Toward a Model of Mind as a Laissez-Faire Economy of Idiots
Extended Abstract

Eric B. Baum
NEC Research Institute
4 Independence Way
Princeton, NJ 08540
eric@research.nj.nec.com

Abstract.

I argue that the mind should be viewed as an economy, and describe an algorithm that autonomously apportions complex tasks to multiple cooperating agents in such a way that the incentive of each agent is exactly to maximize my reward, as owner of the system. A specific model, called "The Hayek Machine" is proposed and tested on a simulated Blocks World (BW) planning problem. Hayek learns to solve far more complex BW problems than any previous learning algorithm. If given intermediate reward and simple features, it learns to efficiently solve arbitrary BW problems.

1 Introduction

I am interested in understanding how human-like mental capabilities can arise. Any such understanding must model how large computational tasks can be broken down into smaller components, how such components can be coordinated, how the system can gain knowledge, how computations performed can be tractable, and must not appeal to a homunculus[18].

What these smaller components, or agents, calculate, how they calculate it, and when they contribute, can't be specified by a superior intelligence either. But somehow these things must be determined. This can only happen if they are somehow given rewards so that they learn what and when to contribute. But these rewards can't be specified by a homunculus either. They must, perforce, be specified by agents determined in similar fashion. Thus we are driven to a picture where a set of agents pass around rewards, i.e. an economy.

One who ignores the economic aspect of mind risks implicitly proposing misguided local incentives. In my intuition and experience, complex dynamical systems evolve to exploit their incentive structure. We will give below some examples where suboptimal incentive structure has led to problems.

This paper introduces a system I call The Hayek Machine and tests it on a (simulated) Blocks World (BW) problem. Hayek consists of a collection of agents, initialized at random. Each agent consists of a function which maps states of the world (including any variables internal to Hayek) into a bid, action pair. Computation proceeds in steps as follows. At each step (a) the action of the agent with high bid is taken, (b) it pays its bid to the previous active agent, (c) it receives any reinforcement offered by the world on that step. Since agents pay and are paid their supply of money fluctuates. Any agent with negative money is removed. New agents are created by a random process.

What motivates Hayek's design? I have access to the world, e.g. Blocks World. This is a valuable resource because correct actions will create wealth, i.e. payoffs from the world. But I don't know how to utilize my resource. So I let a bunch of agents bid for it. The winning bidder buys the world–that is, he owns the right to take action changing the world, and to sell the world to another agent. He is *exactly* motivated to improve his property. The agent whose action adds the most value can afford to outbid all others. Conversely, agents' bids are forced close to the expected value after they act by competive bidding from copycat agents with similar actions. With a few caveats described in the text below, a new agent can enter Hayek's economy if and only if he increases Hayek's overall wealth production. Thus does the invisible hand provide for global gain from local interactions of Hayek's agents.

Thus the entry and loss of agents executes a randomized local search, making small random steps (addition or deletion of one agent) and accepting them if they improve overall performance. For massive, complex optimization problems, local search is often the best algorithm. For the problem of designing a mind, or an economy, where the detailed knowledge necessary to do anything better is dispersed and one can not appeal to a superior intelligence, it is arguably optimal.

In The Hayek Machine, I have (currently) restricted agents to consist of a triplet: condition, action, bid. Here the condition is a Boolean conjunction of a certain type. If the condition is true, the agent makes its bid, otherwise it bids nothing. Thus each agent has a fixed bid and a fixed action. Roughly speaking, this is analagous to a nearest neighbor approach to learning the functions mapping state to a bid, action pair.

In the standard Q-learning[25] or RTDP[2] framework, an evaluation function maps states of the world to numerical estimates of expected value. If there are many states, one must regularize somehow to avoid the curse of dimensionality. Hayek maps condition action pairs to expected value. This generally seems to be a reasonable strategy if there are relatively few actions. Indeed, since Hayek removes unprofitable agents, but keeps all sufficiently profitable agents, it can dynamically approximate the right number of agents consistent with the data it has seen and its ability to learn from it.

After two days of training, Hayek learns to solve Blocks World problems involving eight blocks if started from *tabula rasa*. This compares favorably to the abilities of a simple handcrafted program, using a similar, impoverished representation, and is comparable to the abilities of general purpose planners (c.f. § 7). Given the simple features "top of stack", and intermediate reinforcement, Hayek learns to solve *arbitrary* instances, and its solution is within a logarithmic factor of optimality on almost all instances. Once trained, Hayek instantly solves novel instances, much as humans spend years learning to speak, but then utter appropriate sentences without reflection.

