A Graph-Dynamic Model of the Power Law of Practice and the Problem-Solving Fan-Effect

Jeff Shrager; Tad Hogg; Bernardo A. Huberman


Stable URL:
http://links.jstor.org/sici?sici=0036-8075%2819881021%293a242%3a4877%3c414%3aaamotp%3e2.0.co%3b2-w

Science is currently published by American Association for the Advancement of Science.

Your use of the JSTOR archive indicates your acceptance of JSTOR’s Terms and Conditions of Use, available at http://www.jstor.org/about/terms.html. JSTOR’s Terms and Conditions of Use provides, in part, that unless you have obtained prior permission, you may not download an entire issue of a journal or multiple copies of articles, and you may use content in the JSTOR archive only for your personal, non-commercial use.

Please contact the publisher regarding any further use of this work. Publisher contact information may be obtained at http://www.jstor.org/journals/aaas.html.

Each copy of any part of a JSTOR transmission must contain the same copyright notice that appears on the screen or printed page of such transmission.

JSTOR is an independent not-for-profit organization dedicated to creating and preserving a digital archive of scholarly journals. For more information regarding JSTOR, please contact support@jstor.org.
A Graph-Dynamic Model of the Power Law of Practice and the Problem-Solving Fan-Effect

JEFF SHRAGER, TAD HOGG, BERNARDO A. HUBERMAN

Numerous human learning phenomena have been observed and captured by individual laws, but no unified theory of learning has succeeded in accounting for these observations. A theory and model are proposed that account for two of these phenomena: the power law of practice and the problem-solving fan-effect. The power law of practice states that the speed of performance of a task will increase as a power of the number of times that the task is performed. The power law resulting from two sorts of problem-solving changes, addition of operators to the problem-space graph and alterations in the decision procedure used to decide which operator to apply at a particular state, is empirically demonstrated. The model provides an analytic account for both of these sources of the power law. The model also predicts a problem-solving fan-effect, slowdown during practice caused by an increase in the difficulty of making useful decisions between possible paths, which is also found empirically.

The power law of practice (1), one of the few solid psychological learning phenomena, states that the speed of performance of a task will increase as a power of the number of times the task is performed. In one model, problem solving can be viewed as the search for a path through a directed “problem-space” graph, where nodes represent states of the problem or facts in memory and edges represent operators that move between states (2). Solving the problem involves finding a path from the initial state to the goal state by means of the available operators. Learning in this model corresponds to changes in either the specific topology of the graph or the decision procedure used to decide which operator to apply at a particular step when there is more than one edge emanating from a node. Many sorts of changes in method and operators can be modeled as changes in the topology of the problem-space graph, including restructuring and method selection. In this report we use computer experiments to show that this learning model exhibits the power law and that the phenomenon can be explained analytically by a theory based on graph dynamics. Our theory further predicts a problem-solving “fan-effect” in which performance becomes slower as more operators are learned in certain situations (3). This prediction is also empirically validated by our simulations.

The simple problem that we will use to explore learning phenomena, the “bit game,” is analogous to many real problems. A problem state in the bit game is a 2-bit binary vector (such as 0101). For the sake of concreteness we will use a 5-bit vector (B = 5) in most cases. A “trial” begins with an arbitrary initial state, say 00000. The player (a computer) searches for some other arbitrary vector (the goal state), say 11111, by successively applying operators that change the contents of the state vector. Each operator is composed of 1 to 2 elements indicating a particular bit in the vector that should be flipped if it matches in the current state. Operators specify the bits in the state that actually change and can be written as “pattern → result” pairs, with question marks (?) where the operator pattern says nothing about a particular bit position. For instance, the operator 1??1? → 0001 will take the state 11010 to 10000 or the state 11111 to 10101 but will not apply to the state 00000 because the bits indicated in the pattern do not match this state. As a result of the question mark “don’t care” bits, operators vary in their generality. For instance, each of the two-element operators, such as 01?? → 10??, apply to eight different states (in this case 00100, 00101, 01100, 01101, 01110, 01111, 10100, 10101, 10110, and 10111).

We begin playing a particular bit game with all of the (2B) 1-bit operators (10, in the case of a 5-bit game). This set forms a B-dimensional hypercube and ensures that marks (?) where the operator pattern says nothing about a particular bit position. For instance, the operator 1??1? → 0001 will take the state 11010 to 10000 or the state 11111 to 10101 but will not apply to the state 00000 because the bits indicated in the pattern do not match this state. As a result of the question mark “don’t care” bits, operators vary in their generality. For instance, each of the two-element operators, such as 01?? → 10??, apply to eight different states (in this case 00100, 00101, 01100, 01101, 01110, 01111, 10100, 10101, 10110, and 10111).

