Hazhir Rahmandad hazhir@vt.edu, John Sterman jsterman@mit.edu, Nelson Repenning nelsonr@mit.edu

Supplementary material to accompany:

Effects of Feedback Delay on Learning

The following supplementary material includes the detailed formulations of the four learning

heuristics discussed in the paper, an alternative method for measuring the speed of internal

dynamics of the models, the statistics for parameters and results of robustness analysis,

formulations for Q-learning heuristic tested for comparison with the four learning models

discussed in the text, as well as instructions for inspecting and running the simulation learning

models discussed in the paper and the Q-learning model. The table of contents is followed by the

supplementary material.

Mathematics of the model

This appendix includes the formulation details for all four learning heuristics; Regression,

Correlation, Myopic Search, and Reinforcement; as well as the exploration procedure. Please see

the online material for the simulation models and applets that allow you simulate the models and

run different experiments.

Table S1- Reinforcement Learning Heuristic[1]

Reinforcement Learning Heuristic

Variable	Formulation
$\dfrac{d}{dt}V_j(t)$	$I_j(t) - G_j(t)$
$I_j(t)$	$P(t)^{ReinforcemenetPower} * \overline{A}_j(t)$
$G_j(t)$	$V_j(t) /\ Reinforcement\ Forgetting\ Time$

Table S2- Myopic Search heuristic

Myopic Search Heuristic

Variable	Formulation
$\dfrac{d}{dt}V_j(t)$	$\left(V_j^*(t) - V_j(t)\right)\big/\lambda_{Myo} * Direction(t)$
$Direction(t)$	1 if P(t) > Historical Payoff(t) 0 if P(t) ≤ Historical Payoff(t)
$Historical$ $Payoff(t)$	$Smooth(P(t),\ Historical\ Averaging\ Time\ Horizon)$ Where smooth function is defined by: $\dfrac{d}{dt}(smooth(x,\tau)) = \dfrac{x - smooth(x,\tau)}{\tau}$, Initial value of $smooth(x,\tau)$ is equal to initial x unless explicitly mentioned
$V_j^*(t)$	$FractionofActions_j(t) * \sum V_j(t)$ Where $FractionofActions_j(t)$ is: $\dfrac{\overline{A}_j(t)}{\sum \overline{A}_j(t)}$

Table S3- Correlation heuristic

Correlation heuristic

Variable	Formulation

---

[1]The literature on reinforcement learning has different goals and heuristics in different fields. In the context of machine learning, reinforcement learning is developed as an optimization heuristic. Game theory literature on reinforcement learning focuses on providing simple heuristics that give good descriptive accounts of individual learning in simple games. Our use of reinforcement learning here is more aligned with the game theoretic reinforcement learning models. For a survey of machine learning literature on reinforcement learning see Sutton and Barto's book (1998) and Kaelbling et al. (1996).

$\dfrac{d}{dt}V_j(t)$	$(V_j^*(t) - V_j(t))\big/\lambda$
$V_j^*(t)$	$V_j(t).f(Action\_PayoffCorrelation_j(t))$ Where $f(x)$ is: $2.(\dfrac{e^{2*x*\gamma}}{1+e^{2*x*\gamma}})$ and γ is *Sensitivity of Allocations to Correlations*
$Action\_PayoffCorrelation_j(t)$	$\dfrac{Action\_PayoffCovariance_j(t)}{\sqrt{ActionVariance_j(t)}*\sqrt{PayoffVariance(t)}}$
$Action\_PayoffCovariance_j(t)$	$smooth(ChangeinAction_j(t)*ChangeinPayoff(t),$ $ActionLookupTimeHorizon)$
$ActionVariance_j(t)$	$smooth(ChangeinAction_j(t)^2, ActionLookupTimeHorizon)$
$PayoffVariance(t)$	$smooth(ChangeinPayoff^2(t), ActionLookupTimeHorizon)$
$ChangeinAction_j(t)$	$\overline{A}_j(t) - smooth(\overline{A}_j(t), ActionLookupTimeHorizon)$
$ChangeinPayoff(t)$	$P(t)- smooth(P(t), ActionLookupTimeHorizon)$