Orthodox models within the field of economics make several assumptions. They typically postulate that all economic agents have complete information, and infinite computational power. They also typically discuss equilibrium behavior. Real economies, of course, are not in equilibrium (e.g. technological progress is an evident and interesting economic phenomena) and involve interaction between humans who, while computationally powerful, are not infinitely so. In Hayek the individual agents are simple, automatically generated rules, seeing incomplete and differing information. Thus Hayek has an, inherently evolutionary, economy of idiots. Hayek also models interaction with a complex physical world from which wealth may be extracted by careful, complex, and coordinated actions, and about which it is possible to learn. This too is not a feature of most standard economic models. Hayek is thus also of interest as a model of economics. It sheds light on when and why market forces may lead to efficient wealth production, addressing critical issues bypassed by orthodox economics. This motivation and application will be discussed elsewhere[4],[3].

1.1 Previous Related Work

Many authors have considered how mental capabilities may arise from the interaction of smaller units, e.g. [18][9] [23][22][14][19]. These models are not economic and in many cases arguably have suffered seriously from distorted incentives. See e.g. [17] for a discussion of parasitic behavior in Eurisko. In my view, specifying the interaction of dynamic systems in the Society of Mind[18] would be rather more challenging than centrally managing the Soviet economy.

Holland's seminal Classifier systems[12] were (to my knowledge) the first explicit proposal of an economic analog of mind. Holland also first suggested condition, action, bid agents. Holland classifiers have, however, been viewed as empirically disappointing[26]. I think that Holland classifiers again suffer from distorted incentives. Initial wealth values for agents are a critical problem. New agents inject their fictitious initial money into the system, which, at least in our experiments, is seen to cause inflation and to create misdirected incentives to fleece novice agents, instead of creating real wealth. Perhaps to mitigate this problem, Classifier systems often proceed in generations, with all or many existing rules (or at least their bids) replaced by a new generation periodically. By contrast Hayek learns continuously, slowly adding new agents.

Many rules act simultaneously in Classifier Systems. Typically all rules active at the time share payoff from the world. A phenomenon called "the tragedy of the commons"[11] besets economies whenever property is held in common. Everybody's incentive is to overgraze a communal field even though the long run effect is to destroy the resource. Likewise each rule has incentive to be active even at the expense of lowering overall payoff. This could perhaps be solved by defining each rule's property rights, say by allowing only one physical actor at any given time, who would collect all payoff from the world. Other rules might simultaneously engage in 'mental actions'.

Alternatives to some of Holland's non-economic design choices should perhaps also be considered. Holland Classifiers use a particular encoding[1] which has been shown capable of "universal computation"[10], However, universal computers require infinite memory. Practical considerations have generally forced use of small systems. These are effectively small finite state machines. Are they a particularly efficient or learnable encoding of finite state machines?

Genetic algorithms (GA's) may possibly be highly inefficient at training Classifier systems. While recent results[5] show that some GA's can beat hill climbing dramatically in some contexts, and even provide some direct motivation for their use in training classifier-like systems, GA's often often scale less efficiently, leading to huge performance divergences on large problems. Hayek is (currently) trained by hill climbing.

Miller and Drexler have proposed "Agoric computation", a model of computing based on an economic analog[16]. These authors remark that computational markets may avoid any imperfections of real ones. Hayek incorporates a perfectly accurate credit check which advances credit to exactly those agents who will be able to meet their obligations.

Several authors have studied learning to plan in a Blocks world context. However the versions of Blocks World they have studied are far simpler than that studied here. All previous learning work I'm aware of involved an unrestricted table size, and more importantly did not involve learning an abstract goal. Whitehead and Ballard[24] applied Reinforcement Learning techniques to learn to pick up a green block, with at most three other blocks on top of it. Birk and Paul[6] attempted to learn to move a single block around an otherwise empty table. Koza[13] addressed a simplified problem having an unbounded table, the concrete goal of building the same fixed stack every instance, fixed number of blocks and at most two obstructing solution, and built in pre-supplied sensors giving all pertinent information, including, e.g., the Next Needed Block, and both the top block and the top correct block on the target stack.

There is also a large literature on applying hand coded planners to Blocks World. I compare the results achieved by Hayek to several of these efforts in § 7.

[1] Standard Holland Classifiers use a trinary alphabet $\{0, 1, *\}$. A * in the action string posts the corresponding component value in the condition string. It is not evident why this choice of encoding is desirable. A long sequence of operations must be evolved to somehow post a simple function of input components such as negation.

