: the possy of

.32 → Modifying $\vec{G}_i$ by say $G \to G^k$ or any other monotonic xform, to get a better fit.

It is cheap (cc $\propto K^2$) to make a trial xform.

.01 In the foregoing set of Models of approaches to G_i of cands, I was able
.02 to get ① $G_i \simeq \sum_j c_{ij} g_j$ → 128.17 for Various Criterions
.03 ② $G_i \simeq \sum_j c_{ij} g_j / \sum_j c_{ij}$
or Multiplicative versions of these Models

① Gives More wt. to long cands, ② uses wtd mean of g_j's of cands.
Which is done in Org. Evoln? I think either would work — but which gives
the Best Signal/Noise ratio for detn. of "Synergized Concepts"?
Say A & B are 2 chemicals usable by Organism. Either alone is useless, but
.09 { Combination does ↑ G_i of cand. Is the ΔG given by the presence
.11 { of the 2 chemicals in an additive way (.02)? or is the effect diluted by
↑ many other genes are affective (.03)?
.09 – .11 Seems to be (the) "BIG Question". — There are (at least)
.13 2 Aspects of the Q: One is whether .02 or .03 is better
.14 Another is a special kind of Non-linear "Saturation" effect. As the
.15 amount of G gets larger than many symbols, it becomes harder to
↑ G using new symbols (ε "pairs").

In one calculation (.13, .03) we ↓ effective p_j by factor $\frac{1}{\sum_i c_{ij}}$ —
p_j is index of the other g_j values.

In "either" (.14) the ↓ depends on "links" $\frac{1}{\sum_j c_{ij} g_j}$ —
i.e. the other genes (symbols) affect the new gene only if they
.20 have longer Positive Exts ("ε")

It may be that the answers to .13 & .14 depend much on the extent genes have
used. That if one adopts (like .13) as ones' search scheme encourages
the discovery & keeping "Macrogenes", then either becomes a
"Self Confirming Hypoth" — So the Macro genes discovered will (be
ones) that fit the criterion used.

.27 So there are 3 items of Uncertainty (Questions): .02 vs .03 can
be decided (for a given Set of Genes), quickly, by trying out each of
the (slightly modified) formula for calculating the g_j's.
.30 The other effect between .13/.14 (non linearity −.15 – −.21) can be
dealt w. by 126.37 / 125.39 ff The effect of Subjecting
$\sum_j c_{ij} g_j$ or $\sum_j c_{ij} g_j / \sum_j c_{ij}$ to a "γ" or other N.L. x-fm.

can be obtained in 125.39 ff by subjecting G_i to the inverse x-fm,
(Not exactly, because our Error Criterion is Modified).

It would seem that the decision betw. the 3 above effects (.27)
would be largely an Empirical Matter — it would depend on
the Set of Genes being Evaluated (as well as the Set of Problems) → 128.01

re: 127.00 & Hwr, one may be able to Make Some Theoretical Analysis of what to expect. Looking At Specific Problems Might be very Useful. My "Coin" of r. if input Nuisance might be of interest. It might be well to include genes for AND, OR, NOT to get a much more complex Model.

Re: The 2 Multiplicative Forms Corresponding to 127.02, .03 :

By 127.02 form would go to 0 or ∞ w. ↑ in L depending on whether the g_i were > 1 or < 1. This is invert linear in 127.02 — whether $g_i > 0$ or < 0 determines behavior for large L.

→ Perhaps 1. Mult Model of 127.02-.03 can be obtained by simply using $\ln G_2$ instead of G_2. (which is covered by 127.30)

Another Approach to Theor. Analysis: See what each of the 3 Approaches says about Normal Mutation & Crossover techniques — How they "ideally" should be done: How to order the trials in a non-probabilistic way.

6/30/00

.17 Well, some apparently major Criticisms of 127.02, .03.

In 127.02 (sum) we let $g_1 = 3$, $g_2 = 2$, $g_3 = 1$, Term 1 is tied number of G_1 $S_1^{(\infty)}$ has most 0. If $g_1 = 0$, $g_2 = -1$, $g_3 = -2$, $S_1^{(\infty)}$ has most 0 (sum)

In 127.03 (mean): " If $g_1 = 3$, $g_2 = 2$, $g_3 = 1$

S_1 ; $S_1^{(2)}$; $S_1^{(3)}$ . . . $S_1^{(\infty)}$ all have 1 some of 3.

Using Multiplication Wr, in 127.02 (sum), If $g_1 = -1$, $g_2 = -2$, $g_3 = -3$,

order of G: S_1^{-1} ; $S_1^{-2} S_1^{-2}$ S_2^{-2}, $S_1^{(3)-3}$, $S_1 S_2^{-3}$, S_3^{-3}

This seems like a more natural ordering. We can subject true G to some X'form so it gets better.

We could do same Prob with 127.03, but in general in this case, G would be in sup or large & would depend only on the relative sizes of the various Symbols.

127.02 then can be modified to work for multiplicative g_i's, so have all g_j's < 1. For multiplicative g_i, I don't think $\sum g_i = 1$ is desirable.

Just to 127.02 but $g_i \to \ln g_i$ and disregard apparent Max formula

If $\sum g_i = 1$ Several Some g_i's can be very close to 1 (Very by g_i) (In $g_i < 0$, but (g_i) is small,)

Since g_i corresponds to $\ln g_i$, $\sum g_i = 1$ does not correspond to $\sum g_i = 1$.

So the "normalized" mult model does not completely correspond to the normed additive model, but the unnormed mult model pretty closely corresponds to the unnormed additive Model, exactly.

.36 Except for the other criterion. If we use $\sum$ sq. error this reflects differently in the additive & Mult Models (I think).

.38 i.e. $\sum_i \left(\prod_{j=1}^{K} g_j^{c_{ij}} - G_i \right)^2 = \text{error}$ is not the same as $\sum_i \left(\sum_{j=1}^{K} g_j c_{ij} - \ln g_i \right)^2 = \text{error}$.

I'm not so certain about -.36 -.38 !

I think the error criterion must be

.02 $\sum_i \left(\left(\sum_{j=1}^{K} c_{ij} \ln g_j \right) - \ln G_i \right)^2 = \text{error}.$

So here the error is below the log of the multiplicative approx.

Then (128.38%) error is below the multiplicative approx.

.02 is easy to solve & need cc for solving methods:

I don't know about (128.38%)'s cc for solving; but even if it's easily solvable all!

But anyway I think .02 may be closer to what I want.

It's nicer to get G_i into the proper functional form by applying some kind of distortion $x \to \ln x$ or $(G+\alpha)^\gamma$ or $G^\gamma + \alpha$ or whatever.

.10 I guess we want to map G onto $(-\infty, 0)$. As $G \to 0$, $\ln G \to -\infty$.

So if G is originally from 0 to $+\infty$; $\ln G$ is from $-\infty$ to $+\infty$.

If G is on $(0,1)$ then $\ln G$ maps it onto the desired form $-\infty, 0$.

So if G is on (a, b) first map (a, b) onto $0, 1$ then take $\ln$.

But there are ∞ ways to do the mapping. $G \to G^\gamma$ would multiply $\ln$ by γ.

.15 A very simple x form, yet probably very important.

.16 So far, the most reasonable model is unnormalized multiplicative. It is not similar, but not the same as using a p cost-like model (which has $\sum p_i = 1$)

Also, we do have to x form G so its attributes on $(-\infty, 0)$ & probably x form it further to get good fits. This fine tuning of G can be done by presumably several parameters at a time by Non-linear curfit, rather quickly.

i.e. $G \to f(\alpha, \beta, \gamma, G)$: we want to optimize α, β, γ. See 125.29 ff for the general idea: For each α, β, γ pt. I use to get a total error, I try a bunch of α, β, γ pts & get error on each, then I form a best fitting quadratic form from the data pts & pick the $\alpha\beta\gamma$ pt. that gives min. I may have to repeat this & move data in the region of the first min — but usually I don't need as much precision in this.)

For the Models of 127.02, 03. I had various criticisms of (28.17

(127.02 may be ok, if the c_{ij} are all <0)

Can I find any good criterion for fair-ness on the multiplicative model

or (.02) (which is perhaps the same).

Another reason for not using normalization in Multi: That finding a new symbol (or function) should be allowed to fit G estimate w.o. decreasing the partial χ^2 accuracy that were due to other symbols. — so we may want additive p_i w.o. normalization.

Ah! for "production" when considering the definition $A^* B = C$, we might just consider modifying only the g_i assoc. w. A & B & the newly introduced one for C. Computationally, this could be very cheap — much cheaper than recomputing

$$\sum_i \left(G_i \sum_{j=1}^{K} c_{ij} \ln p_j \right) = \min$$

(1) easier to compute

constraints $\sum p_i = 1$; $p_i \geq 0$.

6/26/00 Bolg.

Listing all subsums of a set of nos. in order of size (02)

" " " products " " " " " " " " " "

Doing it even when nos. to be multiplied are not normalized or are "super normal" (0).

all of the B_i's ! Using all operators, using n functions of k types w. b inputs (12)

.02 How to list all subsums of the set of nos. $[\sum a_i]$ in ~ order of size!

.04 Say $\sum 2^{-a_i} = \alpha$; Then find a constant $\beta \Rightarrow \sum 2^{-\beta a_i} \leq 1$

If the z_i are a finite set, this is always possl. setting $\beta \leq 0$

highest sum ≥ 1, setting $\beta = +\infty$ makes it sum $= 0$. So there must be

a β that makes it 1.

So we will order the [Basic] subsums of $\sum b_i$] ; $\sum 2^{-b_i} \leq 1$,

Assoc. each b_i its binary Huffman code.

Then For $n = 10$, say, list all binary strings of length 10. This can be uniquely decoded Huffman-wise into each of the b_i's/symbols. They represent all allowed strings of b_i's that sum to length < 10.

We can then get more subsums by using $n \to 15$, say.

(or even just 11), each.