Table S4- Regression Heuristic

Regression Heuristic	
Variable	**Formulation**
$\dfrac{d}{dt}V_j(t)$	$(V_j^*(t) - V_j(t))\big/\lambda$
$V_j^*(t)$	$Max(\alpha_j^*(s)\big/\sum_J \alpha_j^*(s), \varepsilon)$, if at least one $\alpha_j^*(s)$ is non-zero otherwise $V_j^*(t) = V_j(t)$ $s = E* INT(t/E)$ where E is the evaluation period (the decision maker runs the regression model every E time periods). INT function gives the integer part of t/E
$\alpha_j^*(s)$	$\alpha_j^*(s) = Max(0, \alpha_j'(s))$ The $\alpha_j'(s)$ are the estimates of α_j in the following OLS regression model: $\log(P(t)) = \log(\alpha_0) + \sum_j \alpha_j * \log(\overline{A}_j(t)) + e(t)$ The regression is run every Evaluation Period (EP) periods, using all data between time zero and the current time (every time step is an observation).

3

Table S5- Exploration procedure

Exploration	
Variable	**Formulation**
$\hat{V}_j(t)$	$V_j(t) + \hat{N}_j(t)$
$\hat{N}_j(t)$	$N_j(t).\sum_J V_j(t)$
Var ($N_j(t)$)	*Minimum Action Standard Deviation+(Maximum Action Standard Deviation-Minimum Action Standard Deviation)*S.D. Selector(t)*
S.D. Selector(t)	*Min(1,Max(0, 1/Slope Inverse for S.D. Selection*(Recent Improvement(t)/Min Improvement-1))* Where *Recent Improvement* equals: Smooth(*Max(0, P(t)-Historical Payoff(T)), PayoffImprovementUpdatingTime), Initial Recent Improvement=1*

Speed of internal dynamics of the model

In the paper we used the convergence times of different models in the base case as a proxy for the speed of internal dynamics of the models. An alternative measure which does not directly depend on our implementation of convergence criteria can be obtained by fitting a first order negative exponential curve to the base case (no delay) average performances of the heuristics. Following the practice in control theory, we measure four times the time constant of the fitted curve, which gives the time needed for the model to settle in the 2% neighborhood of final value. We use this as a sensible measure for the time needed for the organization to settle down into a policy, and therefore a good measure of the speed of internal dynamics of the model. The following table shows the values of this measure for different heuristics:

Table S6- The alternative measure for speed of internal dynamics.

Model	Regression	Correlation	Myopic	Reinforcement
Exponential Time	38.32	190.64	227.76	60.4

Constant* 4

These results are consistent with the premise that delays tested in the model are not too long
compared to internal dynamics of the system.

Statistics for detailed analysis parameters and results

The independent parameters in the regressions included all those changed randomly for the

Monte-Carlo simulation analysis, picked from uniform distributions, reported below. For each

parameter we report the low, high, and base case values, as well as the description of the

parameter. In table 7 we also report the summary statistics for the dependent variables of the

regressions.

Table S7- Parameter settings for detailed analysis

Parameter		Heuristic	Low	High	Base	Description
Payoff Generation Delay[activity 1]	$T1$	All	0	20	0	How long on average it take for resources allocated to activity 1 to become effective and influence the payoff
Perceived Payoff Generation Delay[activity1]	$\overline{T}_1$	All	0	20	0	The decision maker's estimate of the delay between resources allocated to activity 1 and the payoff.
Action Noise Correlation Time	δ	All	1	9	3	The correlation time constant for the pink noise used for model exploration
Value Adjustment Time Constant	λ	Regression Correlation	3	20	10	How fast the activity value system moves towards the indicated policy
Value Adjustment Time Constant	λ	Myopic	0.5	8	2	How fast the activity value system moves towards the indicated policy
Maximum Action Standard Deviation		All	0.05	0.2	0.1	What is the standard deviation of exploration noise, when the exploration is most vigorous
Minimum Action Standard Deviation		All	0	0.05	0.01	What is the standard deviation of exploration noise, when the exploration is at minimum
Minimum Improvement of Payoff		All	0.005	0.02	0.01	Normalizing factor for changes in payoff. Changes lower than this will trigger minimum standard deviation for exploration
Slope Inverse for Standard Deviation		All	0	20	9	Inverse of the slope for relating recent improvements to standard deviation of exploration. Higher values suggest smoother transition from high exploration to low.
Payoff Improvement Updating Time		All	3	20	5	The time constant for updating the recent payoff improvements (which is then normalized to determine the strength of exploration)