2 Blocks World

My Blocks World simulates the following physical situation. A 'table' contains $S = 4$ stacks of blocks, each one of $c = 3$ colors. The zeroth (leftmost) stack is a 'goal' stack. Hayek controls a hand which if empty may pick the top block off any stack but the goal, and if full may drop its block on any stack but the goal. Hayek's object is to copy the zeroth stack in the first (or 'target') position. It does not, however, know that this is the task- it must discover it by reinforcement.

This world is encoded as a set of vectors of form $(a_0, a_1, a_2, \cdots, a_{n-1})$. Here a_0 runs from 0 to $S-1$ and denotes which stack. $a_i \in \{0, ..., c\}$ represents the color of the block at height i. $a_i = 0$ means there is no block at that height. If $a_i = 0$, then $a_j = 0$ for $j > i$. As Hayek takes actions an operating system maintains the state of the world consistent with the physical world modelled. Hayek can take actions of form (i, a) where $i \in \{1, 2, \cdots, S-1\}$ and $a \in \{g, d\}$. This action moves the hand to column i and either grabs the top block or drops a block held on top.

Hayek sees a series of instances of this form. It has $N = 100$ actions to solve an instance. We[2] have experimented with three schemes of reinforcement. In the first Hayek is rewarded for partial success. When it either removes a block from the target which must be removed, or places a correct block on the target it is given reward 1. When it removes a correct block or places an incorrect block, it is penalized 1. When it solves an instance, it is rewarded 10.

In the second scheme Hayek is only rewarded upon correct completion of an instance with no intermediate reward. Hayek was presented with successively larger instances. At stage i, Hayek dealt with instances having $\lfloor \frac{i+2}{2} \rfloor$ blocks distributed over the target and goal stacks and $\lceil \frac{i+2}{2} \rceil$ blocks distributed over the other two stacks. It works on a given stage until nearly perfect and then moves onto the next.

In the third scheme Hayek is "taught" both by intermediate reward and by successively larger instances.

There are several motivations for studying how much Hayek's performance is improved by intermediate rewards. One is the hope that in coming decades, when computers are much faster and programs more complex, and hence less robust, rather than program your computer one might instead code up an intermediate reward function, teaching a successor of Hayek to solve

[2] All computer code was written by Charles Garrett. See the acknowledgement at the end of the paper.

your problem. Another motivation is that, while evolution learned from *tabula rasa*, humans are born with complex reward functions, built into them by evolution, which allow them to learn much more rapidly.

3 Hayek

Hayek consists of (1) a planning system, and (2) a learning system. The planning system consists of a collection of agents. Each agent has a condition, a numerical bid, and an action. The condition is a Boolean formula. If the condition is true, the agent bids. The highest bid wins and that agent's action is taken. It pays its bid to the previous active agent, and receives in turn payment from the next active agent, plus reward if it solves the instance. Whenever an instance ends with any agent having negative wealth, that agent is removed from the system and the wealth and bids of all other agents are reset to what they were before the instance.

The agents have form

$$A \cap B \cap \cdots \cap D \Rightarrow (a, b)$$

Here (a, b) describes the action of moving the hand to column specified by a and taking action b. Here a takes values from the set $\{*x, 1, 2, \cdots, S\}$ where $*x$ is a wildcard variable and b takes values from $\{g, d\}$, where g means "grab" and d means "drop".

A, B, etc. above are expressions of form u(op)v where (op) is either $=$ or $\neq$ and u and v may be either b (meaning blank) or h (the current content of the hand) or a grid location of form (x,y) where $x \in \{*x, 0, 1, 2, ..., S-1\}$ is a stack and $y \in \{*y, 1, ..., h\}$ is a height. The condition is deemed valid if some instantiation of its wild cards makes it true[3].

3.1 Agent Creation

The learning system is composed of two modules. The first produces agents by a randomized process, and the second assigns them bids. The randomized rule creation samples from the set of potential rules roughly uniformly. Half the rules are novel and half are mutations of previous rules. A new rule has i clauses in its condition with probability $p_i = 1/2^i$. Each clause is chosen uniformly from the space of alternatives. A mutation involves insertion, deletion, or replacement of a random clause. We have not experimented with the probabilistic parameters, simply choosing them roughly uniformly, except that we tried two settings for P_*, a parameter determining the probability that a wildcard appears, and we tried a naive metalearning scheme, in which all parameters were self determined. Hayek's performance did not vary noticeably between these experiments.