.06 This technique can also be used to order sub products? subproducts of p_i's

.08 $\Rightarrow \sum p_i \geq 1$. ; we map the p_i into $p_i = z^{-a_i}$, then $\to p_i' = z^{-\beta a_i}$

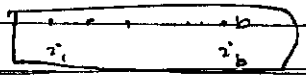
.18 so $\sum p_i' = \sum z^{-\beta a_i} \leq 1$, $\alpha \in c$. If α amounts to $p_i' = p_i^{\beta}$, → 131.19 → 135.01

.19 → 142.01 for details of Huffman

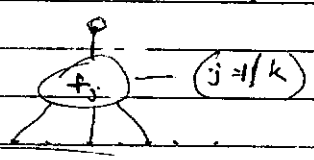
So where I am now! The unnormed product or unnormed sum formula Models of G as a funct of c_i and $B_{ern,i}$ I think I understand My better Number: (5) 3 essential ideas (1) Products are not normed (2) Sums are of negative nos. (3) It is usually essential to modify the G function $G \to f(G)$ (f monotonic ↑). — but this is not very difficult. (q.v. 129.16 - .26)

.25 To list all of the functions in solving by functions [types, 2nd Bernoulli] having 1 output 2 B inputs for minimally $n = 0$ connect $\bar{z}_i$ to 0 in any of b inputs

e.g. z_2 to 0 or z_b to 0, ect.



$n = 1$ for each functional type its output = 0 Its inputs come from z_1' to z_b' in all possl. ways.



for $n = 2$ we start with $n = 1$, but no connection to 0. We insert a new funct. with output 0. Its inputs can be $z_1' \ldots z_b'$ and the output of the n=1 functions.

So, given any n functional operator, to construct all possl. n+1 functional operators Premise. Let the inputs to each of k functional types be $z_1' \ldots z_b'$ and output of the n functional operators (or output to 1-1 other functionalities) Each its output be the n+1 functional operator output.

How many Pset operators are there, as a function of n, k, b ?
(no. functions; no. Pset types; no. inputs to operator)

(This ↑ very rapidly w. n, k, b.) I may need this info so as to compute p.c. of various operators. I think I obtained a form. for this problem a/o a ~ problem in Saarb. I didn't prove the form. correctly, but I should it to be true for many values of n, k, b.

The recursion formula: Say $\phi(n,k,b)$ is the no. of functions, and $c(j)$ $j=1|k$ is the no. of inputs for the j^{th} function type.

.11 $$\phi(n+1,k,b) = \phi(n,k,b)\cdot\left(\sum_{j=1}^{k}(b+n)^{c(j)}\right)$$ // b+n is no. of poss. inputs to each of the c(j) input points of the function.

Maybe I found it for $k=2$; $c(1)=c(2)=2$?

— But I found a problem quite diff. (analytically) — so I'm not at all certain as to what I found!

In practice, I wouldn't be using very large values of ϕ, so n & k will probably be very small.

.18 $$\phi(1,k,b) = \sum_{i=1}^{k} b^{c(j)}$$ So, say $\phi(0,k,b) \equiv 1$. ← The RHS is not literally true!

.187 Numeric / String / Boolean → See 219.13 on TYPES restrictions on b/o of ports → 136.21 on BASIC → 143.20 on SYMBOLICS

N=0! $\phi(0,k,b)$ is ... many ways, except as a specially defined value of ∅.

.19; 130.18 Hmm! 130.16-.18 suggests that one can use normalized products!!

— In fact, that it is necessary to do so than not! I think it's equivalent to $G_i \to \gamma \ln G_i$ (γ is same as β of 130.03-.18)

So the g_i's act just like pc's, and the D.R. over the candidates acts like a regular P.D. However, the g_i outputs have to be taken to γ power before they are to be used as approximations of G. → Branch → 133.10

So, what we do: we get G to (0,1) by suitable monotonic function (see 129.10-.15). Then we try to find a γ s.t. $\gamma \ln G$ can be well approximated by our stochastic language of diversity of the candidates. Also Note 129.16-.26 — on "shaping" G to get a good value of γ. Probably these 2 things can be done at same time using 129.16-.25 (& 125.29 ff)

.30 What it all implies is that I really need not concern myself with models of G in terms of the diversity of the candidates. All I have to do is find a stochastic (say, (it need not be a Bernoulli lamp or anything like it) that fits the $(\text{candidates}, G_i)$ data set, using suitable exponents to get G on the interval (0,1) then find an optimum G^{γ}

So that I can find a stochastic grammar that models it well!

.36 So this is closely related to the original SGA! → 132.09

• Re: "FINE TUNING G" : Here are 2 aspects :

1) finding good γ and fitting G into the (θ, l) range during various possible runs

2) Actually using a different "fitness function" : This can become important in INV problems (related to OZ problems). The fitness function can change markedly as one gets to different degrees of "closeness" to a final solution.

09:31.36 ! While in theory, I will not be limited to Horn Grammars: In fact, I will be mainly using them (cf Rissanen): partly because the fitting of γ: $\beta_i (i=1/k)$ params of 12.02 is relatively easy. In general, I don't know how to "fit" params. of a stoch grammar to data sets like [corp of Cands G_i]. My main experience is to fit grammars to a corpus, which is quite another thing. [Not Quite," The present problem is to fit the params of a grammar to the $[G_i]$ data — the grammar given.]

Well, I have not thought about this. Given a stochastic PSG (context free or context dependent), If we select a single ("most likely") parsing tree pc we get for G_i cands, will be some product of a certain set of pc_i's. This will be true for all data cands in the set. So for each (Cand$_i$, G_i) we will get an expression, $(\sum c_{ij} \ln p_i - \ln G_i)^2$ that we must try to minimize. Unfortunately, the normalization constants will be rather complicated. — 100.13 ff deals w. this (maybe!) — say 100.25 For each normed candidate, we have a different λ_i. Hmm, look at 100.13 ff. I think the notation is bad. Grammatical ideas are Good! I don't know how to apply one easily somewhere.

Actually, the normalizations may be quite simple! It may be possible to get all data involving one λ into the same/eqs. So that we have a few different λ's, we have a few different equations.

It may be necessary to use the "proper" error criterion to get these equations into the most easily possible form to solve.

Some possible forms of error: $\sum (G_i - g_i)^2$, $\sum (\ln G_i - \ln g_i)^2$, $\sum G_i (\ln G_i - \ln g_i)$ (Gamma - $\sum$ for $\ln g_i$ ...) || Maybe try $\sum p_i (\ln G_i - \ln g_i)$ — this divergence is apparently more complex, however.

.01 | 130.18 | Also, say we have a bunch of potential trials w. pc's p_i: ($\sum p_i < 1$)!
One way to do MC choice would be to arrange the p_i's end to end, or non-overlapping intervals on $(0,1)$. A random no. between 0 & 1 picks one of the intervals w. prob $\propto p_i$.

Another (maybe better) way: If Huffman code for p_i: This gives a binary tree > decoding all p_i's. Start at root & make random choices w. pc's .5, .5: we end up in proper code w. proper frequency. ≠

This is a very efficient use of a random binary string requiring no search. In .01-.04, we waste ½ randomness of each new bit (between 0 & 1)

.10 | 131.24 | However, if we do not normalize our Multiplicative Model — ~~does this~~ is this not equiv. to automatically fitting in an optimum γ?

Using error $\sum G_i \ln g_i$: $g_i = \prod_m p_j^{c_{ij}}$

.19 $= \sum_i G_i \sum_{j=1}^{n} c_{ij} \ln p_j$ ∴

.20 $\frac{\partial .19}{\partial p_j} = \sum_i G_i c_{ij}/p_j = 0$ so $p_j = \sum_i G_i c_{ij}$ ← No! Very Complex!

γ is already fitted in.

We can further optimize $\{G_i\}$ however, by various xforms. We try various xforms of G like 129.16 – .26

Woops! .20 is right on but it doesn't give (.20 R)! It gives $p_j = \infty$, → see .31

We do need some kind of Normalization cond. If we used $\sum p_i = \gamma$

Then $\frac{\partial (.19 + \lambda \sum p_i)}{\partial p_j} = \left(\sum G_i c_{ij}\right)/p_j + \lambda = 0$

.22 We get $p_j = \gamma \left(\sum G_i c_{ij}\right) / \sum_j \sum_i c_{ij} G_i$

Could I simultly optimize γ & $\{p_j\}$? — rather than doing sequentially as 129.16-.26.

.19 | 129.16-.26.

Substituting .22 back in .19 & solving for γ that gives min:

$$\sum G_i \sum_j c_{ij} \left(\ln \gamma + \ln \sum_{i'} G_{i'} c_{ij'} - \ln \sum_{i'} \sum_{j'} G_{i'} c_{i'j'} \right)$$

$\sum \sum G_i c_{ij} \ln \gamma = \min$: $\gamma = 0$. Look like BWG!

WOOPS!

.31 The condition wasn't $\sum p_i = \gamma$ but $\sum p_i^{\gamma} = 1$ ∴ All wrong!

.32 Done this simult optzn. looks difficult/impossible! try substituting back in .19!

try $G^{\frac{1}{\gamma}}$? Unclear as to correctness! We may have to use

$$\text{error} = \sum G_i (\ln g_i - \ln g_{\text{est}i})$$

N.B. The usual way GA deals w. $G \to p$ is to use normal: $\to p_i = \frac{G_i}{\sum G_i}$

This always gives too large a value (by a constant factor), because $\sum G_i$ doesn't include all of the counts. However, may be only using it for relative probability, so it makes no difference.

.01 On t. other hand $G \to G^\gamma$ is a serious change in G.

If we use G_i^γ instead of G_i we can get error; try various γ's

.02 Best for best: Soln. is simple in terms of finding optimum $p_i = \frac{\sum_j G_{ij}^\gamma c_{ij}}{\sum_i \sum_j G_{ij}^\gamma c_{ij}}$

We just try for 3 values of γ in s.v.e for min, Then use best 3 values. Then use for to compute min & find it empirically, best. — Should converge. Fast.

.06 Trouble is, when we change $G \to G^\gamma$, the meaning of t. error criterion Changes! E.g. to give an extreme case: if $\gamma = 0$ or $-\infty$, G becomes constant & very easy to predict. If $\gamma = 0$, $G = 1$. — On t. other hand, if G allways ≥ 1 $g_i = \prod_j p_j^{c_{ij}}$ can never be 1, because p_j is not 1: Concretely,

No! (one $p_j = 1$ & rest $= 0$, but Then only cands containing that p_j whould give $g_i > 0$; all others would give $g_i = 0$. $\sum_j G_i (\ln p_{ij}) c_{ij}$ would be $-\infty$ for any cand that had any pc other than $p_j = 0$. — So if $\gamma = 0$, t. fit would be poor.]

← So This Counter Example is N.G.