Action Lookup Time Horizon		Correlation	2	10	6	The time horizon for calculating correlations between an activity and the payoff
Sensitivity of Allocations to correlations		Correlation	0.05	0.8	0.2	How aggressive the activity values (and therefore allocations) respond to the perceived correlations between each activity and the payoff
Reinforcement Forgetting Time		Reinforceme nt	3	20	10	The time constant for forgetting past activity values.
Reinforcement Power		Reinforceme nt	4	20	12	How strongly the differences in payoff will be reflected in action value updates.
Evaluation Period	*E*	Regression	1	9	3	How often a new regression is conducted to recalculate the optimal policy

Table S8- Summary Statistics for Dependent variables

Variable	Observations	Mean	Std Dev	Median
Achieved Payoff Percentage[Rgr]	3000	87.2	17.8	93.0
Achieved Payoff Percentage[Crr]	3000	75.6	25.4	83.3
Achieved Payoff Percentage[Myo]	3000	79.6	20.2	85.0
Achieved Payoff Percentage[PfR]	3000	81.7	18.7	86.7
Distance Traveled[Rgr]	3000	18.1	11.5	15.1
Distance Traveled[Crr]	3000	17.5	11.0	14.8
Distance Traveled[Myo]	3000	18.4	12.2	15.4
Distance Traveled[PfR]	3000	18.0	12.3	14.7
Convergence Probability[Rgr]	3000	0.236	0.424	0
Convergence Probability[Crr]	3000	0.122	0.328	0
Convergence Probability[Myo]	3000	0.178	0.383	0
Convergence Probability[Pfr]	3000	0.200	0.400	0

A Q-learning model for resource allocation learning task

Among different learning heuristics we considered in the original design of this study, a Q-learning heuristic taken from machine learning literature had some interesting properties to make it a potential candidate for inclusion in the original paper, however, it was excluded from the final draft under space constraints, its relative inefficiency in comparison with other models discussed for a comparable number of trials, and complexity considerations. Here we briefly discuss this heuristic because it gives a better insight into optimization-oriented learning heuristics available in machine learning and artificial intelligence literatures, it has been

introduced into organizational learning literature (Denrell, Fang and Levinthal 2004; Rahmandad 2007), and it helps build a more concrete mathematical framework to relate complexity of learning to action-payoff delays. In comparison with other reported models, this heuristic has a relatively high level of information processing and rationality, however, it does not use any cognitive search, i.e. it assumes no knowledge of payoff landscape. Consequently the heuristic is quite general.

In the development of this model we follow the general formulation of Q-learning (Watkins 1989) as discussed by Sutton and Barto (Sutton and Barto 1998). Typical reinforcement learning heuristics in machine learning (The general category to which Q-learning heuristic belongs) work based on a discrete state-space and use discrete time. Therefore to adopt this heuristic to our task which has both continuous time and action space, we break state-action space of our problem into discrete pieces and use a discrete time in the simulations reported here. Also, here we only report the development of the Q-learning model for the case with no delays. As the analysis reported here and the discussions regarding the extension of the model into delayed feedback show, such extension is possible but increases the complexity of the model exponentially with the size of delays considered and therefore the number of experimental data-points to allow for a successful performance under that condition goes beyond the neighborhood of what is considered feasible in an organizational setting.

The action space in our setting depicts what resource allocation the organization is using at the current time, i.e. resources allocated to each of the 3 actions, A_j (j=1..3), however, since sum of these resources equals a constant total resource (R=100) at each time, we have a 2-dimensional action space, A_1 and A_2, each changing between 0 and 100, and with the constraint that $100 \geq A_1 + A_2$. We break down each action dimension into N possible discrete values. We choose N=12 in the following analysis since that allows for observation of the peak of the Cobb-Douglas payoff function used in our analysis (PF=$A_1^{0.5}.A_2^{0.3333}.A_3^{0.1667}$), consequently, each A_j can take one of the $R/N*i$ values (i=0..N) provided that sum of A_j equals R.

To define the action space, we assume that at each period the organization can only explore one of the neighboring states, i.e. by shifting 1/N resources from one activity to another, or can stay at the last state. Consequently the action space includes 7 possible actions (No change, A_1 to A_2, A_2 to A_1, A_1 to A_3, …). However, at the boundaries of the state space (where $A_j=0$ for any of js) the feasible action set reduces to exclude possibility of existing the feasible state space. Consequently our action-state space, (S, a), includes 559 possibilities, where each S consists of a (A_1, A_2) pair and "a" can take one of its feasible values.