3.2 Bid determination

In the human economy, humans use their considerable computational abilities to decide what and when to bid. Hayek's agents must discover this autonomously. If their bids estimate accurately the value of their action given their condition is valid, then several desirable things occur. First, the bidding process then chooses at each step the agent most likely to lead to a solution. Second, new agents can enter precisely if and only if they improve overall performance of the system (with some caveats discussed shortly).

The simplest method for choosing bids is that each agent when created is assigned a fixed bid. Agents with appropriate bids survive. To speed learning, I modified this slightly. Agents are created without an assigned bid, and called novices. The first time a novice agent's condition is true, its bid is fixed at ϵ more than the high bid from applying veteran rules[4] and it becomes a veteran.

Mutations create new versions of successful agents, which are often virtually identical to their parents, except in the bid. But now the mutated rule with a higher bid supercedes its predecessor. This is analagous to entrepreneurs entering a real economy to compete with a successful business, driving his profit margin to zero. Thus the bids of all rules get driven upward to the point at which the agent is no longer profitable. That is, they get driven to the expected value of the states the agent's action reaches.

If any agent ends an instance with negative capital, it is removed and the remaining agents' bids are reset to where they were before the instance. New agents frequently make bids they can not subsequently cover. If you run Hayek without the reset, this injects fictitious money into the system which causes speculative bub-

[3] This encoding implicitly encodes topology, departing from *tabula rasa*. Human infants are apparently born with implicit knowledge of topology and object permanence[8]. Of course, by the time he had any chance of solving these BW problems, a child would have learned much about the world. Arguably our current encoding is comparable in prior knowlege to an infant's, and vastly less than that of a child capable of BW solution.

[4] If two novices apply, one is selected randomly. ϵ was set by fiat to be 0.2. If no veteran applies than the novice's bid is fixed at *epsilon*.

bles, as agents can afford to bid any amount, so long as the next agent bids more. Misguided incentives cause this inflation. A niche exists in the economy of fleecing novice agents, who lose money they don't really have.

In a real economy, agents must have capital to make purchases. Entities (companies, nations, or people) do sometimes purchase on credit and then default. However the sellers in a human economy are intelligent and make considerable effort to assess worthiness before extending credit. By resetting, Hayek effectively runs a perfect credit market, in which agents extending credit appeal to an oracle that foretells whether the applicant will meet his obligations.

We experimented with other means of bid selection. To compare with Real Time Dynamic Programming (RTDP) or Temporal Difference type (TD) learning[21], we updated the bid of active agents by $b \to b(1-\epsilon) + (b' + \text{payoff})(\epsilon)$ where b' is the bid of the following agent, payoff is any reward on that particular step, and ϵ is a small parameter. The idea here is that b' is an estimator of the value of being in the subsequent state. $b' + \text{payoff}$ thus estimates the value of being in the current state and using the given agent. b averages such estimates. This method did not work well because it was subject to inflation. The constant entry of new agents injects noise.

Holland's classifier systems[12] set bids by $b = \alpha s W$ where α is a small constant, s is the specificity of a classifier, and W its wealth. This type of rule is sometimes justified as setting the bid (a dependent variable) in terms of the wealth, the notion being that wealthier rules, and more specific rules, should be applied preferentially, However I see this rule setting the wealth in terms of the bid. If the bid is less than the expected pay-in, the wealth will rise when the rule is used, and with it the bid, until the bid is exactly equal to the expected pay-in. Thus I ignored specificity as irrelevant in determining the limit bid.

Setting bids by $b = \alpha W$ requires initiating the wealths of new agents. If these wealths are non-zero, money is pumped into the system, perverting the incentive structure and causing speculative bubbles. Instead, I merged the Holland rule with the fixed bid scheme. A novice agent's bid was determined and fixed when first active, as in the fixed bid scheme, and it was labelled an "apprentice". When it accumulated sufficient capital to justify this bid, i.e. when first $W \geq b/\alpha$, it was designated a "veteran" rule and subsequently we set $b = \alpha W$. In practice, becoming a veteran is a lengthy process and at any typical time most agents were apprentices. I call this scheme "Modified Holland".

3.3 Time Scales, Cherrypicking, and Stability

The previous section remarked that if every agent's bid is equal to the expected value of the states it generates, then a new agent can profitably enter if, and only if, the expected reward to me (as owner of Hayek) is greater when it is applied than that when its initial direct competitor is activated. This motivates the hope that Hayek will hillclimb in performance, but there are some important caveats. First, the system is stochastic, so the bid estimates are inherently noisy.