We begin playing a particular bit game with all of the (2B) 1-bit operators (10, in the case of a 5-bit game). This set forms a B-dimensional hypercube and ensures that marks (?) where the operator pattern says nothing about a particular bit position. For instance, the operator 1??1? → 0001 will take the state 11010 to 10000 or the state 11111 to 10101 but will not apply to the state 00000 because the bits indicated in the pattern do not match this state. As a result of the question mark “don’t care” bits, operators vary in their generality. For instance, each of the two-element operators, such as 01?? → 10??, apply to eight different states (in this case 00100, 00101, 01100, 01101, 01110, 01111, 10100, 10101, 10110, and 10111).

We begin playing a particular bit game with all of the (2B) 1-bit operators (10, in the case of a 5-bit game). This set forms a B-dimensional hypercube and ensures that marks (?) where the operator pattern says nothing about a particular bit position. For instance, the operator 1??1? → 0001 will take the state 11010 to 10000 or the state 11111 to 10101 but will not apply to the state 00000 because the bits indicated in the pattern do not match this state. As a result of the question mark “don’t care” bits, operators vary in their generality. For instance, each of the two-element operators, such as 01?? → 10??, apply to eight different states (in this case 00100, 00101, 01100, 01101, 01110, 01111, 10100, 10101, 10110, and 10111).
there is at least one path between any two
nodes. When there are several applicable
operators for a particular state, a decision
procedure is required in order to choose
among them. The number of edges tra-
versed on a trial (that is, the number of steps
required to find the goal state from the
initial state) measures performance. For sim-
plicity, we assume that all operators contrib-
ute equally to performance, although opera-
tors with variable cost could be used to
model problems of specific sorts. A series of
trials, beginning with a common initial
problem-space and with learning between
each trial, will be called a problem-solving
"run."

We separately simulated operator addi-
tion and decision-procedure improvement
in the bit game. Operator learning takes
place after each trial is completed in the style
of SOAR chunking (4). Specifically, we add
the operator that most generally summarizes
the solution obtained in the trial. For in-
stance, suppose we begin with the game, 01110 → 10101, and find a solution. The
"subproblem solving" for this game is sum-
marized by adding the operator that solves
this particular game in one step. In this case
the new operator is 01110 → 10101. Notice
that the ? element of this operator appears
because the third bit did not change be-
tween the initial and goal state in this trial,
and so this new operator connects two pairs
of states in the problem-space, that is, it adds
two (directed) edges to the problem-space
diagram.

Decision procedures can be arbitrarily
complicated algorithms with changes lead-
ing to entirely different problem-solving be-
behavior. For a given organization of opera-
tors and choice of initial and goal states,
certain decision procedures will be more
effective than others. Decision procedures
generally change radically only in the face of
some new insight into the problem struc-
ture. Without such an extreme change it
only makes sense either to slowly vary the
parameters controlling the decision pro-
cedure in order to try to hill-climb into a best
solution mode or to vary them randomly,
hoping to discover a good decision pro-
cedure serendipitously. We consider changing
between a "poor" decision procedure and an
"optimal" one. The optimal decision pro-
cedure finds the fastest way to the goal. On
the other hand, the poor decision procedure is a
random walk. As operators are added to the
problem-space through learning, it becomes
more densely connected. Hence one has to
do less searching to find a path leading to
the goal, but it is also easier to get off the
path.

To explore the range of decision pro-
cedures that lie between optimal problem solv-
ing and a random walk, we use a simple
descriptive model of the effectiveness of the
decision procedure in which, at any node
during the search for the goal, each unpro-
ductive edge is eliminated with probability r.
Improvements in the decision procedure corre-
pond to an increase in r and change the problem from an exponential random
search to a linear drift toward the goal. Note
that r = 1 corresponds to a perfect decision
procedure in which search and backtracking
are never required, whereas r = 0 corre-
sponds to a random walk on the graph.