At each period the organization takes an action (goes from state S to state S', through action "a"), observes a payoff (*P*) from that move, and updates the value of the action-states Q(S,a) according to the following updating rule:

$$Q(S,a) \leftarrow (1-\alpha).Q(S,a) + \alpha.(P + \gamma.\underset{a'}{Max}\,Q(S',a'))$$

Parameter α is used to avoid too-fast an adjustment of values in case of stochastic payoff functions. Since here the payoff is deterministic, we use a very high discount rate, α=0.9, in the analysis below. Central to the contribution of Q-learning heuristic is that it not only takes into account the reward received from the last action-state, but also gives a reward for how good a state that action has brought us to (maximum value available from S' through any action a'). Therefore this heuristic allows for taking into account the contributions of different states in leading us towards better regions of payoff landscape and creates a potential avenue through which one can capture the effects of time-delays more efficiently, i.e. by rewarding not only what payoff we get at this step, but also what new state the action takes us to, a future looking perspective. Parameter γ determines the strength of that feedback on model performance.

The action taken at each period depends on the last state, the value of different actions originating from that state, and the tendency of the organization to explore different possibilities. Here we can have two types of exploration, one includes exploring actions which we have never taken from the current state (exploring new (S,a)'s), the other reflects organizations propensity to

try an action which has a lower value than maximum, even though it has been tried before (exploring (S,a)'s which have already been explored). Both these types of exploration are needed to lower the chances that the organization gets stuck exploiting a low-payoff location on the state-space. Starting from a value of 0 for all Q(S,a)s and taking into account that all actions in this landscape, except for those on the borders, give a positive payoff, we use the following equations to define the probability of taking action "a" from state S, P(S, a):

$$P(S,a) = \frac{\upsilon}{\upsilon.Z + (1-\upsilon).K} \text{ if Q(S,a)=0}$$

$$P(S,a) = \frac{Q(S,a)^{\omega}}{\sum_{a'} Q(S,a')^{\omega}} \cdot \frac{(1-\upsilon)}{\upsilon.Z + (1-\upsilon).K} \text{ if Q(S,a)>0}$$

Here υ represents the attention given to exploring new actions and ω represents the priority given to exploring already-tried state-action pares. In the following analysis we use the values of 0.3 for υ and 4 for ω which maximized the speed of learning for this model on the given task within a 3 by 3 ([0.1, 0.3, 0.5]*[1, 4, 7]) grid of values for υ and ω.[2]

Since this model follows a different logic both in defining the space and time states and in exploration and value adjustment heuristics, we have to find the appropriate simulation length to make a comparison between this model and those reported in the paper meaningful. This is important in the light of the fact that given enough time/experimental data and exploration, the Q-learning heuristic is guaranteed to find real values (their expected payoff in an optimal policy) of different state-action pairs, and therefore be able to make optimal allocations (Watkins and Dayan 1992). The important question is then whether such convergence can happen faster or slower than

---

[2] To find these parameter values, we ran linear regressions with dependent variable of average payoff over 10 simulations of 100 periods and independent variable of TIME, to find the largest learning rate (the largest regression coefficient on TIME) within the given parameter settings.

the other learning models, and what is the impact of introducing delays on the time/trial requirements of this model.

The basic idea behind such comparison is the availability of similar amount of data about the payoff landscape[3]. Since such data arrives through the learning agent's exploration of the landscape, gathering comparable amounts of data requires similar speeds for different heuristics when walking on the state-space in a non-converging mode. That is, the distance traveled on the state-space should be similar through the length of simulation for Q-learning and other four heuristics.

In the case of other four models the total distance traveled remains below 20^4 for most of the cases (See table S8) where most simulations have not converged. Therefore a reasonable comparison allows the Q-learning heuristic enough time to walk on the state-space for a similar distance.

In the case of the Q-learning heuristic where activity dimension is divided to N discrete values for each dimension, each step to a new position moves the organization on the state-space for distance d:

---

[3] Note that because of the continuous-time nature of the four original heuristics, the auto-correlation of exploration term, and the independence of their behavior from time-step of simulation (for dt<0.2), it is not realistic to count the total number of time steps in 300 periods of continuous simulation and do a direct comparison with same periods for the Q-learning method. Such logic will give increasing number of trial periods to Q-learning, as we reduce the dt for the continuous model, which is obviously wrong.