Second, the expected reward for using an agent is averaged over the instances where the agent is applied. In any given instance, one agent's action might be better than another's, even though the other is better on average. This gives rise to the possibility of "cherrypicking". Say we have a profitable agent A, and an agent B is introduced into the system whose domain overlaps A, and say B has a higher bid than A. Then B may cherrypick A's best instances, so that A can no longer profit at its old bid. This might occur even though A was better at handling these clients, since A's bid reflects its handling as well of harder instances. This cherrypicking phenomena is an artifact arising from the agents' inflexibility in assigning bids to configurations. However, when cherrypicking occurs, the assignment overall becomes more accurate. B here more accurately prices its subset of A's client states. If A's bid can adjust (e.g. if A is a "veteran" rule in the modified Holland pricing), then A may readjust its bid to more accurately reflect the value of its remaining clients. This more accurate pricing is a valuable step towards learning itself.

That Hayek learns is evidence his economy achieves considerable stability. Hayek extends knowledge gained on relatively small instances to learn solutions to larger instances. Evidently this will not happen if intermediate market crashes cause loss of memory.

4 Statistical Observations

Consider the rule

$$(0,1) \neq (1,1) \Rightarrow (1,g)$$

which says- if the bottom color in the target is not the same as the bottom color in the goal, then remove the top block in the target stack. Hayek's chances of generating this rule de novo can be seen to be about

$1/32,000$ assuming $P_* = 0.1$, or about 2×10^{-6} assuming $P_* = 0.5$. (Recall from §3.1 that P_* determines the probability of wildcards in random rule creation.) This rule may mutate to

$$(0,*) \neq (1,*) \cap (1,*) \neq e \Rightarrow (1,g)$$

in the $1/50,000$ rule creation range[5] with $P_* = 1/2$ or $1/6 \times 10^6$ with $P_* = 0.1$. If any block in the target is not the same as the corresponding location in the goal, this rule removes the top block on the target– a universal clearing rule. The probability of Hayek creating this rule randomly, instead of by mutation from a smaller rule, is roughly 2×10^{-8} if $P_* = .1$ and 4×10^{-7} if $P_* = .5$. Likewise one can estimate probabilities to create rules useful for placing correct blocks on target.

Such exercises lead to several observations. First, tens to hundreds of thousands of rules is a lot to sort through in order to find a useful one. On the other hand, there is no plausible alternative to sorting through a large number of agents unless Hayek is somehow told what its goal is.

Hayek's price mechanism effectively speeds this search. Contrast, say, an alternate approach, Brand X, where agents received payoff based on actual end result, as opposed to TD-like estimates of the next state's value. Say Brand X is solving 95% of simple instances. Say a novice rule is introduced which simply wastes an action, and hence slightly worsens performance, to 90% say. This will be typical. Brand X must apply this rule 10-100 times before it can decide to remove it. Hayek, by contrast, would estimate on the first application whether this rule went from a better to worse state, and if so discard it.

Hayek's local search on collection of agents is a big win over, say, random search. The probability of producing even the one universal clearing rule above would be much smaller if it were not possible to build it out of smaller useful components. The probability of producing a set of rules working together would be prohibitively small if sets were not built of useful agents discovered separately.

Parameter settings could in principle impact greatly the probabilities of finding useful rules. If $P_* = 0.1$ Hayek has a much easier time producing the first rule discussed, but a much harder time generalizing it to the second. Having several different types of creation rules seems called for.

[5] These estimates are explicitly calculate probabilities of one or two simple paths, ignoring many alternative paths. Plausibly many other paths might sum to a significant contribution.

5 Explicit Variables

Hayek's current representational language, as specified in §3.1, does not seem sufficiently general to express a general solution to arbitrary BW problems. Consider the rule:

$$(0,*) = h \cap (1,*) = e \cap (1,*-1) \neq e \Rightarrow (1,d)$$

which says: if holding the next needed block, place on target. Currently there is no notion of "$*-1$" in Hayek, so Hayek can not express this. The main thrust of ongoing work is how to improve Hayek to expand its representational capability autonomously. (see §8).

In the meantime, I have added some useful variables to Hayek's representation by hand. I introduced three terms: top[1], top[2], top[3]. top[i] is the height at which the next block will go, if placed on stack i. When creating a new random condition, wherever a term of "type" height might appear, top[i] is placed randomly with probability $(1/3)P_{top}$.