.15 → I am uncertain about the effect of G^γ on t. "error Criterion"

.16 Also, in terms of Pitting the stack grammar $G \to \alpha G$ is also very impt! —

It leaves t. computed p_i values (.02) invariant, hmm.

so how does $\sum G_i (\ln g_i - \ln G_i)$ change w. $G_i \to \alpha G_i$: g_i is invariant.

$\sum \alpha G_i \ln g_i - \sum \alpha G_i (\ln G_i + \ln \alpha)$

.20 $= \alpha \sum_i G_i (\ln g_i - \ln G_i) - \alpha \ln \alpha \sum G_i$

$.20 = \alpha A - \alpha (\ln \alpha) B$

$\frac{\partial .20}{\partial \alpha} = A - B (1 + \ln \alpha) = 0$

Max: $(1 + \ln \alpha) = \frac{A}{B}$; $\ln \alpha = \frac{A}{B} - 1$; $\alpha = \exp\left(\frac{A}{B} - 1\right)$

$A = \sum G_i (\ln g_i - \ln G_i)$ $B = \sum G_i$

.25 $\alpha = \exp\left(\frac{\sum G_i (\ln \frac{g_i}{G_i} - 1)}{\sum G_i} \right)$.

So we have to do t. α & runs of G_i, at least — This leaves $[p_i]$ invariant.

Next, we try to fit a good $G \to G^\gamma$! But before we measure t. in error, we have to compute a new α. via .25.

[Note, t. operations $G \to G^\gamma$: $G \to \alpha G$ do not commute — unless $\alpha = \alpha^\gamma$. (for what γ?)]

It would seem that we would have to optimize γ & α simultly; but

This is not difficult → Try 3 values of γ, say $-.5, 1, 2$, but with each γ, obtain t. assoc. α via .25.

Then get t. error for each of these γ, α pairs: Extrapolate γ of (estimated) minimum error;

.33 Calculate $[g_i]$ & for that γ & compute α via .25; Then, compute true error;

Using Best 3 γ, α values this far, extrapolate γ of with expected error

→ loop to .33.

T. Perugg. should moreover keep work. $G \to \alpha G$ & $G \to G^\gamma$ are perhaps t. most impt kinds of X Runs we can do on G, — after G is in $(0, 1)$.

7/1/00 Bulp.

135

6/7 → 9)

7/1 → 135

2 d; 4 app.

6/11 → 2 ppd.

O.K.: Say we more or less have a (ALP grammar model) for G.

How do we best do GA? Mutations seem to be distinct from Crossover

W.r.t. analysis Methods: For Mutation, we don't need a grammar: We just have a

set of Mutations that give an expected D.F. of ΔF. This D.F. can be

empirically determined. Seems that best approach would be a first —

Mutate all of the G cards only, perhaps w. enuf "space" between G.

Cards so that they are rather uncorrelated when one Mutates them.

We may have to use STEIN to obtain d.f.s of various Mutations(?)

However, if the D.F. is very broad, we need not be very efficient! —

I.e. if D.F. has width 10, then Doing the 10 of the G

values of Cards would be O.K.

Or we just keep mutating the top cards to lower & lower G

levels, until we produce enuf new, by G values.

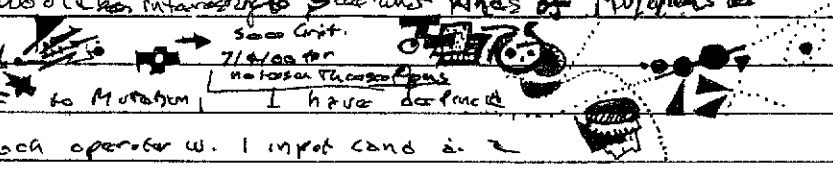
→ Another Q is: How many mutations to do one particular of the G cards?

That never would seem to be much value in Non-Elitism.

An interesting Q: D Fogel claims a.v.g. checkers player uses

GA & Mutations only! I.e. would be interesting to see what kinds of Mutations

he used & other details.



In Comparing "Crossover" to Mutation, I have decided

General "Mutation" as a search operator w. 1 input card & 2

diff. or output cards.

"Crossover" = Mutation from 2 input cards.

In Koza's presentation, his Mutations less General than I've indicated; On the other

hand, "Mutation" is being interpreted in a broader & broader sense.

In Both Cases, because SSZ is small, the D.F. depends heavily on the APRIPD.

Both can be (perhaps MUCH) improved by periodically updating the APRIPD,

by recalibrating the weights of various subroutines, based on experience in

the corpus.

BN NB The "Normalizer" I've been using for multiplicative models

& that for "Additive Models" are quite distinct! In the additive model,

"pc" ≈ $\frac{\sum_j c_{rj} B_j}{\sum_j c_{rj}}$; In the Mult model $\equiv \prod p_i = 1$; where p_i is the probability

of a Symbol S_i appearing at any particular pc.

In the additive model, pcr is the wtd mean of the B_j of a card.

Can I put a spreadsheet to do "Functional pcms?"

i.e. some of the w_{ij} will be $\neq 1$

$\sum_j w_{ij} = 1$

N.B. It doesn't pay much to do Trials in pc order (v.s. MC or Carlo) unless

there is a tendency of non-uniformity in/ pc of cards.

predictable

□ I assume "arity of a subtree" is no of inputs it has.

.01 Thots on Crossover & Mutation: In Mutation we are interested in Amount of Non Greed, on which In Sim Anneal, this is adjustable as "Temperature". Degree of Non Greed, is measured by ⓐ how low ΔG is after one pass up, ⓑ how long a seq. of mutations one is willing to evaluate before giving up. "ok" result is nearly.

In Crossover, the equivalent mutations obtained have been calibrated wrt p_{S_i} and the p_{C_i} wts are continuously updated — since they are of utility in the other cands. Mutation uses constant wts for its mutations — tho Some Mutation schemes may update the utilities of the mutations. (This last could be done via X window or X windo)

Mutation could devise "srtns". One could write for it to take any subtree and "srtns" substitute it for a terminal or use it to replace another subtree of same "grity". "Arity".

Def I think a BIG Question is: If I had a reasonable O.K. model of G_i as a function could dcrn: Should I do trials in order of expected G_i, [Or, if I have μ_{G_i} & $(\sigma_{G_i}^2)$: say the cands w. most prob of succeeding (present & Gmax).] This is a determinism trick. — But it seems very "Elitist" Another objection is that the Cond dcrn "most" may be too small, & wasteful — i.e. close cands will found to have some G. This later Suggests that some randomness might be useful. NOTE hvr: interesting set of [136.21 — 137.27] The "spacing" seems large! → 152.13 to ~160.40 Has a sequence that sort of solves this problem.

.21 : 136.18 How to construct functions in Basic: See 13p/25 — 13(.13) for refreshers.

Say X_1 (≡ X(1)), X_2, X_3 are inputs.

$X_4 = F_1(X_i, X_j)$ [i, j ≤ 3]

.24 $X_5 = F_2(X_k, X_\ell)$ [k, ℓ ≤ 4]

⋮

$X_{n+1} = F_n(X_m, X_r)$ [m, r ≤ 2+n]

X_{n+1} = output!

This covers all possible operators, if we only allow functions of 2 args ea. The trouble is I don't know how to vary the functions $F_1 \ldots F_n$ in Basic.

Is there a way to refer to srtns? There is a "Select case" function to choose srtns.

So Inq. 24 could be implemented as

narrowed 8, steps functions types available

$Z = X_5$; $X = X_6$; $Y = X_8$ Select Case (7): GOTO 100 | Z = F(X, Y)

replaced by 7 in later 24 so i: γ_i (i=1|8)

100 Case 1

Case 2

Case 7: Z = (sin X + cos Y)/(X+Y): return.

So the basic plan for any/depth arby function:

X1, X2, X3 are inputs; X6 is output.

~~Z = X4 ; X = X(i) ; Y = X(j) :~~ Select Case R(1) : GoSub 100

Z = X5 ; X = X(m) ; Y = (n) " " R(2) " "

Z = X6 ; X = X(p) ; Y = (q) " " R(3) " "

100 case 1 : Z = X^2 + Y^2 : return

2 Z = Sin X + Cos Y : return

3 Z = X Y : return.

(maybe "end select" on this line then "return")

It is illegal to separate "Select case R" from case 1, etc. by the latter being in a sub. Then just do W = R and in sub 1 do Select case W.

If i, j, k, ... r are "selected" arguments in a record w. proper rules, then the result is a legal function.

It is possible in this scheme to have functions w. various nos. of args.

0.9. So now I have to find a way to Mutate / "crossover" these functions. Each function corresponds to a Node in the tree.

Any function is represented as a set of integers in the X array.

Next strategy section consists of X(i) : for i = 1, 2, 3 : inputs,

for i = > 3, X(i) < i is probly good enough. So for depth n, w. b inputs,

b·(b+1)···(b+n) different X(-) array entries. = $\frac{(b+n)!}{(b-1)!}$

c^n possible functional configs; if there are c functions typed.

$\sum_{i=1}^{n} c^n \cdot \frac{(b+n)!}{(b-i)!}$ distinct functions.

Say b = 3, n = 3, c = 5 $3 \cdot 4 \cdot 5 \cdot 6 \cdot 5^3 = 9{,}500$ only

b = 5, n = 4 c = 5 $5 \cdot 6 \cdot 7 \cdot 8 \cdot 5^4 = 1.05$ M

.29 If we represent second arg to be not the first arg. we gain some, but not very much.

Anyway, it would be easy to pgm exhaustive search: For 2 computers in ||

Consider (each 2 functions; say there are 3 possible combinations; Divide them up

.32 between Y. 2 computers, in way, proportional to their speeds. → (139.01)

.33 ? → On the Advantage of assigning pc's to "Parts": This may have to do with

the fact that almost all of the vol. of a n dim hypersphere is within $\frac{1}{n}$ of the outside surface.

.33 The expected no of trials is $\sum_i i\, p_i$. Maybe not — Maybe $\sum_i \left(p_i \prod_{j=1}^{i-1} (1-p_j)\right)$

.34 ~~$\sum_i i\, p_i \, e^{\sum_{j=1}^{i-1} p_j}$~~

.35 Corridor $p_i = \frac{1}{n}$ (max i = n) : $\sum_i \frac{i}{n} \prod_{j=1}^{i-1} (1-n)^j \quad \frac{1}{n} \sum_i i^2 (1-n)^{i-1}$ I worked this out recently.

It's .34 the derivative of any Poly? (like in .35) Maybe not derivative but "first difference"?