[4] The distance traveled on state-space is calculated in four original models based on summing the Euclidian distance between points the heuristic visits every period on the 3-dimensional space of fraction of resources allocated, we follow the same logic here.

$$d = \sqrt{(\frac{1}{N})^2 + (\frac{1}{N})^2} = \frac{\sqrt{2}}{N}$$

Taking into account the probability that the organization stays where it is in a random walk

(which equals $\frac{91}{559}$), the distance traveled with each step is d'~ 1/N. Consequently, M, total

number of periods the Q-learning model needs to run to travel the distance of 20 is:

M=20*N=240

Therefore a reasonable comparison between the Q-learning and other models can be

made when the Q-learning model is simulated for about 240 periods.

Using the parameter values reported above, we run the Q-learning heuristic for as long as

600 periods in order to make sure that we observe the behavior of the learning heuristic outside

the range of data-availability allowed for other models. We average the results over 400

simulations for higher confidence.

Learning Performance
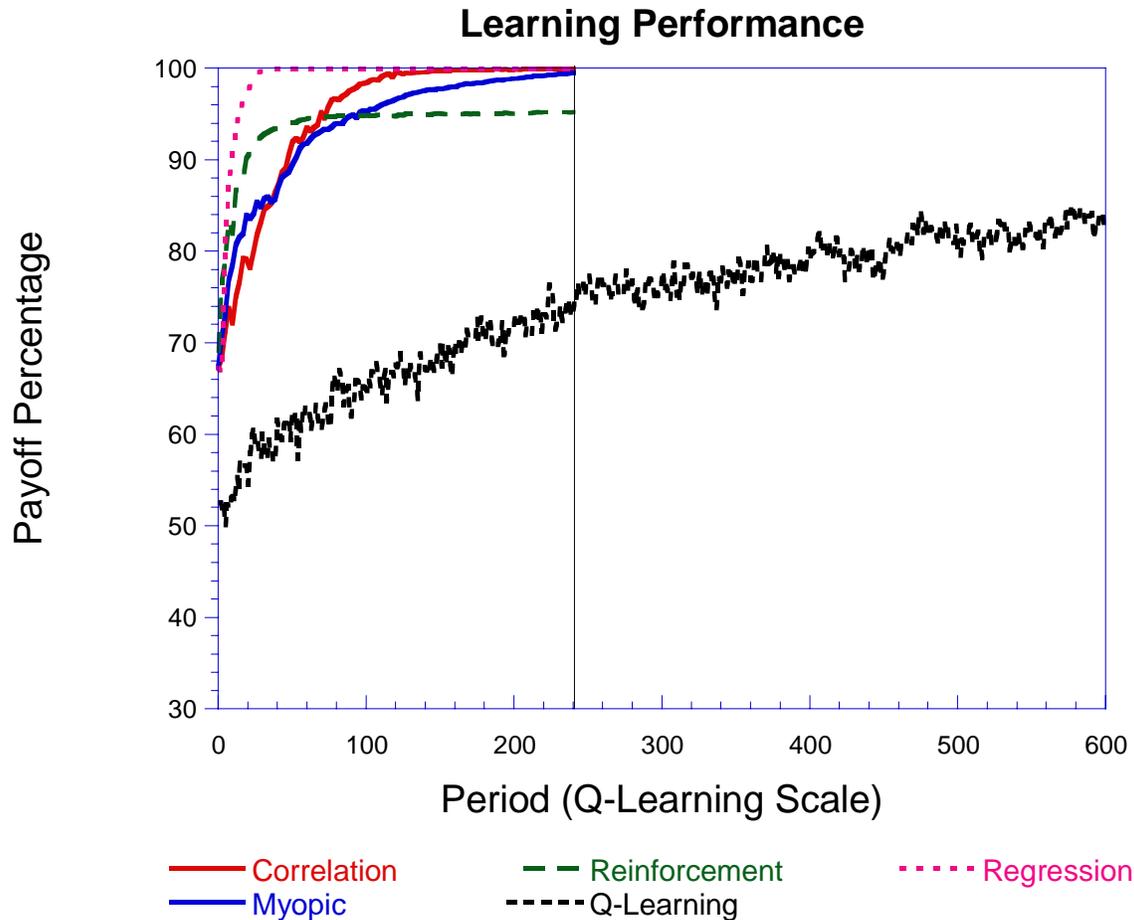
Figure S1- The behavior of Q-learning heuristic (average of 400 simulations) in the resource allocation task, no-delay condition, as compared to the other four heuristics (as reported in the paper). Note that the Time scale is adjusted for the other heuristics to represent the equivalence of data availability. Also note that by making the state-space discrete, the expected payoff of a random allocation has decreased (mainly because of the 0 payoffs in the boundaries) thus the lower start-point for the Q-learning heuristic.