We tried this with $P_* = .25$ and $P_{top} = .25$, and with incremental rewards as well as staged learning. It then produced a system capable of solving arbitrary instances of BW problems. The final system solves all but exponentially rare instances with a number of actions within a logarithmic factor of optimal. (On such pathological instances, it can use quadratically more actions than optimal.) Here is the rule set it finds (omitting some rules that are never applied, because they are superceded by more general rules with higher bids):

```
(1) Bid= 14.4166 (0,{*0})!=(1,{*0}) &
(1,{*0})!=HAND & EMPTY==EMPTY => 1,G

(2) Bid= 14.0712 (0,Top[1])==EMPTY => 3,D

(3) Bid= 13.3013 (1,{*0})!=(2,{*0}) => {*1},D

(4) Bid= 13.2446 (0,Top[1])==(3,{*0}) => 3,G

(5) Bid= 13.2242 (1,15)==({*0},{*1}) => 2,G

(6) Bid= 12.1369 (3,{*0})!=(2,4) => {*1},D
```

Agent 1 is a universal clearing rule: whenever a bad block is on the target, it lifts the top block off target. Agent 2 places these blocks on stack 3 whenever the target is taller than the goal. Agents 3 and 6 drop on a random stack whenever a block is in hand. Agent 4 grabs from stack 3 whenever stack 3 contains the next useful block. Agent 5 grabs from stack 2. Roughly speaking the program works as follows. Agent 1 keeps bad blocks off the target. This is highest priority. Agents 4 and 5 dig for useful blocks. Because of agent 4's higher bid, it digs from stack 3 if a useful block is found there, else 5 digs from stack 2. Whenever a useful block is picked up, it is dropped on the target with probability one third, and no correctly placed blocks are ever removed from the target.

Note the role of the intermediate payoffs in allowing this program. This program relies on the differing bids to prioritize agents. In the current economic model, without intermediate payoffs, the agents' bids must get larger the *later* they are used in an instance, and may not be made high in order to prioritize an agent, precisely so that it will be used early. Thus the intermediate payoffs allow greater flexibility in *representation*. In fact, Hayek succeeded in learning a general (and efficient) solution to the problem the first time it had the evident capability of expressing such a solution. As yet it is unclear the extent to which the intermediate payoffs are necessary to guide the learning, as opposed to allowing a sufficiently flexible representation.

6 Other Empirical Results

In experiments, TD systems and "straight" Holland-style systems, not utilizing any form of perfect credit verification, exhibited interesting phenomena such as speculative bubbles in bids and crashes, but could not solve BW instances with more than a few blocks.

The fixed bid systems and the Modified Holland were able to learn the multistage problem, with end payoff only, up to stage 5 or 6 in almost all cases. Experiments typically ran for about 2 days on a MIPS R4400 cpu at 150 MHz, generating 1-2 million agents in the course of the run. At any given time, 20-250 agents were alive. Stage 6 involves 8 total blocks.

These two systems also learned with intermediate payoffs and 2 blocks per stack (no staged instances). Applied to 3 blocks per stack with incremental payoffs, they solved only about 25% of the instances.

Hayek running with stages plus incremental payoff learns up to stage 9 of the multistage problem, implying that it was near perfect on stages 1-8. Stage 9 involves 11 blocks. It solved 946 out of 1000 instances from its training set and 939 out of 1000 from a separate test set. Thus Hayek seems to generalize well to examples drawn from the same distribution as its training set.

7 Discussion

Hayek has learned to solve Blocks World planning problems which are substantially more complex than any addressed by previous learning approaches, c.f. § 1.1. I expect it would be at best difficult to train a monkey to solve these problems, and hopeless for a dog or a preverbal child. Hayek solutions (even without incremental feedback and top[i] variables) are better than our simple programming efforts, using a similar impoverished representation, in which it is impossible to specify a general program.

Bacchus and Kabanza[1] tried some sophisticated planning algorithms such as SNLP [15],[20], Prodigy 4.0 [7], and their own TLPlan on Blocks World Problems related to (but different from) ours. Even when run on a Blocks World with an unbounded table, SNLP, Prodigy, and TLPlan all exceeded resource bounds on problems of about six blocks. TLPlan augmented with hand-coded special-purpose knowledge about the blocks world problems solved problems with around 50 blocks. TLPlan was then run on a table which like ours had room for only three blocks. With simple handcoded rules, it solved problems with about 12 blocks. With a complete backtrack-free strategy, it solved 35 block problems.

