6. Conclusion

Much further research is needed on explore/exploit strategies. The issue is crucial for true autonomy—consider the Mars robot or any system in a substantially unknown environment that cannot be teleoperated. This paper suggests that strategies based on error or its rate of change can permit systems to control the tradeoff autonomously. Major desiderata now are further experiments—to build up experience with these (and other) strategies, and greater theoretical understanding of how a system can tell—for instance via better estimation techniques—how well it's doing.

References

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems*. Ph.D. Thesis, Department of Computer and Communication Sciences, The University of Michigan, Ann Arbor. Available from University Microfilms International, Ann Arbor, Michigan.

Goldberg, D. E. (1988). Probability matching, the magnitude of reinforcement, and classifier system bidding (Technical Report TCGA-88002). Tuscaloosa, AL: University of Alabama, Department of Engineering Mechanics. (Also *Machine Learning, 5*, 407-425.)

Goldberg, D. E. (l989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press. Second edition 1992, Cambridge MA: The MIT Press.

Kaelbling, L. P. (1990). *Learning in Embedded Systems*. Ph.D. Dissertation, Stanford University. Teleos TR-90-04. Also Cambridge, MA: The MIT Press/Bradford Books, 1993.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. Cambridge, MA: The MIT Press/Bradford Books.

Schmidhuber, J. (1995a). On learning how to learn learning strategies. Technical Report FKI-198-94 (revised), Fakultät für Informatik, Technische Universität München, Munich, Germany.

Schmidhuber, J. (1995b). Environment-independent reinforcement acceleration. Technical Note IDSIA-59-95, IDSIA, Lugano, Switzerland.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning,* (pp. 216-224). San Mateo, CA: Morgan Kaufmann.

Sutton, R. S. (1991). Reinforcement learning architectures for animats. In J.-A. Meyer & S. W. Wilson (eds.), *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior* (pp. 288-296). Cambridge, MA: The MIT Press/Bradford Books.

Sutton, R. S. (1996). Generalization in reinforcement learning: successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*. Cambridge, MA: The MIT Press/Bradford Books.

Thrun, S. B. (1992). The role of exploration in learning control. In D. A. White and D A. Sofge (eds.), *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Florence, Kentucky: Van Nostrand Reinhold.

Whitehead, S. D. (1991). Complexity and cooperation in Q-learning. In *Proceedings of the Eighth International Workshop on Machine Learning,* (pp. 363-367). San Mateo, CA: Morgan Kaufmann.

Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation, 3*(2): 149-175.

which the system should learn to give the correct output of the function (0 or 1) upon presentation of a random 6-bit string. The other was “Woods2”, a grid-like environment in which the system, started at a random cell, should learn to reach “food” sites in the minimum number of steps. As demonstrated in Wilson (1995), XCS does indeed learn (non-autonomously) both of these tasks when given sufficient explore trials. That is, the experimenter permits XCS *n* explore trials, then switches it to exploit trials and finds perfect performance if *n* was sufficiently (and not outrageously!) large. In addition, XCS evolves classifiers whose conditions generalize over states of the environment that have the same payoff prediction for a given action.

The current explore/exploit experiments have so far investigated versions of global strategies 4 $(p_1=f(E))$ and 6 $(p_1=f(\dot{E}))$, and local strategies 8 $(p_i=f(i,\{P_j\},\{E_j\}))$ and 9 $(p_1=f(\{E_j\}))$.

For strategy 4, p_1 was set equal to the lesser of 1.0 and the product of a gain factor times *E*, a moving average of the absolute value of the difference between predicted and actual payoff. For strategy 6, p_1 was equal to the lesser of 1.0 and the product of a gain factor times $\dot{E}$, a moving average of the absolute value of the rate of change—with explore trials—of *E*. This rate of change was calculated by a more elaborate routine which found the difference between moving averages of *E* before and after a “bout” (series, usually 100, with no intervening exploit trials) of explore trials. Then p_1 for this strategy was the probability of initiating another bout of explore trials, instead of continuing with exploit trials.