As the graph shows, the Q-learning model does not find the optimum allocation even after receiving over two times the data used by other models. The behavior of the Q-learning heuristic under these conditions suggests that in order to find good policies this heuristic requires far more data than the other four models, even under no delay condition. This should come as no

great surprise since the discrete state-action space of this heuristic includes many points for which a few trials need be realized before a useful mental model of the space is built. Moreover, this heuristic does not use any information about the slope of the payoff landscape in moving to better allocation policies. Nevertheless, the amount of data needed by this heuristic puts it in a disadvantage in terms of applicability for typical organizational learning instances, where few data points are available. In fact this is the main reason we did not include this heuristic in the analysis reported in the paper.

Extension of this model to capture action-payoff delays through an enhanced representation of state-spaces is possible (See (Rahmandad 2007) for and example), however it is very costly in terms of data requirements for the model. In the simplest case, where we only can have delays of 1 period between action and payoff, our state-space will need to include not only the current location on the payoff landscape (the number of its members are $O(N^2)$), but also the last action. This is necessary because both the current state and the state from last period (which can be deduced from last action) contribute to the payoff for the current period. This additional complexity results in expansion of the state-space and the corresponding data requirement to $O(N^2.A)$, where A is the size of the potential action space (in our setting 7)[5]. Consequently to be able to learn about processes with delays of length K the heuristic needs a state-space of size

---

[5] In fact our model is limiting search to local exploration of neighboring states, and therefore size of A is limited to 7 (boundary states have smaller sizes; for a m-dimensional allocation problem, size of A becomes m*(m-1)+1; here m=3). If long-distance explorative moves were possible, as most models of organizational learning assume, the A set will expand to all the states, and will itself be of size $O(N^m)$, which increases the speed of complexity growth as a function of delays even further.

$O(N^2 . A^K)$, which is exponentially growing with the length of the maximum delay in the space, resulting in exponential growth of data requirements to learn about such payoff function.

In the very general case one can consider a maximum delay length of K and arbitrary distributed delay structure, an arbitrary m-dimensional state space where each dimension can take one of the N possible values, where exploratory actions are possible, and under an arbitrary payoff function. The "NK" type models (Kauffman 1993), widely used in organizational learning literature, are a special case of this general model (where N=2 in our terminology and no delay is present). For our general case the complexity of state-action space is of the order $O(N^m . (N^m)^K)$ where the first term, N^m, represents, in Levinthal's (2000) terms, the spatial complexity and the latter term, $(N^m)^K$, represents the temporal complexity. This clearly shows how temporal complexity quickly dominates the complexity of organizational learning for larger values of delays (K>1 period).

Simulating the Q-learning model

Two versions of the Q-learning model are posted with the supplementary material. Both are developed through Anylogic ™ software. One interface is a stand alone Java applet with accompanying files and runs under any Java enabled browser with no need to install additional software, the second includes the source code and equations of the model and can be used for detailed inspection of model implementation after installation of Anylogic software, as well as running the model.

To open the stand-alone applet, unzip the "Q-Learning_Applet_Files.zip" into one folder (all three files need be in the same folder). You can then open the "Q-Learning-Visual Applet.html" file in any web browser (e.g. Internet Explorer). You can change different model parameters and extend the simulation time as you desire.

To open the detailed Anylogic model, you need to first download and install the Anylogic software (you can get the 15-day free trial version from http://www.xjtek.com/download/). You can then open "Q-Learning-Visual.alp" in Anylogic. Different objects and modules of code are

observable on the left hand column and you can navigate through them and inspect different elements of the model. You can run the model by clicking on run button (or F5) which compiles the model and brings up a similar page as the attached Java applet. You can inspect the behavior of the model both through the interface accompanying the model and applet, or by browsing different variables and graphing them as long as you are in the run mode in Anylogic. For the second purpose, go to "root" tab in the run time mode, where you can see all model variables and can inspect their runtime behavior as well as final value.

The implemented interface allows you to observe payoff performance as well as the path of the organization on the state-space, as well as change the following aspects of the model within a simulation (just click pause, make the desired parameter changes, and click run to continue the simulation):

1- Change the simulation time to allow for more information for the learning heuristic.