Hayek is learning to solve (somewhat different) problems involving 8 blocks, with only payoff upon completion of instances, or 11 blocks, with incremental payoff. If augmented with useful variables (see § 5), so that it has the capability of expressing a general solution, Hayek in fact finds one, and one which is furthermore computationally efficient for almost all instances. Thus Hayek generated a general solution the first time it evidently could express one.

Note that SNLP, Prodigy, and TLPlan know what the goal is. This knowledge is not given to Hayek, and indeed is one of the main things it has to learn. Note that Hayek spends all its time *learning*. Once it has produced a rule set, it can solve a given instance instantly. SNLP, Prodigy, and TLPlan are using around 400 CPU seconds to *solve* each instance, and using many actions. Note that Hayek was limited to 100 actions, in the experiments reported. While the results here are not strictly comparable, it is suggestive that Hayek is performing reasonably well on a very hard problem.

8 Open Questions and Ongoing Work

In this first paper, agents have only been allowed physical actions. To tackle more general problems, agents need capabilities for internal, or computational actions. These include, perhaps, creating or writing on purely internal registers, read by other agent's conditions. The full range of human behavior, e.g. early vision, no doubt requires some agents utilizing real, as opposed to Boolean, values. Also, rather than having agents created as described in § 3.1, agents should be able to create new agents by modifying a third agent. This recurses Hayek on the problem of suggesting new agents, replacing the current random search. Thus Hayek should metalearn as creation agents are trained to look in likely directions. Within Hayek creation agents should be rewarded precisely as investors in the created agents. The economic importance of intellectual property rights in the created agent will be studied. Previous efforts at metalearning (e.g. [14]) did not view the system as an economy, and so, in my opinion, did not correctly compensate their meta-agents, resulting in distorted incentives.

Biological evolution learned starting from *tabula rasa*, driven strictly by reinforcement. The struggle for survival is real time. Quick reaction is critical. Evolution might naturally produce reflexive systems analagous to Hayek's current rule sets. My view is that "thought" must have evolved along these lines also. So thought evolved as I am trying to extend Hayek. Early creatures learned agents which just performed actions, and then evolution learned that these actions could be internal, i.e. computational. Hayek should learn to expand his representation gradually, and would hopefully thus simulate the evolution of "thinking". Moreover, recall the saying: "ontogeny recapitulates phylogeny". If you believe that thought evolved in this way from reaction driven systems to reflective systems, then this rule would indicate that thought develops in infants along similar lines. This I believe reasonable. Children start out reactive, and gradually expand their representation. As they do they become able to deal with more complex concepts.

Hayek will also be run on more complex problems. One question is whether it will be able to apply knowledge learned in one problem to another. Another question is whether it can stably apply separate agents to more than one problem. Human economies, like ecosystems, develop multiple niches. Our minds, likewise, have many different interacting skills. I intend to train Hayek on several distinct tasks and observe what sort of niche structure develops. Niche structure may be Hayek's answer to the frame problem. One might also hope to find Hayek "reasoning by analogy" by creating a relatively small number of new agents that allow it to solve some new problem using a set of agents previously created for a different problem. Hayek's creation agents might plausibly engage in traditional search by suggesting many tries when the system is at an impasse (so the bid is cheap).

It is evident that self organization of human-like reasoning and planning capabilities may require significant computation. Say we had a computer with the brain's capabilities, perhaps 10^{15} cycles/second, and we ran the brain's learning algorithm on it for a full year. The results of this massive investment would be a system with the capabilities of a one year old baby. On the other hand, humans labor under potential handicaps. For example, massively parallel, slow, locally connected systems are likely less efficient per raw cycle compared with fast uniprocessors. Also humans were designed by evolution, which is arguably vastly less efficient than market mechanisms (again because of imprecise incentives)[17]. Assuming we understand how optimally to design a human-like intelligent systems, how much raw computing power will we need to solve interesting problems?

Finally, numerous questions are suggested within the domain of economics[4].

Acknowledgements

Charles Garrett wrote all the code and ran all the experiments in the first year, from high level (English language) specifications. I thank him for his efficient and expert assistance which was integral to making this paper a reality. Michael Buro modified Garrett's code (again from English language specifications) to run some of the experiments reported in § 5. I would also like to thank Brian Arthur, Andreas Birk, Michael Buro, Charles Garrett, Adam Grove, Michael Kearns, Melanie Mitchell, Steve Omohundro, Harold Stone, and David Waltz for useful comments on a draft or a talk, and Fahiem Bacchus, Jose Scheinkman, and Warren D. Smith for conversations.