Take first difference of (.33 B) $\sum_i i \frac{p_i}{\prod_{j=1}^{i-1} (1-p_j)}$

"$\frac{d}{dp}$" $\prod_{j=1}^{i} (1-p_j)$ —?

r solomonoff
8 Fifth Ave
http://world.std.com/~rjs
/pubs.html

(122.01 121.29 117.8)

(115.14 ≈ 116.35) (117.2)

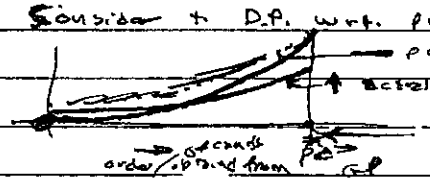
Anyway, the point of the search is not minimum time to find a particular cand. If this were so, GA would not work! (I wrote about this in ... it ....) Anyway, the reason of the problem was something "like":

—— If we do a Mt. Carlo search w. trial of "pc". we will get a certain D.F. of G values. This D.F. of G values will be 'better' (how much better?) than if we used a flat d.f. for Mt. Carlo srch.

→ Viewed in this way: How much better is deterministic, properly ordered srch, than Mt. Carlo? May depend much on how good the "probs of G" model is.

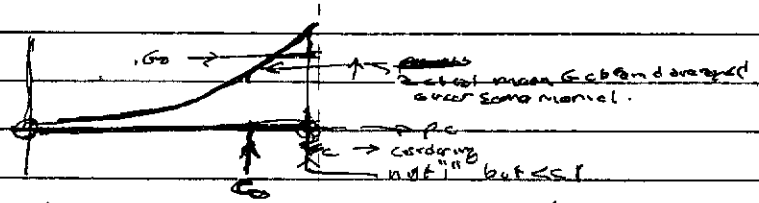
My Main Present Problem: To estimate how much Better my system is, than GT-GP — or any of the Monte Carlo srchs. More Generally, to understand just how my System is expected to differ from them & if it does it work ("too short") to Understand Why!

Consider the D.F. wrt. pc's obtained by GP:



These 2 curves result in a certain P.D. of G obtained.

Using random any deterministic srch:



The D.F. I'd obtain would depend on how far down in the search I'd soften. If I stopped from down to pc = Co, I'd have a mean of ~ Go, but a small σ², so, considering variance, I'd maybe not have such a good chance of a low G as low by G conds. The lower Co goes & the lower Go goes, & σ_G^2 ↑ but σ² ↑ so better chance of low G conds.

In GP: (.15 L) As we ↑ σ², the D.F. (contrary) stays constant (but low) & the σ² ↑

Using the systems, study the expected no. of conds w. G > Gmax, as a function of no. of conds evaluated (= "srch time")

Or, a simpler way to look at it, Compare the probs of a cand < Gmax per trial cand. In GP this probs is constant. In My system it starts out >> that of GP, then ↓. It may take a long time before this probs < that of GP.

→ GP might get my probs of G by ~~Dividing~~ Subtracting Beta computed p_i to p_i^{γ} ($\gamma > 1$). This is a kind of Criticism: Make γ as large as possible, until we begin to waste too much time by duplicating trials.

(SPEC)

.01: 13732 : On using several computers in ll for search. If we didn't w. 1 machine & we had distinct pc's for each "function", we need to order the sequences in pc. order; This can be done by assigning a Huffman code to each function. Then to search for all strings w. $pc > 2^{-\ell}$; Consider all 2^ℓ ℓ bit codes & express them as sequences of the proper Huffman symbols. We then do $\ell \to \ell+1$ & do the search again. We can know when a code has been tried before, because it will have been decided in $\le \ell$ bits. Only cands w. code lengths of exactly ℓ bits are considered!

IN this technique ~~we~~ may be expensive to keep track of cands that have been modified, But it will be OK. if evaln. of G, is very expensive (as by use of SPICE!).

Anyways if we have α computers w. distinct speeds, in ll, we assign a fraction of trials to each : the no. of trials assigned is $\propto$ speed. We can do this by considering the first ℓ bits of the codes. There are $1024 = 2^{10}$ such sequences. If the i^{th} machine has relative speed s_i ($\sum s_i = 1$) then we assign the 10 bit codes below. $0 \le 1024 \cdot s_1$ to the first machine, we assign the " " " " $1224 s_1 + 1 \le 1024(s_1+s_2)$ to the 2^{nd} machine, etc. So each machine has a subset 10 bit codes that it trials and starts with, and exhaustively considers all continuations of these 10 bit codes.

We want to retain in Memory, all cands in top 1000, or whatever. So we initially use the machines to send to central computer or just put in their own RAM, defns of all cands w. $G > c$ certain G_0.

After the run, we collect all the G cands & then review our Model of G as a func of cand defns. We will, ideally, notice useful sub trees, etc.

→ (146.27 on PanPiumIII Dell proc.
→ (147.01 on RanNapTuman on cabsmall computers

Next on agenda: (1) Review of results to date (2) Analysis of normal GP crossover, ADF & ADR vocs, & recursive ADRS in terms of my SGA model. (3) Some analysis of Mutation

7/3/00 Bulg REV

Rows 141
148

140

Mark Size!
122.17

This is a Review of recent Results on Analysis of [redacted] GA/GP in terms of ACP: Many using SGA model.

Some ideas:

1) Mechanics of sorting in PC order; for single or Many (1 processors. 137.29 ~.32 — 139.01 ~.22 — 128.90 ff and Comparison of Models

2) How to list Function forms [redacted] 130.25 – 131.18 — Details of Mathematica.

3) How to list Bimoulli subsets in PC order 130.02, 131.19, 133.01, 142.01

Also, This may be relevant to listing all subsums of subsets of a finite (?) set of reals.

4) How to Get BASIC to Implement Nested functions & evaluate & set of them: 136.21 – 137.52 — also sum Φ: 140.09

5) (Serious) Reasons why GP/GA using Mt-Carlo meMods should NOT work at all 115.14 – 116.35

6) How [redacted] GP/GA probably actually does work & a comparison of speed to that of SGA: 117.21 – .31; 122.29 – .34; 122.01 ... 138.01 ÷ .40

13 7) Some models of G as a function of Cand. dcrm: ⇒ (120.01 – 129.40)

3 Additive models: G_i is a [redacted] forms of "Parts" & Multiplicative models

$G_i \approx \sum_j C_{ij} g_j$ "Symbols" $G_i \approx \prod_j g_j^{C_{ij}}$: $\sum g_j = 1$; g_i [redacted] > 0.

$g_i < \phi$ for additive models

$G_i \approx \sum_j C_{ij} g_j / \sum_j C_{ij}$

$G_i \approx \sum_j C_{ij} g_j / \sum_i \sum_j C_{ij} g_j$ must

In both multe Additive Models, C_{ij} are case counts of symbol j in i^{th} Cand.

$G_i \to \alpha$ Gk [redacted] either G_i's on $(0,1)$

In all cases, might modify G_i w. various xforms so that Resc models can fit well.

Argos, Testing! Variety Models 119.28 ± .90 V.S. Additive models (no negative g_i not yet done).

120.01 – 129.40 Develops a multiplicative model. There are 2 possible criteria: Σ score 129.01 – .02 (129.01 R is easier to evaluate, but

I should try both ways to see which gives best sense to problems.

I probably need a special review of 120.01 – 129.40 Also Note 135.31 – 134.40 Dozen of GAGP,

141.01

7/3/00 Bulg [Rev] 140 148 159 162

Proto-ranilea

a): 148.40 [R.3 is a proto Review of 129.01 - 129.40 ! Hes to do w. Model selection: corrections, which one(s) worked, whichever didn't.

1) We can use either GORC:

$\sum_i (G_i - g_i)^2 = \min$! (a) $\sum_i G_i \ln g_i = \min$. [$g_i$ is our approx to G_i] (b)

In (b), the actual Gorc = $\sum_i G_i(\ln g_i - \ln G_i)$ which is zero when all $G_i = g_i$

(b) is good for multiplicative models, (129.02 R)

If $g_i = \prod_{j=1}^{k} p_j^{c_{ij}}$: $\ln g_i = \sum_{j=1}^{k} c_{ij} \ln p_j$. p_j are params of "symbols".

.11 $\sum_i \left(\sum_{j=1}^{k} c_{ij} \ln p_j \right) - \ln G_i)^2$ is an O.K Gorc. ; [w.o $\sum p_j = 1$,]

Solved in 125.02 → .06. But $G_i' = G_i \cdot$ is fine:

$\sum_{\ell=1}^{k} p_\ell \sum_i c_{ij} c_{i\ell} = \sum_i c_{ij} G_i$ $j = 1|k$ $\sum p_i$ need not = 1.

← k eqns in k unknowns

This makes $G \to G^\gamma$ unnecessary! Increasing all p_i by $\to p_j^\gamma$ or $\ln p_j \to \gamma \ln p_j$ does it & eq. (1) could take care of the γ.

$G \to \alpha G$ means $\ln G_i \to \ln G + \beta$: So may be include accn/con in eq. (11)

This can be done w.o dummy p_0 that occurs in all cases: $C_{i,0} \equiv 1$

$p_0 = \alpha$

So, very neat! After one solves eqns for $[p_i]$, one can determine the effective γ value to subject G to, so that $\sum p_i = 1$, and G will be more like a pdf!

We want $\gamma \Rightarrow \sum p_i^\gamma = 1$. ; we'll do $G \to G^\gamma$ (or $G \to G^{\gamma}$?)

This is easy to solve by successive approxns. — Maybe quadratic conv. to closest pt. to zero. Newton involves $\frac{d}{d\gamma}$: difficult to do.

$\sum e^{\gamma \ln p_i} = \sum \ln p_i \cdot p_i^\gamma$ = derivative of $\sum p_i^\gamma$

.29 A simpler Gorc is (129.01 R.) $\sum G_i \ln g_i = \sum G_i \sum_{j=1}^{k} c_{ij} \ln p_j = \min$ $\sum p_i = 1$

(Pro $\sum p_i = 1$ (for any other constant than 1) is nasty, a Pyrrhonian min is $p_j = 0$ for all j.

To put a good fit, it's nasty to effectively Modify G by $G \to \alpha G^\gamma$.

Anyway: eq. .29 has soln

$p_j = \frac{\sum_i c_{ij} G_i'}{\sum_i \sum_j c_{ij} G_i'}$ We can multiply p_j by any constant α, to change norm to α.