In order to implement a decision pro-
cedure incorporating this parameter for the bit
game, we first find all applicable operators
from the current state and then order them
by asking for each operator how many bits
would be correctly set (for the desired goal
state) if this operator were actually applied.
We then separate these into the "good" ones
(those that minimize the Hamming distance
to the goal) and all the remaining "bad"
ones (those that do not minimize this dis-
tance). Next each operator from the bad set
is removed from consideration with proba-
bility r. Finally, we choose one operator at
random from the union of the remainder of
the bad set and all the optimal operators.
When r = 1, all of the bad operators will be
deleted, leaving only the good ones. When
r = 0, all of the bad operators are left in the
set, making the decision procedure a ran-
donum search. It is important to note that even
when r = 1, this implementation is only
heuristic—it only approximates optimal
problem solving. Actual optimal solution
paths can only be found by exhaustively
exploring the graph beforehand, a very
lengthy computation. However, this heuris-
tic comes very close to the optimal path
(with r = 1) in the bit game.

Figure 1 shows how operator learning
affects performance of the 5-bit game in the
three main decision-procedure effectiveness
regimes: optimal (Fig. 1C, r = 1), mediocre
(Fig. 1B, r = 0.5), and random (Fig. 1A,
r = 0). In all cases we randomly chose a start
and goal state, solved the problem according
to the indicated decision procedure, and
recorded the performance. Recall that learn-
ing takes place after each trial by adding the
operator that most generally summarizes the
solution path just found. The possible
games are uniformly distributed among the
225h bit configurations. All experiments were
run on a 16384-processor Thinking Mas-

mles Connection Machine (CM-2).

Notice, first, that the results in Fig. 1, A
and C, are consistent with the power law—
they appear approximately straight on a log-
log plot. More interesting, however, is the
fact that adding edges improves perform-
ance in the optimal and random cases but
initially degrades performance in the regime
of the mediocre decision procedure. This is
the problem-solving fan-effect, wherein
learning hurts performance rather than helps
it. Nevertheless, the absolute performance
ranges are, as expected, best for the optimal
strategy, worst for the random strategy, and
medium for the mediocre strategy. Thus, in
order to actually improve as a result of
learning, one must start with a moderately
good decision procedure or else, as learning
takes place, one must improve the decision
procedure in addition to learning new oper-
ants.

The power law and fan-effect are observed
in many situations (1, 3, 5, 6) although the
quantitative details of their forms will differ
for each different task. In order to under-
stand the general nature of these phenome-
na, we now show how learning that results
from the addition of edges in a problem-
space graph or improvements in the decision
procedure lead in some cases to a gradual
reduction path length with a corresponding
gradual improvement in performance that is
a power of the number of trials (the power
laws), and in other cases to a gradual in-
crease in path length (the fan-effect). Recall
that a problem-space can be modeled as a
diagram with n nodes representing various
problem states and with edges representing
instances of possible operators. We are inter-
ested in the learning behavior for situations
involving a large number of states and typi-
cal problem-spaces rather than any specific
one (7). This leads us to consider typical
elements of the class of all problem-spaces of
a given size, namely, random graphs, where
the initial operators are distributed at ran-
dom and where new edges are added inde-
dependently of one another. When large
graphs are involved, this model is mathema-
tically equivalent (8) to one in which
every edge between a pair of states exists
with independent probability p. As new
operators are learned during the trials, p will
correspondingly increase. In order that all
nodes are almost surely reachable from any
given node, p should be greater than (ln n)/n
(8).

For this model we want to obtain an
expression relating the expected number of
steps, s, required to obtain a solution, to the
values of r and p. We make a number of
simplifications to the model which neverthe-
less retain its essential features. First, we
assume that all nodes of the graph have the
average number of links: μ = (n - 1)p. We
are left with a regular graph consist-
ing of n nodes with uniform branching ratio μ. Second, we assume that the cycles in
the graph are long so that, in general, at any
node there will be one edge that is one step
closer to the goal while the others are one
step farther away. In this limit, which applies when the graph is sparsely connected, the behavior will be similar to a walk on a tree. Because of the initially exponential growth in the number of nodes with distance, the initial and goal nodes will usually be separated by the diameter of the graph, which can be approximated as $D = \ln n / \ln \mu$.