Strategy 8 was an implementation of variance-sensitive bidding (VSB). On each time-step, and for each action a_i, the system sampled a gaussian distribution with mean equal to the predicted payoff, P_i, and variance equal to the product of a gain factor and the error estimate, E_i, for that action. Then it executed the action for which the sample value was largest. In Strategy 9, the system on each time-step calculated a mean of the error estimates for each action, then set p_1 equal to the lesser of 1.0 and the product of a gain factor times the mean of the error estimates.

We are not able to include statistically reliable results, since the experiments are incomplete at the time of writing. But several qualitative observations can be made.

1. All four strategies “worked” in the sense that, starting with no prior knowledge of the environments, online performance improved from random to near optimal levels and the incidence of explore trials fell in general to low levels. However,

2. Progress was sensitive to the gain factor. Larger values caused more exploration to occur early on, when little was known, resulting in more rapid accumulation of competence but less rapid achievement of high online performance as error (or error rate of change) fell. In contrast, if the gain was lower, good on-line performance came sooner but optimal levels took longer to reach.

3. The gain setting had a further consequence. For large values, the system could be very sensitive to temporary *increases* in error (or error rate of change) once good on-line performance was achieved. Greater sensitivity meant that more explore trials would be undertaken to correct a given degree of error, resulting in a greater temporary fallback in on-line performance. In contrast, lower gain settings were less sensitive to such error “blips”.

4. Error (or error rate of change) “blips” occurred under all four strategies. Though progress toward low error via exploration is definite, it is bumpy. Assuming that the strategies are basically sound, one would like to be sure that the system’s error *measurements* are as reliable as possible.

5. There is a tension between modeling and performance reminiscent of that between exploration and exploitation. Larger gain factors—more exploration—result in greater development of generalizations within a given number of time-steps. The price for this is the on-line performance cost noted in (2) and (3).

6. Strategy 8 (VSB) implements a biased sampling of alternate actions. As suspected, this resulted in a biased generalization model in which overgeneral classifiers evolved that retained high fitness because the actions that would contradict their payoff predictions were rarely tried. There does not as yet appear to be any obvious offsetting performance superiority for VSB compared with the other strategies.

Among the strategies tested, it is perhaps most encouraging that strategy 6, which depends on the rate of change of error, is no worse than the others. This is important, because strategy 6 is the only one that is in principal unaffected by either a changing or a stochastic environment.

Broadly speaking, the explore/exploit dilemma does not go away via the strategies tested (or, surely, the others), but they may succeed in managing it “up to a constant”—the gain parameter mentioned. In a sense, the gain parameter measures the amount of *risk* the system is willing to take for potentially higher future performance, and thus summarizes the trade-off in a single number. Where does that number come from? If it is a reality in animals, it comes from evolution, which “knows about” their environments. In artificial systems it now comes from the designer, but future research may show how it can be brought within the system’s own adaptive compass.

tion "desires" (utility measures) compete to set the system's action. For example, action a_1 may have a high predicted payoff (exploitation utility), but a_2 may have a high exploration utility because it has never been tried. The balance between these "desires" is set by a task-dependent parameter Γ. Though focussed on efficiency of exploration, Thrun's framework can also be considered a (local) explore/exploit strategy, or set of strategies, providing a different approach to the objectives of the strategies of Section 3.2.

However, Thrun's article emphasizes exploration efficiency for learning a task—meaning acquiring the competence to accomplish it—after which the system will be switched to exploit mode. The present paper places the emphasis on autonomy, and the continual on-line issue of when and when not to learn in possibly changing or stochastic environments. This has in particular motivated consideration of strategies based on the rates of change of performance and error statistics.