154.15 on $G \to \alpha G$

We can find γ value $G \to G^\gamma \Rightarrow$ Gorc is minimal — but I'm not sure that it is O.K., since the target criterion changes as $G \to G^\gamma$ → (134.06 → 15)

Rev → 148.01

A Good Soln to "TRIALS IN PC ORDER"

Nominally for Bernoulli Seqns in order of pc.
Possibly Adaptable to More Complex Models.

.01 : 130.19 On Huffman trees & Guided Searches:

Let X be a binary no. starting w. X=0:

.03 (loop) Use X to guide path thru Huffman tree, starting at Root. When we get to a leaf, write down the corresp. symbol & return to Root. Use X to continue guiding the search. One ends up w. a seq. of symbols, one for each X. If we've come to last bit of X on a leaf, then (Evaluate/execute) that corresp. seq. of symbols. If not, then X = X+1; loop to .03.

Heidegin: We never repeat an Execution

If we are doing a search w. N computers in ||, then each machine has a set of binary strings $C_{i,j}$ which it tests as "X" values. For a given Y_i will be equiv. to a seq. of symbols plus a printing Huffman tree. For each machine will start out at the corresp. pt. in the tree & use X to guide its subsequent behavior (X=0 | 1).

When a new fact is completed $Y_i \to Y_{i+1}$ & we use this same X for Y_{i+1}; when i+1 gets to its limit, then $i \to 0$ & $X \to X+1$, & we start out again.

small Trouble is: Most of the runs end in the middle of the tree & are discarded.

(Fraction of useful runs) / (total no. of runs) = (no. of leaves in the tree) / (No. nodes in N tree)

~ 1, if depth of tree (?) The ratio probably is an insignificant part of TM's CC, & may be large if CC of Evaluation is very small.

.23 (SN) In SGA, we use a stock Grammar to get a β_i assoc. w. the i-th seq. of Cand's. What we really need is the U.R. of β_i — And in the SGA approach, I think $\beta_i = \mu_i$ & we have a common $Q^{(2)}$ for all of the cands. This is clearly impossible — but I don't immediately see how!

.27 (SN) I had been uncertain about how to integrate ideas about coding cost w. the ideas of methods of finding Cand descr. (K_i) w. high ξ values:

Actually, it's very simple! We want $P(G_j, X_i)$ & Pd. We can think of it as a stock phenomenon w. X_i as input & the Pd of G_j as output.

There are 2 variations of this induction problem:

(a) We are allowed to use regularities in the data set $[X_i]$ to help obtain "short codes". (b) The info of (a) is not available. — we assume $[X_i]$ were drawn at random.

The problem of Searching over the resultant $P(G_j, X_i)$ function for X_i of Max G_j is Discussed in 155.30 – 156.05 & is a part of my analysis of OZ problems in MCT.

It's not clear how large is restricted to making $P(G_j, K_i)$ models in which X_i of hi G_j are easy to find. → (430)

7/5/00 Bulg.

I think it would be easy for CFG's: (No) It's easy for Derived langs
Well, it's not too bad for "Non-derived" CFG's & diff for non-derived langs!
But for derived CFG's, — very diff.
I'm not sure basing "non-derived" helps!
T. didn't. $\log_{12}$ still holds! — The "Lookahead" may help!

94.23-95.07

p.125 ff Functions; Syntactic

.01: 142.90 On SGA: One of ideas was selecting a G function or PC(G) function in which it was easy to find t. rand. (or x t. conds) of hyps & G.
For Bernoulli langs this is easy, using (Huffman Codes) or any other methods) → Lo 142.01, 130.02, (Also note 137.33!?)

SGA: Is counting Multinomial! Depends on the rules & terminology!

94.23-95.07

.05: 95.07 For CFG's it is not nearly so easy!: If even Derivation can legally end at any point: Then we can use some reasonable approx.
(I think this is called something like a "derived CFG" → 152.01
For CFA's in which one has a specific set of terminals (leaves → 152.01

.09 The problem is much more diff.: At any branch (say t. pc's are .3, .7) the .7 branch can have a maximum pc sub branch that is arbitrarily low. I.e. the .7 branch can split into an infinite number of leaves of ≥ pc

.12 $\frac{1}{n}$ pc of each (where n is arbitrarily large)! If one has a lower bound on the relative pc of a terminal (e.g. an upper bound on n, it's not so bad! — but it can still be rather hard diff. to guess at lowest pc branches!

SO: T. Moral is: To try (in SGA) to use CFG's for t. approx. Restricted "Derived" languages. (Recursive Bern lang. is of this sort) ← No this doesn't really fix it! (see Note on top of page) → 144.01

Note: Wrong Note on top of p. Missed t. pt.!
It's easy for derived langs diff for non derived langs.

.18 .20: 131.48 On t. effect of symmetries in t. sets of usable functions.
If a function has symmetries, like $f(x,y,z) = f(y,x,z)$
Then t. no. of unique combinations in 131.11 has to be reduced.
Each term $(b+n)^{C(j)}$ has to be multiplied by $S(j)$ the recip of the "symmetry no.": for a pair of commuting args, $S = \frac{1}{2}$
For 3 commuting args ($f(x,y,z)$ is indep of order of x,y,z)
$S = (3!)^{-1}$; for $f(x,y,z,w)$ w. x & y commuting & z & w commuting
$S = \frac{1}{2} \cdot \frac{1}{2}$; etc.: Can we use analogs to Deal w.

Type restrictions on I/O of functions? This is impl. in 219.13

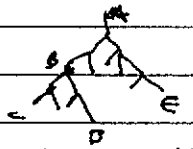
.29 :

from 143.18

.01 : for "Derived langs (not "prefix cons") $\sum p_i > 1$ usually ("Non-" terminals are also terminals)
for nonderivable langs (terminals $\neq$ Non-terminals) $\sum p_i \leq 1$

[I'll have to re-think much of my/recent ideas on CFG's in view of this!

T. Harris: it's like a "zoo" of "prefix sets" The analog of "prefix" is that one tree is an extension of another.



The tree ABCDE is an extension of the tree ABE; the extension being tree CD.

T. above is very impt. in Chomsky & Miller's of 140.13ff; I don't think I was completely aware of this effect when I was considering gossip models.

In general, the possy. that $\sum G_i$ = ∞ is quite reasonable.
Only in certain cases might $\sum G_i$ be bounded.

[One possy. Normalize as assoc. w G_i: The probly that Cond'l has to branch G a value from a branch of $CC = CC_0$.

Anyway, Expressions for μ_G of r.v. G d.f. assoc. w. a Cond'l — in general, I don't expect this to be normalizable: E.g. probly $\sum (\mu_{G_i})_j = \infty$.

The Question is: Can Real "Non-prefix" forms of CFG's give a pc distribution that could be any Renyi like a G d.f.?

A simple additive model like this is GIVEN (145.06)

.19 Doing an additive: $g_i = \sum c_{ij} p_j$ (write vertebrae): $\sum p_i$ not restricted. "Skew"

Crame is $\sum_i \left(g_i - \sum_j c_{ij} p_j\right)^2 = \min$ p_j ($\neq pc$) is assoc. j-th "feature"

See [1250.1-06] so $\sum_k p_k \left(\sum_i c_{ij} c_{ik}\right) = \sum_{i,j} c_{ij} G_i$ $i,j = 1,\dots,k$ "Symbol"

Keeps it nuts.

See 12 f. 27 for details, & some "rationale"; but somewhat difficult; I'm sorry G_i not $\frac{G_i}{p_{ci}}$.

There is also the Q of whether I want $g_i \propto \sum c_{ij} p_j / \sum c_{ij}$ i.e.

.25 i.e. g_i is to be with mean of those p_j.

.26 Soln G_i is $\propto \sum c_{ij} p_j / \sum_j c_{ij}$: $\sum_i \left(\sum_j c_{ij} G_j - \sum_j c_{ij} G_i\right)^2 = \min$.

.27 [illegible]

$\frac{1}{2} \frac{\partial}{\partial g_i^2} (.26\,R) = \sum_i c_{ij} \left(\sum_j c_{ij} p_j - \sum_j c_{ij} G_i\right)$

$$\sum_j p_j \left(\sum_i c_{ik} c_{ij}\right) = \sum_i \sum_j c_{ik} c_{ij} G_i$$

So either way is possl.

.34 AND we can decide whether a particular set of subtrees or other characteristics of Renyi assoc p_j are a "Good description of the corpus", if the resultant μ_G is such to pay for the description of the model!

.34 is true for either pure additive (.15 ff) or w.t.d. mean (.25 ff)

SO FINE! —> (But see 145.01 for OBJECTION!) —> (145.01

7/8/00 Bulg.

On A BADNESS of G^γ as an x.fun (.32)

(The not so difficult to fix!)(.06 – .32) R has reasonable "soln." ("f.x")

.01 A kind of BUG in the Range, Incr. We are using Range Models for a correction of G to the data. We then pick pts. of expected G for investigation known. When we get one of these we calculate a new Set of p_j values (but we may first find some new features (Sub-areas).)

Would this Give → "Self Corr. Hypoth" effects? (I don't immediately See Why!)

.06 → The other (OLDE usual) Bug: In the simple additive model the expected Choice for-comparison' card of [bypass expected G] is one w. 00 repetitions of the Basic "feature" ("sub tree" or whatever), Four. with mean, 1.1, 3 sigma, using only the single basic features. [7/23/00: Hvr, there was the downward Modal Short codes have hyperbs.]

A poss'l. approach: In linear regression we can have only Rank no. of coeffs, but we pay each no. of coeffs, there is an assoc. "compression factor" — a certain no. of Coeffs give "Best" compression.

For a given C_{ij}, there are many C_{ij}'s corresponding.

A Poss'l Approach: do a distrib determination of p_j for each value of $\sum_{i,j} C_{ij}$

(In the case of Features, Etc. This is the total no. of "functions" used.

Then, if a certain addition of a certain function involves r choices between Then we need log of r bits to specify that function.

However, we usually think of the Cond. dcrn. as being "Free"!

.19 As we ↑ "dcrn. size" (N) of Cond, the no. of Conds ↑ exponentially. So observing only one of the Conds of (N) (size dcrn) is of not great importance — Just

.21 G observations is given by 2^{-N} only — since there are 2^N of them. (2x)

.22 KOZA Volume I may give some usable data on the G d.f.

Some info: Fig 7.10 p 192.

.24; .26 → So perhaps the mean G in expected G w. "N" is reasonable.

I really need to list all of the models I've tried along w. their good & bad pts.