With these approximations, at each node there is only one choice that gets closer to the goal state and $\mu - 1$ choices that move farther from the goal. However, the decision procedure eliminates each incorrect choice with probability $r$ so there are (on average) effectively only $(\mu - 1)(1 - r)$ incorrect choices at each node. Thus the problem reduces to a bounded, one-dimensional random walk in which one starts at distance $D$ from the goal and moves randomly until the goal is reached. This motion is constrained to remain within distance $D$ of the goal and, at each step, to move toward the goal with probability

$$P = \frac{1}{1 + (\mu - 1)(1 - r)} \quad (1)$$

and away with probability $P_0 = 1 - P$. The average time required to reach the goal can be derived by standard techniques (9). It is given by

$$s = \frac{2}{(2P - 1)^2} \left[ P^2 + P_0 (D - 1) - (D/2) + \frac{(1 - P_0)^D + 1}{P_0^2 - 1} \right] \quad (2)$$

This provides an explicit form for the expected behavior of $s$ as a function of $P$ (topology) and $r$ (decision effectiveness) because the values appearing in Eq. 2, $P_0$ and $D$, are expressed in terms of these two basic parameters.

When $P > P_0$, the dominant behavior for large $D$ is a drift toward the goal so that

$$s \approx \frac{D}{P - P_0} \quad (3)$$

When $P = P_0$, Eq. 2 gives $s = D^2$, corresponding to symmetric diffusion. Finally, when $P_0 > P$, $s_0$ is greater than $P_0$, and

$$s_0 \approx 2 \left( \frac{1 - P_0}{P_0} \right)^{D - 1} \frac{1 - P_0}{(1 - 2P_0)}^2 \quad (4)$$

which grows exponentially with $D$. We thus see a dramatic change in the nature of the search process as $P_0$ passes through the critical value of 0.5.

Equation 2 produces the range of behaviors observed in the bit game experiments.

![Fig. 2. Log-log plots of the theoretical predictions of $s$ versus $\mu$ for various $r$ values, from Eq. 2. (A) the random walk; (B) mediocre decision procedure; (C) optimal decision procedure. In all cases $\lambda = 10,000$.](image)

For instance, when the decision procedure is weak ($r$ near 0), $P_0$ will be small ($\mu$ is at least as large as $\ln n$) and roughly equal to $1/\mu (1 - r)$. When $\mu (1 - r) > 1$, one obtains

$$s_0 \approx \frac{2 n}{\mu} (1 - r)^n / \ln \mu - 1 \quad (5)$$

In this case, when $r = 0$ (choices made at random), increasing the number of edges reduces the time to solve the problem. However, when $r > 0$ (but $\mu (1 - r)$ remains much larger than unity), there is a range in which increasing the number of links will result in a gradual increase in $s$, the expected number of steps required to solve the problem, because the smaller diameter of the graph is more than balanced by the increased difficulty of choosing the correct operator from among the larger number of choices. Specifically, $s$ will increase as links are added when $1/\ln (1 - r) / \ln n > (\ln \mu)^2$, which holds when $r$ is not too close to zero, and there are many nodes and not too many edges. Conversely, when the decision procedure is strong ($r$ near 1), the system's behavior is governed by the drift behavior of Eq. 3. When $\mu (1 - r)$ is much less than 1, Eq. 3 gives

$$s_0 \approx \ln n / \ln \mu \quad (6)$$

Thus as long as the decision procedure improves sufficiently fast as new links are added, one obtains a power-law decay in $\ln \mu$. Figure 2 shows the behavior of $s_0$ for increasing numbers of links in the three important $r$ value regimes. In all cases the path length decreases or increases, corresponding to the experimental behavior.

We have shown that by applying the theory of graph dynamics to a problem-space viewed as a graph, we can capture, explain, and experimentally demonstrate the power law and the problem-solving fand- effect. Our approach should be contrasted with other theories of the power law and fan-effects. First, Anderson's ACT* model (5) obtains the power law by a rule-strengthening mechanism that itself operates according to a power law. Second, our approach is more general than the approach of Rosen- bloom (1, 6), whose account of the source of the power law is restricted to addition of operators resulting from the chunking of problem-solving subgoals. We also incorporate improvements that result from adding operators and improvements in the decision procedure. Furthermore, we predict a power law for any sort of operator addition (in the appropriate decision-procedure regimes), whereas Rosenbloom predicted power laws only in the case of subgoal chunking.

REFERENCES AND NOTES


10. We thank J. Heikamp and F. Ritter for detailed discussion of these issues and for comments on the paper. S. Card, J. Laird, A. N. Newell, P. Rosenbloom, and S. Stornetta contributed to our thoughts on this research. We thank the reviewers of Science who also provided many helpful comments.

5 July 1988; accepted 26 August 1988