4.2 Schmidhuber's autonomy proposal

Recently, Schmidhuber (1995a,b) introduced an intriguing, if sweeping, model for environment- and method-independent autonomous learning that purports to solve the explore/exploit problem more or less *en passant*. Central to the model is the concept of *reinforcement acceleration*. The system continually applies "policy modification processes" (PMPs) to its policy (the mapping from inputs and internal states to outputs and new internal states), and measures the resulting change in the rate of reinforcement intake. The aim of the PMPs is to achieve reinforcement acceleration, meaning that the rate of reinforcement intake since application of the PMP exceeds that since application of any preceding PMP. Upon completion of a current PMP, information sufficient to restore the previous policy, plus other information about the current PMP, is pushed onto a stack. Subsequently, the system tests to see if the current PMP satisfies the *reinforcement acceleration criterion* (RAC). If so, it is retained until the next PMP occurs. If RAC is *not* satisfied, the system pops the stack, restoring previous policies, until an older PMP is reached such that its rate of reward intake, *measured from its inception up to the current time*, exceeds that for the PMP below it on the stack. Thus the only PMP (records) maintained on the stack are those for which the reinforcement acceleration criterion is satisfied. Schmidhuber offers a proof sketch that this will be the case, independent of the "(possibly changing) environment".

The PMPs can be *any* policy-modification processes consistent with the system's architecture. For instance, for a Q-learning system based on a state-action table, a PMP could simply randomly mutate the table contents. All that is necessary is that the previous policy be restorable. Still, one worries that such a general framework, though capable of improvement and maybe eventual optimal performance, would improve impractically slowly, or would require an unacceptably deep stack. It sounds much like "random mutation with preservation of the best", a technique that experience has found impractical for most problems of interest. However, Schmidhuber presents experiments in which, though times of order 10^9 time-steps are required, substantial learning occurs. In addition, for a specific implementation of the concept, he introduces a low-level, general programing language that contains "self-referential" instructions. Schmidhuber argues, and claims his experiments show, that via these instructions, the PMPs (also expressed in the language) not only improve the system policy as predicted, but let the system adaptively determine its own rate of applying PMPs and for how long. Since to apply a PMP is in effect to engage in exploration (versus further exploitation of the policy that resulted from the previous PMP), Schmidhuber argues that the system adaptively controls the explore/exploit tradeoff.

It is possible that, besides its theoretical interest, Schmidhuber's all-encompassing approach will yield practical autonomous systems in which, as a distinct issue, the explore/exploit problem simply disappears! This will depend at least on how rapidly the PMP/RAC algorithm can bootstrap itself and also how accurately the system can in practice measure its rate of improvement in challenging environments. Meanwhile, there is a resemblance between the improvement criteria of Section 3 and Schmidhuber's RAC that suggests his theory can be an important guide to other approaches to the explore/exploit problem.

5. Experimental observations

Experiments have been conducted on several of the E/E strategies using XCS, a novel kind of classifier system in which the fitness of individual classifiers is based not on their payoff prediction as in traditional classifier systems, but on the *accuracy* of that prediction (Wilson 1995). XCS is appropriate for E/E experiments on the ten strategies because its classifiers keep both prediction and error estimates (and error rate-of-change statistics can easily be added). In addition, XCS inherently evolves classifiers whose conditions are maximally general subject to an accuracy criterion on the prediction estimate, thus making it possible to evolve a generalization model over the payoff "landscape". Finally, in contrast to earlier classifier systems that employed a roulette wheel action-selection strategy, XCS can employ any strategy of the sort represented in Section 3.

Two environments from Wilson (1995) were used. One was the "6-multiplexer", a Boolean function for is to explore to the extent that the current situation has large error estimates, but the choice of explore action is random, in order to avoid the potential sampling bias noted above.

10. $p_i = f(i,\{P_j\},\{\dot{E}_j\})$ ***or*** $p_1 = f(\{\dot{E}_j\})$

These strategies are similar to 8 and 9, but replace the dependence on E_j with dependence on $\dot{E}_j$ in order to deal with stochastic environments. In a classifier system, each classifier would keep an estimate not only of its prediction and the error in that prediction, but of the rate of change of the error, $\dot{E}_j$. That is, $\dot{E}_j$ is a moving or recency-weighted average of the difference between the current error and the error estimate. Purely stochastic effects would not show up in $\dot{E}_j$ (if the averaging period was sufficient). A large value of $\dot{E}_j$ would mean that trials of a_j in that context (x) yielded substantial change in the prediction error—thus more trials should be made there. A small value of $\dot{E}_j$ would suggest that the current value of P_j was as accurate as can be determined.