.19 could be a good rationale for multiplicative p_j exponentiation: or The idea of

12.4.27 w. an additive model, Mult by the pc of the cond.

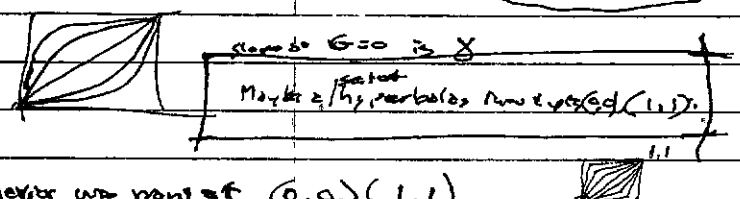
The idea of 12.4.27 of multiplying & dividing G_i by p_{ci} seems backwards!: i.e. it would seem that for large N, we'd want to mult. G_i by 2^{-N} because each data is so "rare" an representation of conds of that "size"

.32 On the γ exponent for G: G^γ has an apparently bad behavior near G=0.

for $\gamma = 1$ its ✓ for $\gamma < 1$ its ✓ for $\gamma > 1$, its ✓

The slope at G=0 is discontinuous in γ.

A fact that would seem better:



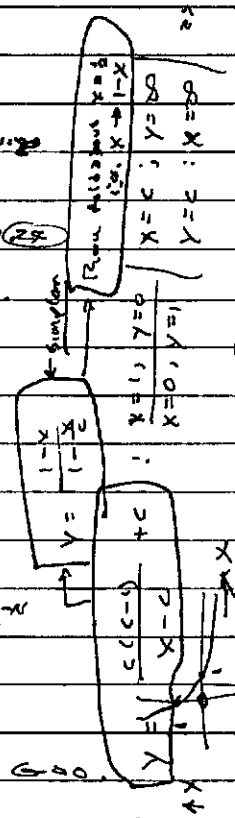
$\gamma x \cdot (1 - \alpha x)$

Well, it will depend on what sort of behavior we want at (0,0),(1,1)

G^γ is o.k. at (1,1): perhaps . . .

$\frac{d}{dx} x^\gamma = \gamma x^{\gamma-1} =$ ∞ or 0 i.e. $\gamma < 1$ or $\gamma > 1$

On the other hand G^γ is fine for a approx of piece-wise linear approx. may be good enough



7/8/00 (H.W.) PENTIUM III: From (6.2200) (.27)

Bulg. U.S. MMX

$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \rightarrow \frac{e^{2x}-1}{e^{2x}+1} = \frac{e^{2x}}{e^{2x}+1} - \frac{1}{e^{2x}+1}$

$f(x) - 1 = \frac{e^{2x}-1}{e^{2x}+1} - \frac{e^{2x}+1}{e^{2x}+1} = \frac{-2}{e^{2x}+1}$ | $f(x) = 1 - \frac{2}{e^{2x}+1} = \frac{1}{e^{-2x}+1} - \frac{1}{e^{2x}+1}$

.01 t. form $\beta_i = \alpha \prod_{j=1}^{k} p_j$.

Perhaps write a quick "Point Analysis" of what I am! Just what I want to do in this part of GA/SGA, & my path (≈ TSP's) after this — & what from to be biggest

.05 But Thresholds that I can employ ... → 148.01 or : 148.01

Determining Board evaluation function.

.07 Re: [redacted] For (Chess checker) playing, [redacted] one simple way would be to use 1 or 2 ply extra to evaluate a bd evaln. function. This Gov must be tied to the final Gov (of winning Games), because, by itself it could yield ? → a self-cont. Hyperb. (e.g. (not so easy to Make examples!) stay in checkers or chess 2 Bd. evaln. function was very much based on Mobility evaln. Then this [redacted] formula Bd evaln would contain some of one did (a ply extra! — (I'm not so sure of this example ... In fact it may not be easy to find examples!) ...)

5N [redacted] Modeling a chain of ANN's can be very fast using Pent III.

It can do fast floating pt. inner product, using 4 way parallel regs for pairs. (This can be done in MMX & Pent II, but on a degraded level: On the other hand, one probly don't need floating pt. for most ANN modeling. Also looking to AMD Another by speed trick is to use TLU for the "squashing function" $\left(\frac{1}{1+e^{-x}}\right) = (1 + \tanh x/2)/2$

Since this Checker pgm doesn't use Back Prop, one doesn't need much else in the way of Arithmetic.

.23 Some other ways to Evaluate Gov fast for checkers/chess: In playing both, 2 Machine pgms; use quick-and-dirty Gov from "endgame" eg. if one player is a certain amount ahead in pieces.

.27 Pent III (has serial no.) SSE: Streaming SIMD Extensions. $\phi(x) = x^2$

MMX is SIMD; but operate on 2 32-bit integers, simultly $g = \nabla\phi$

SSE " " " " on 4 32 bit floats "

SSE has 8 registers, each 128 bits (= 4×32 bit floats)

Some things benefitted in SSE: Matrix operations: add, Multiply, dot-product, vector Normal. ... If it could do DOT Product, Pent (x4) Pent 2 almost all (matrix operations are (x4)

SSE has 70 new instrs. | There is also Enhancements to Loading, unloading registers.

(T. only O.S.'s at present that support SSE are Wgg & W.2000. 11/6/22/00

only Intel C/C++ compiler & Microsoft Macro Assembler support SSE instrs

There are 2 Pent III files in /ss/: but the Pictures are commented from Program files: I may be able to get them at the final site. ← Version of ten TABLES.

7/12/00 Bulg : H.W. (.01)

[See Aug 2000 Wired for Art on Cost buying/selling time on Machines of Many Users.

.01 : Using part-time on computers (Like CETI Project?)

I use Rhesus machines for TM2 development. Periodically, "users" get updates of TM2 : They get TM1 base, normally.

However, as time goes on TM2 will become very large (!?).

Perhaps only send from parts of TM2, or changes in TM2.

Even for TM1, the PEM could be too large for most users, or require too much Spanned Ram, Disk space etc. We may be able to devise a sub-machine that could run on a normal Desktop PC.

7/13/00 Bulp. REV Rev: 140 148 158 158 162

.01 : 146.05: A "POINT ANALYSIS": [WHERE AM I?]

At present, I've been working on SGA: Trying to derive a good approx model for Probty of G >>> point of Gold dcrn (~~for any problem~~) ~~etc~~.

— Or a $\underline{\underline{h}}_G$ plus a σ_G^2. I have several Model types, but they all seem to be inappropriate for one reason or another. Some of them may be O.K., but I don't have enuf empirical or theoretical info to be decided if they are O.K., or suggest how to fix them.

Perhaps an approach: First write good review of r. models, w. discussions of Advantages, disadvantages. Then go onto next step, which is optimizing, generalizing mutation/crossover. —

— In particular, criticize these various Mutation, Crossover techniques that are used. — Particularly Koza's crossover in ADF, ADF loops, Recursive ADF's

→ [IMP'SN] I reason I'm spending much time on GA/hotel scheduling: Solving OR problems is a known good OT's is a very important (perhaps principal) activity of TM: Therefore GA are an important class of OT's, so I have to understand/improve them. I'd like to be able to define it so it is the only OT that I have to consider, that any other OT is a "special case", & TM has to LEARN how to decide which Special case(s) to use in a particular problem.

↳ Also Note that INV problems are usually solved as OR probs: —

A kind of General OR problem (i.e. Newell & Simon's Gen Prob. Solver)

7/13/00 bdf

psg diary .15 (.38)

On "LSRCH" (.01 - .14)

.01 On "Learning" for [sequences] Induction Models for [sequences] or for [unordered sets of finite strings]. In both cases, we can have an approximate dcrn. of ε-corpus plus a cost for errors. If we are using a hill climbing to find a grammar or Model, This kind of search can be quite different. From one in which each model (say a grammar) has to be able

.06 → to generate the corpus exactly (Tho it may be possible to always express the "corrections" in terms of a special/sub set of "grammar rules")

[It would seem that this is not "Learning" at all: One starts out by writing a grammar, (a short grammar) Then one adds on ad hoc corrections to make it fit. Also, if one starts out w. a A.H. or a Promise. grammar we can regard much of the codes as being "corrections" (?). I'm not sure .06 is a really "new idea" but it may be a necessary or looking at an "old one" ? ! ☺.

.14 1-5 Aug 4, 2000 Aug 8, 2000 2nd 2nd 2nd

.15 Another Q is could one use "conventional" GA to discover CFG's (or PSG's)? The main Q is: What are the Mutations/ crossover-atoms? — One idea: We express Langs as sum of Modules concepts of sub-langs. Crossover betw. Grammars could

.19 add or subtract sub langs between Cand Grammars. (.25)

How Recursion is done re. Crossover (Only via Seq. Koza III)

Note that Sublangs also correspond to NGrams (Ngram Sets) of predictions

To do predictions, one does a wtd. mean of predictions of "Base" grammars in the population!

Maybe try this Empirically?

.25 (.19) On a Big Q is .17-.19 How to post facto "initialize" operation. The Promise crossovers & A.H. (Hoch) grammars should be included. Are there mutations/crossovers on these that could ↑ pc of corpus? Would like mut/cross that preserve ability of cands to have pc ≥ ϕ for corpus! Any crazy observed w. set of ops would ↑ pc some, but would also want "experimental" (non-greedy) xfmns. Anyway, Noting "E often follows R" in the corpus. can Heuristics give us an xfmn on one of the Cand. grammars

It would seem that the main innovation in .15 ff is the idea of "Crossover" — of mixing (some what) successful cands. The idea of simply "Mutating" cand. grammars is not new. May be keeping cands that don't necessarily ↑ pc (non-greedy) is new idea.

Say we have 2 (or more) cands w. different ways: How to combine them to make promising offspring? Offspring?

(.38) An easy way to get mutants: Pick a symbol, R. Define a set of symbols that (Precede) R. Find in the data, compressed corpus that:

New Instructions

INTRODUCTION of New Instructions/opcodes/concs into TM.

I have been using the "Precorpus" idea to introduce all symbols, w ≡ 2 pr.: wts of 1 (& csct = 1). I could (in theory) start out w. a very large (complete) set of all concs/instructions that I might conceivably use, but give them most of them csct's essentially infs of ε, & (~ 10^{-10}, say) so they would not affect operation. To "introduce" a new inst, I merely fix its csct/wts etc. This change of csct, wt. is something that TM normally does automatically, & I should be able to use my control over this mechanism to introduce new insts.