References

[1] Bacchus, F., Kabanza, F. (1995) Using temporal logic to control search in planning, unpublished document available from http://logos.uwaterloo.ca/tlplan/tlplan.html.

[2] Barto, A. G., Bradtke, S. J., Singh, S. P. (1995) learning to act using real-time dynamic programming, AI Journal, to appear.

[3] Baum, E. B., (1996) Toward a Model of Mind as Laissez-Faire Economy of Idiots, full paper, http://www.neci.nj.nec.com:80/homepages/eric/eric.html

[4] Baum, E. B., (1996) Models of Both Mind and Economy, to appear in *The Economy as an Evolving Complex System II*, eds W. Brian Arthur, Steven Durlauf, and David A. Lane, Santa Fe Institute series Addison Wesley, Reading MA.

[5] Baum, E. B., Boneh, D. Garrett, C. (1995) On Genetic Algorithms, in *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, pp 230-239.

[6] Birk, A., Paul, W. J., (1995) Schemas and Genetic Programming, document to be published.

[7] Carbonell, J. G., J. Blythe, O. Etzioni, Y. Gill, R. Joseph, D. Khan, C. Knoblock, S. Minton, A. Perez, S. Reilly, M. Veloso, and X. Wang (1992) Prodigy 4.0: The manual and Tutorial. Technical Report CMU-CS-92-150, School of Computer Science, Carnegie Mellon University.

[8] Cosimidies, L. and J. Tooby (1992) Cognitive adaptations for Social Exchange, in Barkow, J. H., L. Cosimidies, and J. Tooby (1992) *The adapted mind*, Oxford University Press. NY., pp 163-228.

[9] Drescher, G.L.(1991)*Made-Up Minds*,MIT Press.

[10] Forrest, S. (1985) Implementing semantic network structures using the classifier system. In *Proc. First International Conference on Genetic Algorithms*, pp 188-196. Hillsdale NJ: Lawrence Erlbaum Associates.

[11] Hardin, G. "The Tragedy of the Commons". Science, 1968, 162, 1243-1248.

[12] Holland, J. H. (1986) Escaping brittleness: the possibilities of general purpose learning algorithms applied to parallel rule-based systems. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, (eds.) *Machine Learning II* pp 593-623, Los Altos CA Morgan Kauffman.

[13] Koza, J.R., (1992) Genetic Programming, MIT Press, Cambridge MA, pp 459-470.

[14] Lenat, D. B., "The role of heuristics in learning by discovery: three case studies", in Michalski,R.,S., Carbonell, J.G., and Mitchell, T., eds, *Machine Learning: An Artificial Intelligence Approach*, (Tioga Pub. Co., Palo Alto CA 1983 pp 243-306.

[15] McAllester, D., and D. Rosenblitt (1991) "Systematic nonlinear planning" in *Proceedings of the AAAI National Conference*, pp 634-639.

[16] Miller, M. S., and K. E. Drexler, "Markets and computation: Agoric open systems", in B. A. Huberman, ed, *The Ecology of Computation*, Studies in Computer Science and Artificial Intelligence 2, North Holland, New York, pp133-176, (1988).

[17] Miller, M. S., and K. E. Drexler, "Comparative ecology", in B. A. Huberman, ed, *The Ecology of Computation*, Studies in Computer Science and Artificial Intelligence 2, North Holland, New York, pp51-76, (1988).

[18] Minsky, M., (1986)*The Society of Mind* Simon and Schuster, NY.

[19] Newell, A. (1990), *Universal Theories of Cognition*, Harvard University Press, Cambridge MA.

[20] Soderlan, S. , Barrett, T., and Weld, D. (1990) The SNLP planner implementation, contact bug-snlp@cs.washington.edu.

[21] Sutton, R. S. (1988) Learning to predict by the methods of temporal differences, Machine Learning 3: 9-44.

[22] Valiant, L. (1995) Rationality, in *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, 3-14.

[23] Valiant, L. (1994) *Circuits of the Mind*, Oxford University Press.

[24] Whitehead, S. D. and D. H. Ballard. (1991) "Learning to Perceive and Act." Machine Learning 7, 1, 45-83.

[25] Watkins, C. J. C. H. (1989) *Learning from delayed rewards*, Doctoral thesis, Cambridge University, Cambridge England.

[26] Wilson, S. W., Goldberg, D. E. (1989) A critical review of classifier systems. In J. D. Schaffer, ed. Proc. Third International Conf. on Genetic Algorithms San Mateo CA, Morgan Kauffman.