This completes the list of strategies. We turn now to three authors who have made important contributions to the explore/exploit discussion.

4. Related Work

4.1 Holland's bandit model

The operation of a genetic algorithm (GA) incorporates an explore/exploit tradeoff, since the probability that an individual will be selected for reproduction is an increasing function of its fitness relative to the mean fitness of the population. The GA searches regions of the problem search space near (with respect to genetic operators) the locations of the selected individuals. Thus selection of higher fitness individuals means searching their parts of the space more heavily, a kind of exploitation of the best. Conversely, to the extent lower fitness individuals are also selected, the GA is exploring other regions "just in case" they contain unexpectedly higher fitness individuals than currently estimated.

Holland (1975, Ch. 5) seeks to determine the degree of optimality of the GA's selection mechanism. To do this he investigates the "two-armed bandit" (two slot-machine) problem of statistical decision theory, and shows that the optimal (loss-minimizing) allocation of trials to the observed better (higher-paying) arm is very nearly an exponentially increasing function of the number allocated to the observed worse arm. Taking this solution as the optimal way to allocate trials between competing uncertain alternatives, Holland shows that, in a GA, selection proportional to relative fitness is like the derivative of an exponential and thus implies that the GA is searching near-optimally. (See De Jong (1975) for a good discussion of Holland's assumptions and derivation; also, Mitchell (1996) for a review of subsequent critiques).

How are Holland's results useful for autonomous learning? Only indirectly. For one thing, the optimal allocation for the bandit is based on knowledge of the "observed best" alternative, which can't be known for sure until *after* all trials are over, when it's too late to perform the allocation. How to allocate trials as one goes along is less understood, though De Jong (p. 33) does simulations suggesting that allocation with probability proportional to the product of relative payoff and the *previous* time-step's allocation probability (which parallels the way the GA works) can approach the bandit optimum for large numbers of trials. In the present context, this idea might be applied by modifying strategy 7 to make each selection probability proportional to the current prediction times the probability of that selection the last time around (implying storage of the latter).

The major problem, however, is that Holland's allocation formula contains parameters dependent on the statistics of the bandit process, including the means for the two arms, the variances, and (in an improvement of the formula (Holland, 1992)), higher moments, apparently. Although the parameters are not required in order to know that the form is exponential, they *are* required to actually carry out the allocation—even ex post facto—of the trials! In contrast, the adaptive strategies discussed in Section 3 make no assumptions about payoff statistics, but instead attempt to estimate both the means and in some cases the variances (errors). Though the strategies have as yet only scant connection to theory, they are practical for autonomous learning in a way that the bandit results are not.

4.2 Thrun on exploration efficiency

The present article overlaps in several respects with Thrun (1992), in which the emphasis is on exploration techniques that are efficient in the sense of minimizing the costs (time-steps, negative payoffs, etc.) of learning an environment to a given performance criterion. Thrun distinguishes *undirected* exploration—in which actions are selected either uniform randomly as in strategy 1, or with probabilities related to action utilities as in strategy 7—from *directed* exploration, which aims to choose actions that will maximize information gained, as in Kaelbling's (1990) interval estimation or Sutton's (1990) recency-based exploration. The discussion of these exploration techniques is very comprehensive.

Thrun's thesis, demonstrated in two (simulated) robotics experiments, is that efficient exploration benefits greatly not only from the use of directed techniques, but also from the right amount of exploitation—not so much as to waste time exploring, but not so little as to take too long to learn the task. His demonstrations involve a framework in which exploration and exploita-

In $f(\dot{E})$, the system's decision is based on its estimate of $\dot{E}$, which should be subject to fluctuations similar to E, but less subject than estimates of reward and reward change averages. Parametrization of $f(\dot{E})$ should be similar to $f(E)$.

3.2 Local strategies

The next four strategies are termed *local*. In them, the degree of exploration is based on a function of quantities associated with the system's response to the current input. The global adaptive strategies sense a condition that is a property of the whole system-environment interaction, then set the system's general explore probability accordingly. In the local (adaptive) strategies, the system senses a condition that is a property of a *niche* in its interaction with the environment, and chooses its action accordingly. The idea is that learning may be needed in certain input situations but not in others, or, more generally, that statistics associated with a given niche should determine the degree of exploration that takes place there.