One "Natural" way would be to have new inst occur as a soln to a problem. Since G. CJ's would be used by TM to find of √ε would have to "Tool" it to soln. This would give csct of only about 1, & it would seem to be very small for a Mature TM — Hvr, a csct of 1 can be o.k. in a "New Problem Area" or if TM has some other ways to realize that this new inst. should be used only in a relatively narrow field.

→ Alternatively TM, (or TM$_2$) may find the new inst useful for recognizing much of the Old corpus, in which case its csct would be much lower, & applicable to a Broad range of problem areas. Hvr, this "recognition of crystal" is expensive, & TM would have to have good reason to believe it would be a promising thing to spend cc on.

7/14/00 Bulp

ONKOZA's More "Complete" System: having ADP's, loops Recursion:
He says His system is "several orders of Magnitude "slower than Simpler Systems. So - far larger "Diversity", enormous pay a price.
T. System is capable of solving examples w. much more "complex" Ecos, but at much greater cc. No surprise!
Hur, if Done properly, this cc could be reduced tremendously!

7/14/00 Bulg:

.01 on Th "FLAT P.D." problem.

It is clear that a flat (non-normalizable) P.D. is not really a P.D.

One recent example was: The pc's in a sequential process. Another (related

.04 problem) is pc's of a "damaged" laser. (143.05) → .20

.05 Another (sort of) example, is Continuous Distributions. Even if they are "normalized", once can't do search on them, because the "grain size" is not 'dominant'. The, in practice, overcome, by digitalizing the data is done.

It really an Approximation, — The "Phasing" of the digitalization can be varied, & may modify the results (some). One could choose an initial random phasing to avoid possibility of (lattice pts) beating of Symmetries.

.11 [What is amount to is an exhaustive search of the entire space at higher

.12 higher resolutions, each round (like the fractal Generating pgms: e.g. FRACTINT)

.13 .11 - .12 is searching for ENV prob solns. If we had a continuous space of problems, we would surely do it differently.

We start at rather coarse mesh, then ↑ the steps (say halving mesh edge or mesh volume halves). However, as soon as we get from a

.16 region to a certain region, the mesh size over the entire space is no longer Uniform.

.18 → It is how much work is saved in by the regions, (unclear) ← .23

.20 : (.04) I guess the problem of "flat pd" was most in many problems, there is ≥ 1 soln. — which SIMPLE, Direct .. .. Normzn. The many soln situation

.22 occurs in both ENV & OZ probs.

.23 : (18) One way is to assume a Lipshitz condition. This can be done, & one has

.24 some kind of "Metric", (i.e. $|G(x_1) - G(x_2)| < \beta |x_1 - x_2|^\alpha$)

It's a limitation on how "wiggly" the G function can be. Another

Another kind of limitation of this type: that used by Barron (& Rissanen 1993) in his discn. of Neural Nets & their ability to approximate functions. It had to do w/ rate of ↓ of size of the successive components of Fourier (integral expansion).

The Lipshitz condition is very nice, if one can use it. Say (.24) $\alpha = 1$ & β is known. Using successively finer mesh, in hunting for Max G, (also start w/ small β & double it for $\beta \to 2\beta$)

→ One can neglect regions of large mesh of sufficiently low G.

If the largest value at edges of a region is G_0 & the Edge is Δ, then the highest value G can have in the region is $G_0 + \Delta\beta$.

.35 Normally one doesn't know α or β, but one can assume $\alpha = 1$, then for β, use the largest β that has been observed thus far. This could cause "Backtracking". To avoid backtracking, maybe use $\beta = 2 \times$ Max β observed thus far! | See 160.08 for Goodway to do $\alpha \neq 1$

If the space is Quite Discontinuous, it's not clear as to how to search for peak. In a sufficiently 6 & continuous space, it may → 155:a1

7/16/00 Bulg

Rough/fast approximate G functions : Two (Views/Aspects)

.01 In my Analysis of GA (& SGA) I was interested in finding a good compact approximate G d.f. $\rho(G,x)$, in which it was [redacted] relatively easy to find cases of my expected G.

.04 The problem of Dealing w. Very Expensive even functions is similar. We want a cheap approx of consider → true G. Even if it is no easier to find by G values in it than 4 original "True G", this will be a BIG CC Savings.

Is there some way we can unify, mix or in some way, have these 2 aspects/ways of looking at the problem help one another?

Def X In .01 Our form of G(x) ($x \equiv x_1, \ldots$ descr of Cand$_i$) was analytically simple, so Marking it was easy or relatively easy.

In .04, 4. G(x$_i$) was simply a way of approximating & true G(x$_i$) at low cc — & it wasn't nearly easy to find by G values — the 1st way, this would be a great "pay-off".

In both .01 & .04, 4. Models we use can change/Markedly, as we slowly move to Better & Better regions of X space.

A (standard) example of .01 would be a PSG-type model of G(x$_i$). An example of .04 : In (Brown Cow) balancing problem, one normally calculates Rat path of 4 system in dynamic space by Monte Carlo step wise (Marching) solns. (≡ hy cc. {e.g. long time}). An analytic soln of 4. problem could yield very fast G evals. — but, more generally this is not poss'l. for Many dynamic (& other) problems. So one does approxns: Phy can be by using larger step sizes in 4. "Marching" soln., or by trying to find (symbolic analytic) expressions that approximate G (Symbolic Regression) → 156.06

.23 →

L = SSZ + 1 for both Mutation & Crossover.

Mutation & Crossover as ways to get $G \in GCX_i)$

.02 I originally thot of Gaussian Mutation as a general $x \to x$ probabilistic mapping

so $x \to y$ would be ~~same~~ conditional complexity of y (in view of X).

.03 This would do good heavily on ones OMC/ompd.

.04 It is a SSZ+1 type of induction. Its validity depends crucially

on how "educated" the (UMC/prior) is. It looks like a SSZ+1

In forms of formal languages, however, I think it's $L \geq 2$ (1 short

.07 (very sp. — SSZ+1)

Similarly, Crossover in induction from ~~Simple~~ SSZ = 2, but

L=3 **So REMEMBER** $\Rightarrow$ **[L = SSZ + 1]**

However, ~~the~~ T. method of .02–.05 is very general & could include

the method of .04–.07 — [I was thinking of perhaps Bernoulli Grammars

($\approx$ 2.141)] with L=2; So say X is a cand. deriv = string, ABCD.

Then, if Y = ABCE, we can derive X, Y by deriving α = ABC from

$X = \alpha D$, $Y = \alpha E$: Using .02–.03 Y could be obtained from X

by a "simple" xform. — i.e. $D \to E$ in RH-most character — T. pr of

such a xform, depending on how useful it had been in deriving t.

previously known Corpus.

NB We haven't been considering t. G prodn. here, directly.

In the case of Mutation, SSZ = 1, $\exists$ μ_G = G of original Cand. — "

Then σ_G^2 is t. by $\mathcal{Q}$. — (Also maybe μ_G = Grand Unifying Same Constant) — A STEIN ISM View

For Crossover below 2 Cands of diff G, the G d.f.'s of offsprings are

more diffl. to conjecture. ..... Conceivable t. μ_G of offspring

might be wtd mean of parents: wt. OC p_c? or maybe dependent

on how much similarity the child had to each parent. Say parents

are (parent1 A B) & (parent2 C D) child A D has (2/3 from parent1, 1/3 from parent2)

wt. of parents will be OC $(p_c A)^{-1}$ } The idea being that t. Bigger A is

" " " 2 " " OC $(p_c D)^{-1}$ } ($\equiv$ low p_c) t. more it's related to parent1.

[redacted] Bolg

7/17/00

≡ Akind of Soln to a General Maximization problem.

(.30) A Non-EC model for Ordering of Trials (Deterministic) to find Max value of G (.30) → to 156.05

Also Note Notebook (.19-.29 R)

157

(see 160.24-.31)

.01 | 152.40 | be impossible — i.e. no heuristics, so simply do 152.11 (exhaustive search) progressively longer reach.

Anyway, the Lipschitz cond of 152.23 ff is nice if one has a reasonable "Metric". Occasionally a strict Lipschitz cond isn't available but one can get PROBABILISTIC Constraints

An impt problem is the Searching of the Conventional GA/GP space w. Mutations/Crossover. How do we decide on Mesh Size, i.e. analogs of α & β of 152.24 (Lipschitz cond)? We certainly don't do an Exhaustive search like 152.11-.12. Just What Do we do? Well, in my Crossover Model, (≅ Bernoulli Grammar) Conventional GA/GP does ≅ Monte Carlo trial probably ≅ "PC". [What this is the "pc" of, is quite unclear!]

I Guess my "Thesis" is that normal GA is an approxn. to something, & it would be well to know What it is an approxn. to!

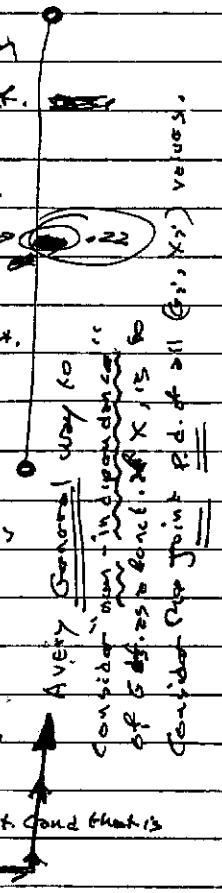
I. approach I'm taking, [redacted] is that it would be better to Get Given an approximate G as a function of X (or more exactly, the G d.f. as a func of X), Then decide how to best do the search.

Well, for the Models being used, it usually easy to find X values of hy, or Max approximate G. [In a discrete space, one should then do → (.22) .19

SN In a partly Continuous, Partly discrete space, could one not try the Lipschitz cond "(152.23 & note 152.35) — for the continuous part. .22 (.19) "trials in PC order." Unclear as to just what this means: One might try X's in order of expected value of G. (This assumes the x's are indep) .24 A more exact way (that allows the x's to be dependent) is to sort X's in order of "probby of having Max G". (I think I did work on this in the analog analysis of the "MCT as related to OZ probs

In the analysis referred to in .24 — I don't think I considered [redacted] (for non-independent X) [redacted] When one has, as part of the strategy, looked at one or more of the conds, the probby for the rest, all change. .29 (.30) So One simply computes the cond probby (more likely to be Max); One evaluates that Max Cond; & then, one calculates the cond that is the most likely to have Max G, etc. (Loop) → Evaluating Learning the G of a single cond, will (perhaps) Modify the rest of the D.P. in a simple way, so getting the next "Most likely to be max" cond, may not be so diffic!