2- Change exploration parameters, υ and ω, for changing the propensity of the model to explore new action-states or re-visit those already tried.

3- Change the value adjustment parameters, discount rate and strength of feedback to states that become stepping stones for better payoffs (Parameter γ in the heuristic above).

4- Change the shape of payoff function with changing the exponents of different activities' effect on payoff. Note that Cobb-Douglas function requires the exponents to add up to 1.

 You can reload the model by pressing the refresh button of your browser. You can also change the speed of the simulation through the simulation control slider on the top left corner of the applet.
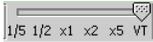
Simulating the four other learning models
The simulation model used to examine the other four learning heuristics is also posted with the online material. To open the model, download and install the Vensim Model Reader from

http://vensim.com/freedownload.html. Then open "LearningModelPresent.vmf " from Vensim

file menu. Navigate through different views of the model using the "Page Up" and "Page Down"

buttons or by the tab for selecting views at the bottom left of the screen, or by use of navigation

buttons enabled on most of the views. View the equation for each variable by selecting that

variable and clicking the "Doc" button in the left toolbar.

Note that this model includes more functionalities than discussed in the paper. It allows you to

introduce noise in payoff and perception delays, and change the payoff function to linear.

Moreover, you can change any of the model parameters and explore different scenarios with this

model.

Simulating the model:

- First choose a name for your simulation and enter it in the field for simulation name in

  the middle of the top toolbar:

- Click on the SET button to the left of this name.

- Now change the parameters of the model as desired. The best place to do so is to go to

  the "control panel" view (Press Page Down until you arrive at the second view) where

  you can change all the main parameters. The current values of the parameters are shown

  if you click on each parameter.

- Simulate the model by clicking the Run button in the top toolbar: .

Examining the behavior:

- The control panel includes graphs of the payoff, distance traveled in the space,

  convergence time, and climbing time (time during which an heuristic improves its

  performance), based on your latest simulation.

- You can also use the tools in the left toolbar to see the behavior of different variables.

  Select a variable by clicking on it and then click on the desired tool.

Additional notes on model formulations and performance metrics

- Different types of delay, including first- and third-order Erlang delays, were examined; the results were qualitatively the same.

- The model is formulated in continuous time but simulated by Euler integration with a time step of 0.125 period. Sensitivity analysis shows little sensitivity of the results to time steps < 0.2

- Note that by optimum we mean the best payoff one can achieve over the long-term by pursuing any policy which need not to be the highest possible payoff. For example, with a one period delay for activity 1 and no delay for the two others, the organization can allocate all 100 units to activity 1 during the current period and allocate all the resource between two other activities during the next period. Under these conditions, it can achieve higher than optimum payoff for the next period, at the expense of getting no payoff this period (because activities 2 and 3 receive no resources) and the period after the next (because activity 1 receives no resources in that period). A constant returns to scale payoff function prevents such policies from yielding higher payoffs than the constant allocations in the longterm.

- We consider a particular learning procedure to have converged when the variance of the payoff falls below 1% of its historical average. If later the variance increases again, to 10% of its average at the time of convergence, we reset the convergence time and keep looking for the next instance of convergence.

References

Denrell, J., C. Fang and D. A. Levinthal (2004). "From T-Mazes to labyrinths: Learning from model-based feedback." Management Science **50**(10): 1366-1378.

Kaelbling, L. P., M. L. Littman and A. W. Moore (1996). "Reinforcement learning: A survey." Journal of Artificial Intelligence Research **4**: 237-285.

Kauffman, S. A. (1993). The origins of order : self organization and selection in evolution. New York, Oxford University Press.

Levinthal, D. (2000). Organizational capabilities in complex worlds. The Nature and Dynamics of Organizational Capabilities. G. Dosi, R. Nelson and S. Winter. New York., Oxford University Press.

Rahmandad, H. (2007). "Effect of delays on complexity of organizational learning." <u>Under Review in Management Science</u>.

Sutton, R. S. and A. G. Barto (1998). <u>Reinforcement Learning: An Introduction</u>. Cambridge, The MIT Press.

Watkins, C. (1989). Learning from delayed rewards. <u>Ph.D. Thesis</u>. Cambridge, UK, Kings' College.

Watkins, C. J. C. H. and P. Dayan (1992). "Q-Learning." <u>Machine Learning</u> **8**(3-4): 279-292.