7. $p_i = f(i,\{P_j\})$

In our assumed reinforcement learning model, each time the system receives an input x it responds with a set $\{P_j\}$ of predictions of the payoff available for each action that it might take, then selects one of the actions, which we denote a_i. The set $\{P_j\}$ might be the output of a neural network (or set of networks), a classifier system, or a table, etc. Each prediction is a statistic, based on prior experience in situations (apparently) like the current one. The general idea of the $p_i = f(i,\{P_j\})$ strategy is that actions with higher associated predictions should be selected with higher probability.

A well-known example is the so-called "roulette-wheel" selection employed in some classifier systems (Goldberg 1989). For each classifier that matches the current input, the probability that its action will be selected as the system's action is just the classifier's prediction of payoff divided by the sum of the predictions of payoff of all the matching classifiers. That is, the predictions are normalized and treated as selection probabilities. Roulette-wheel selection achieves a mix between exploration and exploitation. Higher predicting actions tend to be selected over lower predicting ones, but the latter still continue to be tried.

The $f(i,\{P_j\})$ strategy is a compromise, a bit like the constant strategy (strategy 1). It can achieve fairly good performance and, if the environment changes, it will adjust its predictions and reach fair performance anew. The drawback is that even though the system can acquire full competence, it can never realize that competence in performance, since suboptimal actions continue to have a finite probability of selection.

8. $p_i = f(i,\{P_j\},\{E_j\})$

In this strategy the probability of selecting action a_i on a given time-step depends on both the prediction for that action and the estimated error in the prediction. The idea is to select the action for which the sum (or other increasing function) of the prediction and error is high. Then, as further exploration refines the predictions and reduces their error, the system will likely end up always selecting the action for which the prediction is in fact highest. Thus, as exploration occurs and establishes true values for the predictions, the system will in effect gradually switch over to full exploitation.

An $f(i,\{P_j\},\{E_j\})$ strategy was first implemented for learning based on tables by Kaelbling (1990), who termed it the "interval estimation" strategy for its relationship to a method in statistics. Goldberg (1988) proposed a similar strategy for classifier systems called "variance-sensitive bidding", in which the system samples, for each action, a gaussian probability distribution with mean P_j and variance gE_j, and chooses the action with the largest sample value. The "gain" g determines the amount of bias toward exploration.

These strategies have the ability to sense the degree of convergence to reliable predictions and to increase exploitation. Due to their sensitivity to error, they will track a changing environment. For the same reason, they will waste trials if the environment is stochastic. As a practical matter, it should be noted that the strategies are most readily implemented in classifier systems or tables, since both error and prediction statistics are required. Because a classifier is a "record" structure like a table entry, it can easily accomodate an additional "slot" for the error. Neural networks, on the other hand, have no obvious way to estimate errors apart from doubling the size of the network or adding a second one devoted to that purpose.

Besides wasting trials in a stochastic environment, a further potential drawback of the $f(i,\{P_j\},\{E_j\})$ strategy arises because it inherently samples actions with a bias toward higher predicting ones. If the system (as in neural networks and recent classifier systems) tends to form generalizations over the input space, biased sampling can result in persistent overgeneralizations, simply because potential counterexamples are not tried, or tried too little. In principle, the next strategy avoids this problem.

9. $p_1 = f(\{E_j\})$

In this strategy, as above, the E_j are the error estimates associated with the actions available for the current input x. The explore probability, however, as in the global strategies, is simply the probability of selecting one of the actions at random (versus choosing the action with highest P_j). The strategy would be implemented by basing p_1 on the sum of the E_j, or their average. The idea such as a negative exponential; it can also be a step, where p_1 has a high value such as 1.0 for a finite period τ, with value 0.0 afterward. For the strategy to be efficient, parameters must be chosen so that enough exploration occurs to achieve competence, but not so much as to continue exploration long after it ceases to be needed. If sufficient exploration occurs, the system can reach optimal performance; otherwise, it can't. A further weakness of the *f(t)* strategy is that if the environment changes, the system cannot re-learn.