The .30 ff seems Correct; I don't see how it deals with the need for "Granularity". Well, the way B does it: Granularity is needed because of the [redacted] hy expected correln. of G values of nearby X's. Hence, when we evaluate a G, that is probably taken and it not so dy, then it covers — 156.01



Let's go together to gether
Lets go together to get her.
There to get her together.

.01 155:40 t. expected G's of nearby x's is makes them unlikely to be "candidate Max G".
On t. other hand, if we evaluate a cand. w. hy G, t. nearby x values will have much higher probby of being "cand of Max G".
So it looks reasonable — The working out t. "practical details" may not be easy! 155:30 ff does look like a kind of General Solution to optzn. probs: probs, because cc is not considered
.05 .06 153:23 A possibly good idea! I have 2 approxns to G(x): The not nearly OK
.07 One quick & dirty & easy to find x's of hy expected G. (Mousuz! Mutation, Crossover stuff
.08 TWO More accurate, obtained by Symbolic Regression of G, x pairs! (then
— Not easy (conceptually) to find x's of hy G.
.10 We might try to replace .07, by a symbolic Regression pgm, (found by GA) in which t. pgm found are (a) short (∴ hyper ∴ good induction) (b) of fair accuracy (≡ good compression) (c) The functions used, & t. constants among them are such that it is not hard to find x's of hyper expected G.
The population that solves .10 can be updated, when get better new/data pts ((x, G) pairs). — We use same hy G population, but since we have new & data, we update their Gauss.
So .10 ff seems like a nice "Bootstrap" pgm! IT MAY BE possl. to use this idea in other parts of GA! — Or in t. whole TM System!
"Back to Reality": In .10 getting practical useful function/eqn x that might approximate G, is not easy! Perhaps my ideas about SGA, using PSG's to approximate G (or sometimes G) is relevant. I may be able to do a better job on this than GA!
→ Hmm! would .10 ff not get OSC ?? (If not this is a "grievous fault"!)
.26 Conjecture: Usually it is easy to get Max G for a P.D. than t. [Params/form] of t. P.D. are easily determined from its "corpus" (statistical measurements of w.o. GA. Possl. exception: a PSG model.
So .26 suggests that .10 ff may not be such a great idea!
.30 Say, in .26 we get t. GA to actually generate t. hy G cands. Then it would seem that t. GA would be simply trying to solve t. original GA problem — directly!
.32 Re: .30: Not Quite! What we use GA for is to find a model that (a) fits t. known {G, x} data (w. weighting over what hy G) (b) is easy to get hy "G" readings from it ("G" means t. readings of t. approxn.
→ NOTE: Of t. W'ting of errors in t. model of .32 (a)! This can be very skewed! If G is hy & estimate is low! Much penalty! If G is low and estimate is hy Then penalty depends on just how hy. Arrange t. penalties so as to Minz. t. search time needed to find peak. Because we evaluate pts Gauss directly for each data pt, it can be quite complicated Gauss, if we like! → 157.01

157.21

β-1 ε

.01: 156.10 : The simplicity of the analytic form of the total sq. error is not of much value in this nearly situation.

156.10 breaks the Main Problem into 2 (or 3) parts: Does this corresponds to obtaining a good SM strategy by first doing a prediction model of SM, then doing an optimum strategy with that model? In general, this is an [redacted] of goals. ∴ not nearly optimum: But it may be easier to find than a G.C. "Optimum" Soln.

If the true $G(x)$ is "open" to use, we can often make cheap approxns by the resolution of SSZ (in SM predn) or other standard tricks. If $G(x)$ is "slightly open" we may be able to do others this. If $G(x)$ is completely closed (BLACK BOX), we may have to depend on a standard sort of models.

SN For many problems, an ANN can be used for with modeling: Normally, it is slow, but there is this General Non-linear optimn. technique (a "[redacted]", Marquardt) which can be very fast.

@ Another fast way is # Taylor expansion; use Matrix inversion to find optimum Coefts. This last is for continuous process only: I'm really not familiar w. ANN for discontinuous spaces. The we do do it in SM strategies: A space of strategies can be discontinuous, but over a large SSZ, it can become ≃ continuous [redacted] in its continuous parameters

.21: 156.05: The General Maxm Soln. of 155.30–156.05 is perhaps best if the cc of each G evaluation is enormous: If False advantages of regularities in the $G(\cdot)$ function. Most other Optimization Methods do this minimally (the GA does do this).

.28 → One BIG defect of 155.30–156.05 It doesn't do "Experiments" (i.e. trials designed to get Info rather than direct ΔG). A possl. direction of "improvement": We select the max k (or k) trials so as to make expected ΔG.

.28 Criticism of 156.05: The "Expected value" would seem to imply a linearization of G. — A new UN desirable item: Original ($k=1$) case, had no such constraint.

.30 [For $k=1$ we want a choice $x \Rightarrow$ of all x's the probability that $G(x)$ is at
.31 is as large as poss (= "Max")

.32 For $k=2$, we want to select a strategy, which is a way of selecting x_1, obtaining $G(x_1)$, then selecting x_2 based on the new info.
Each such strategy, S_i, has a probability $P(S_i)$ of selecting an x, X_i, of max $G_i(x)$.
.35 We want (?) that $S_i \Rightarrow P(S_i)$ is max (computed for all i).

I'm not certainly sure that $k=1$ (.30–.31) is less sure than $k=2$ (.32–.35)

Because of the Monte Carlo Selection (& Probably the Roominess of the models) 158.01

7/21/00 Bulg.

Rec. 13

Rec 140
141
148
159
159
162

.01 157. to **ordinary** GA can be considered to do "Experiments".
Also Even Sim. Anneal does "experiments" & was so described. Essentially, Experiments are a way of being "UNI Greedy".
I'm not so sure! — Tho in Sim-anneal, an occasional decision to take a lower ΔG path does amount to an "Experiment". It does give info that may be of from Modifies to local P.D.

1) Work out how mut & crossover work (or should work) in new style SGA model(s) — (154.01 … discussion Pg.
2) Work out details of (1 Muxer using "IF" only! Recall idea. (.15 ref.
KMO/OR, NOR. | one big value of this would be to see what forms of P(G,x) are reasonable

.13 **A Brief Summary of Present State:**
Much Interest in Approach to working & understanding OZ & OT's via
.15 approximate P(G,x) functions. —(T. pc that a random. down "x" will have Gene of "G").
These approximation functions vary much in ⓐ how easy they are to discover ⓑ How easy they are to compute ⓒ How accurate they are ⓓ how easy it is to find the X that's most likely to have max G.
I looked at a few functional forms of this Approxn, in the stuff reviewed on 140.01 - 141.40:
Discussion of the interaction of several such P(G,x) approxns in 156.06ff. in General OT, OZ, & GA.
IMP Ideas (after many readings of 140.01 - 141.40 "Review":
Huffman Coding: Using it to Guide Search Provided Such — (for Baru Seq.) 130.02 142.01
Constructing Function Forms for Search: 136.21 is BASIC Pgm "SYMMETRIES" (130.25 - 131.18) No. of Such Functions (143.2a): Correction of Eliz formula for "Symmetries"
T: "Dawid" Lay: Poker 143.01

7/24/00 Bulg: Rev

140
141
148
158
159
162

159

Ideas source 141.40.

142.01 Hoffman coding tricks for pc Gardnerech (continued from 130.19)

142.27 A Better Understanding of OT's! Just what is Gorc's Pair models of $p(G_i, x_i)$ & how to use the model(s) for finding x_i of hy G_i?

143.05: The CFG model for G as a function of X is not nearly one in which X of hy G is easy to find! (94.23 - 95.07 is earlier work on this)

It's easy if a CFG is a "Derived" lang, in which all pts in the derivation tree derive all terminals. (not just terminals). For Non-derived langs, it can be very diff't; see 143.09 - .12.

144.01 Discusses a way to Gen & Test a set of Prefix sets to CFG's.

A Non-derived lang → a prefix set, so $\Sigma P_i = 1$.

Nvr. I prob'ly want a derived lang, so Σp_i usually $>> 1$, and X of Max G is easier to find.

144.19 ff develops a $p(G(x))$ model based on a derived lang.

145.06 — Suggest a Bug in , but I'm not sure it's relevant to a "Derived" lang.

.32 A Bug & fix for $G \to G^x$ xfmn.

.22 Ref to Koza's Volume I for (perhaps) some data on d.f. of $p(G,x)$

146.07 ON Checkers, Chess & Board eval functions

.27 PENTIUM III characteristic.

N.W. 147.01 Availability of "Part time" of ~ 10^6 computers: Cos. Prof Soft such cc.

148. REV Source of "POINT Analyses" (w have a ref)

149.01 On Search: How one (Apparently) does it normally use of to solve probs.

149.15 - .40, 161.01 - .40 PSG Disc: Some new reasonable ideas! Possl. use of GA.

150.01 How to INTRODUCE NEW INSTRUCTIONS / opcodes) into TM

151.01: On & cc assoc w. Koza's "More Complete" system (including ADF, loops, recursion)

152.01 .04, .20 - .21 On "Effective (unnormed) PD" problem for L Srch.

.05 - .12 Search in Continuous space for "INV probs.

152.13 — .40, 155.01 — .40 ANALOGUE 154.01 — .05 · 157.21 & most criterion:

This is a good discn. of OT on Continuous spaces: It gives a good understanding model(s) for search: So that one is certain of Global max of the G function satisfying the assumed Constraints. Discusses (w.o. such constraints) Optimizing Process (160.24 - 31).

The 152.13 ff discn deals w finding Global Max; & it is not the same as the OZ problem at all, it is of much interest. In 152.13 the problem is to find a Global Max as cheaply as possl.

This Differs Much from OZ problems.

NOTE: In OZ probs (T. comment's End of OT problem) We do not look for Global MAX.

Also 166.08 — .40

153.01 - .23; 156.06 - .40; 157.01 - .19 ; Several Different Approxns to $p(G,x)$ are used in GA: Two opt ones: (a) If G(x) is very expensive: we use a cheaper $G'(x)$ that gradually improves as we approach final soln. (b) we use a very diff $G''(x)$ function that max is at same soln for gradient hill climbing in GA. Mutations & Recomb. are 2 aspects of such a $G''(x)$ function. 162.01