As with the constant strategy, the system's learning decision is simple, given by *f(t)*. Similarly, success depends on considerable external knowledge.

3. $p_1 = f(R)$.

This is the first of the adaptive strategies. The explore probability depends on a moving or recency weighted average, *R*, of the system's reward on each time-step, *r*. The idea is simply that if you know how well you can possibly do in an environment, then as you reach that level, you should stop exploring. If the strategy can be correctly parametrized (based on the maximum possible *R*) then, as long as the environment doesn't change, the strategy should be successful. If the environment does change, then the parametrization may be wrong, and the system will perform suboptimally.

In the *f(R)* strategy, the system makes its own decision based on its estimate of *R*. The estimate may be subject to considerable fluctuation, since reward may be sparse and the reward stream is dependent on the system's "location" in the environment. On the other hand, long estimation times for *R* may make the strategy insensitive. Further, parametrizing the strategy depends on knowledge of the maximum *R* for the given environment. For reasonably complex environments, this information may be unavailable.

4. $p_1 = f(E)$.

In this strategy, the explore probability depends on the system's average prediction error, *E*. For reinforcement learning systems, the prediction error is the difference between the system's prediction of payoff *P*, and that actually received (recall that *P* combines current external reward and an estimate of future reward). The value of *f(E)* should decline with *E*, so that the system reduces its exploration rate as the error goes down. This strategy in principle improves on the *f(R)* strategy since it is not dependent on knowing the maximum *R*. Furthermore, if the environment changes, *E* will presumably go up, resulting in desirable renewed exploration. The basic idea of the strategy is that if the system's predictions are good, exploration is not needed; if they are not good, there is something further to learn.

Strategy *f(E)* is the first to be significantly affected if the environment is *stochastic* (or non-deterministic), meaning that the next values of *x* and *r* are not a deterministic function of the current *x* and *a*. As noted earlier, the condition can arise simply because of noise, for instance in the sensors. It can also arise if the environment is noiseless but non-Markov. In any event, strategy *f(E)* is "fooled" into over-exploring if the environment is stochastic, since the stochastic fluctuations—which cannot be reduced—will cause exploration even after the system has reached maximum competence. Note that making $f(E) = 0$ for *E* less then a threshold θ is not a general solution, since if θ is too large, the system may stop exploring before maximum competence is reached.

In the *f(E)* strategy, the system's explore decision depends on its estimate of *E*, which should be subject to less fluctuation than *R*. Furthermore, apart from the stochastic problem just mentioned, parametrizing the strategy is simpler than for *f(R)* since the optimum value of *E* is known: it is zero.

5. $p_1 = f(\dot{R})$.

This strategy bases the explore probability on the rate of change of average reward *with explore trials*, $\dot{R}$. Suppose the system measures its average reward on exploit trials, then does some explore trials, then measures average reward again. If there is little difference in average reward, then there's not much point in doing more explore trials. If there is a big difference in average reward, then further exploration is in order, because there is probably more to learn (note that the difference can be either positive or negative). The $f(\dot{R})$ strategy improves on *f(R)* in that it will "track" a changing environment. Compared with *f(E)*, it should be unaffected by a stochastic environment, since reward variation due to stochastic effects alone will average zero.

The system's explore decision in the $f(\dot{R})$ strategy depends on its estimate of $\dot{R}$, which may be subject to fluctuations similar to *R*. Parametrizing the strategy is simpler than for *f(R)*, since the "target" value of $\dot{R}$, zero, is known. However, coefficients in $f(\dot{R})$ would still be expected to depend on the maximum *R*, which, as noted earlier, may be unknown.

6. $p_1 = f(\dot{E})$.

Here, the explore probability depends on the rate of change of the system's average prediction error *with explore trials*, $\dot{E}$. The strategy attempts to overcome the drawbacks of the three previous adaptive strategies. It will track a changing environment, and it will not be affected by a stochastic environment if averaging periods are adequate. The basic idea is that if further explore trials do not change the average prediction error, then there is no reason to do more explore trials. If average prediction error does change, it must be because there was more to learn, either because the present environment is insufficiently known, or has changed.

To learn the value of actions, we shall assume the system learns a mapping X x A → P, where the P, which are *payoff predictions*, generally combine the reward *r* expected on the current time-step with an estimate of future reward to be received if the associated action is taken. For non-Markov (not predictable based on immediate sensory information) environments, the system may incorporate some degree of internal state (temporary memory) which when its value is combined with the external stimulus, effectively makes the decision problem Markov. For convenience we shall assume the environment is Markov unless otherwise noted.

Part of the system's job is to learn the X x A → P mapping. The other part is to use the mapping to choose actions that maximize the reward measure. Given an input *x*, the system may sometimes choose the action *a* that maps to the highest prediction; this will be termed *exploitation* (of current information). At other times it may choose some other action—termed *exploration*—in an effort to gain new information.

Uncertainty in the current prediction for an action can arise from several sources: (1) the estimation process involves gradual (conservative) adjustments, and the prediction has simply not been sufficiently sampled; (2) there may be *no* prediction yet, i.e., the action has never been tried; (3) if, as with neural networks or classifier systems, the system can generalize, fluctuations in the prediction may occur as the system settles on its "model" of the environment; (4) inputs, actions, or rewards may be affected by noise; (5) the environment may be non-Markov, so it appears to be non-deterministic. (If the environment has either of the last two sources of uncertainty, it will be termed "stochastic".) When uncertainty is high and the payoff mapping is poorly known, the system should evidently explore more than it exploits. Later, it should exploit more than explore. Choosing just when to do each is the essence of the explore/exploit dilemma as it occurs in the framework of reinforcement learning.

3. Explore/Exploit Strategies

If a system has a well-defined way of making the explore/exploit decision at each time-step, we shall say it has an E/E *strategy*. This section lists ten distinct strategies; some are often seen in the literature, others not so often or not at all. For consistency of viewpoint, the six "global" strategies discussed first are characterized in terms of a function $f(\cdot)$ which gives the probability p_1 that the system will execute a random explore trial on the current time-step, that is, the probability that it will make a uniform random selection from its set of available actions. In contrast, $p_2 = 1\text{-}f(\cdot)$ is the probability that the system will execute an exploit trial, i.e., will execute the action with the highest predicted payoff. The explore decision is made uniformly randomly for simplicity. Biasing the choice is of interest (and essential in non-toy problems (Whitehead 1991, Thrun 1992)), but this variation will be discussed later in connection with the "local" strategies where it makes most sense.

For any specific system and environment, the selection of an E/E strategy and its parametrization are naturally dependent on constraints that might include (1) maintaining system "energy" (cumulative reward less cost of operation) above a threshold; (2) maximizing some measure of reward received; and (3) operating for a finite period, or "lifetime", *T*. The interaction of strategies and constraints is complex and beyond the present scope, but some attempt will be made here to indicate their relationship. Note also that strategies below can always be combined to form various hybrids.

3.1 Global strategies

By "global" is meant a strategy where the argument of $f(\cdot)$ is a quantity independent of the system, such as time, or a statistic of the system's overall behavior, such as a moving average of its rate of reward intake. In the former case, the strategies do not change with the system's experience and are in that sense non-adaptive or fixed; in the latter, they are adaptive.

1. p_1 = constant.

In this strategy, the explore probability is chosen small enough to permit adequate on-line performance, yet large enough to allow fairly rapid initial learning; a value of 0.1 is typical (see, e.g., Sutton 1996). Because the system never stops exploring, it will automatically re-learn if the environment changes (e.g., if the environment's reward structure changes, or the system wanders off into substantially new territory). The main drawback of the constant strategy is that while the system can acquire complete knowledge of the environment and thus the *competence* to perform optimally, it can never actually do so, since it is required to execute explore trials at a non-zero rate. Thus, a fixed fraction of the time, the system sacrifices the difference in return between acting optimally and acting randomly; for large lifetime *T*, the total loss can be great.

In the constant strategy, the system's decision of "when to learn" is simple: it does explore trials with a fixed probability. However, the specific choice of that probability is made externally, and depends on knowledge of the environment, task, and *T*.

2. $p_1 = f(t)$.

In this strategy, $f(t)$ is typically a descending function of time, either going to zero assymptotically or after a finite period. The idea is to have a large fraction of explore trials early, when there is much to learn and little competence, followed by a shift to exploit trials later. Usually, $f(t)$ is a relaxation or annealing-like function

To be published in *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*,
P. Maes, M. Mataric, J. Pollack, J.-A. Meyer, and S. Wilson (eds.), Cambridge, MA: The MIT Press/Bradford Books (1996).

Explore/Exploit Strategies in Autonomy

Stewart W. Wilson
The Rowland Institute for Science
100 Edwin H. Land Blvd.
Cambridge, MA 02142 USA
(wilson@smith.rowland.org)

Abstract

Within a reinforcement learning framework, ten strategies for autonomous control of the explore/exploit decision are reviewed, with observations from initial experiments on four of them. Control based on prediction error or its rate of change appears promising. Connections are made with explore/exploit work by Holland (1975), Thrun (1992), and Schmidhuber (1995a,b).

1. Introduction

Research on animats or "autonomous adaptive agents", whether simulated or realized as robots, has made progress on several fronts. A variety of core learning techniques exist for associating input stimuli with output actions to produce reasonable performance; underlying architectures include networks, CMACs, classifier systems, and tabular methods. There is also progress in incorporating generalization, internal state, and predictive modeling, among other aspects of intelligence. However, autonomy per se has not received great attention. Broadly, for an adaptive system to be autonomous implies it can cope or survive in an incompletely known and uncertain environment, perhaps accomplishing a task, independently of external control. To do well, the system must continually decide *when* and *when not* to learn. True autonomy implies that the system makes this decision on its own, according to its experiences.

The decision to learn is fundamentally a choice between acting based on the best information currently possessed versus acting other than according to what is apparently best—i.e., most remunerative—in order perhaps to gain new information that may permit higher levels of performance later. Learning risks a short-term cost—the "opportunity cost" of not doing the apparent best—in order to achieve higher returns in the longer run. Not learning risks those potentially higher returns in order to get known benefits now. The tension between learning and performance is often described as the "explore/exploit dilemma". Holland (1975) was one of the first to discuss the dilemma in connection with adaptive systems. He summarizes: "[obtaining] more information means a performance loss, while exploitation of the observed best runs the risk of error perpetuated". The dilemma crops up as soon as a learning system is supposed to have some degree of autonomy. However, because workers needed to focus on more basic questions, the explore/exploit issue was generally deferred. Now, due to the progress noted above, autonomy can move up the agenda, and with it systematic investigation of explore/exploit.

This paper will not provide any "magic bullets" for what is probably a fact of autonomous adaptive life. Instead it will discuss the advantages and limitations of a number of approaches to explore/exploit ("E/E") in an effort to identify those that most reduce the need for prior information and external decisions—and so increase autonomy. The next section states the incremental, reinforcement learning framework of the discussion. Section 3 describes ten progressively more sophisticated E/E "strategies". Section 4 relates the present work to Holland's (1975) "bandit" discussion, to Thrun's (1992) research on exploration efficiency, and to Schmidhuber's (1995a,b) work on autonomy. Section 5 offers some rough observations from the experimental program that has been undertaken. Finally, Section 6 concludes by suggesting that strategies based on error or its rate of change have promise, but that the biggest need is for further experiments and better methods of self-estimation of performance statistics.

2. Background Assumptions

The explore/exploit dilemma will be examined within the basic framework of reinforcement learning (Sutton 1991). On each discrete time-step, we assume that the system receives a stimulus vector x (one of a large set X of such vectors) from the environment, carries out an action a (one of its own set A of available actions), and receives from the environment a reward r (which varies with x and a and may often be zero). The system's objective is two-fold. On one hand, it must learn which actions maximize some measure of the reward received over time, such as the sum or a discounted sum of the rewards. On the other hand, it must act so as to accomplish the maximization.