Complex Systems **2** (1989) 705–723

Bid Competition and Specificity Reconsidered

Stewart W. Wilson*
*The Rowland Institute for Science,
100 Cambridge Parkway, Cambridge, MA 02142, USA*

Abstract. Experiments were conducted with respect to two classifier system mechanisms: the bid competition and the use of classifier specificity in bidding and payments. The experiments employed a simplified classifier system and so may not accurately reflect the behavior of the standard system. Nevertheless, the results indicated that, in general, (1) specificity should not be factored into amounts deducted from a classifier's strength, (2) the bid competition does not improve performance and does not encourage default hierarchies, and (3) default hierarchies will form under a somewhat different algorithm than the standard one.

1. Introduction

Classifier systems (Holland [1,2]) are inductive systems in which a population of condition-action rules encoded as bit-strings evolves through time so as to maximize reinforcement, or *payoff*, when the system responds to stimuli from the environment. The rules, called *classifiers*, have a condition part consisting of one or more *taxa* from $\{1, 0, \#\}^L$, where # is a "don't-care" symbol; an action part consisting of a *message* from $\{1, 0\}^L$; and finally a *strength* parameter that is adjusted to reflect the classifier's usefulness in obtaining payoff. Improved rules are discovered and multiplied, and unpromising rules discarded, under a *genetic algorithm* that treats strengths as fitnesses and applies operators patterned after those of natural genetics. Classifier systems have relatively simple components — the classifiers — but because these pass messages and are in a quasi-Darwinian competition, the overall behavior can be complex.

The classifier system idea has stimulated a number of investigations of its merit for inductive machine learning (e.g., [3–11]). Although each investigation retained most of the elements of Holland's model, a number of modifications were explored. In particular, Wilson [6,7] exhibited successful systems that omitted two basic elements from the reinforcement algorithm: (1) The

*Computer address (Arpanet): `wilson@think.com`.

restriction of the system's controlling classifiers to a subset — selected by a bidding competition — of all matching classifiers; (2) The factoring of a classifier's specificity into its bid and its payments to other classifiers. Instead, in Wilson's programs, the step of computing the subset — sometimes called the "active" set — is simply left out; and all bid-like quantities and payments are based solely on classifier strength.

The modifications were made primarily for reasons of simplification. However, it was also felt that the narrowing of control implied by the active set competition might serve no essential purpose. The use of specificity — a formal quantity — in bidding and payments was thought to be at odds with the notion that a classifier's fitness should depend only on its ability to obtain payoff. At the time, no comparisons with and without the modifications were made: the programs worked quite well as modified. In this article, we present research in which, using Wilson's **Boole** program [7], effects of active set competition and of specificity were investigated. Because the results appear to raise significant questions for classifier systems in general, we begin by reviewing the reinforcement algorithm of the standard system.

2. The standard approach

For convenience, we shall refer to Holland's [2] classifier system as the "standard" system. Holland, Holyoak, Nisbett, and Thagard [12] adds to the standard — notably the concept of "support" — but that system will not be discussed since there are no published reports of experimental tests. Holland [13] and Riolo [10] suggest certain changes in the standard that will be discussed later.

In each cycle of the standard's reinforcement, or *bucket-brigade*, algorithm:

1. Messages from the input interface are added to the current message list, and a *match set* [M] is computed consisting of all classifiers whose conditions are satisfied by messages on the list.

2. Each classifier in [M] makes a *bid* B equal to the product of its *strength* S, its *specificity* σ (the number of non-#'s in its condition part divided by the condition's length), and a small constant c like 0.1.

3. A subset of [M] is chosen consisting of the high bidders. We shall refer to this subset as [MM] or the "active" set. The competition for inclusion in [MM] is usually made stochastic such that classifiers having high bids are more likely to win than lower bidders. Usually, [MM] has a fixed maximum size; if [M] is smaller than this, all members of [M] go into [MM].

4. Each classifier in [MM] has its strength reduced by the amount of its bid. The classifiers that sent the messages matched by a member of [MM] have their strengths increased by the amount of the member's bid (it is usually divided among them).

5. [MM]'s classifiers place their messages on the message list (the old messages are erased).
6. Any posted messages that would trigger the output interface do so, but with resolution of effector conflicts if necessary.
7.
    (a) If external payoff enters the system, it is shared by all members of [MM].
    There is some uncertainty about step 7a. A number of workers (e.g., Goldberg [5]) have felt that step 7a is insufficiently discriminating and instead have implemented the following step, which we shall consider as replacing 7a:
    (b) If, following an effector action, external payoff enters the system, it is shared by the members of [MM] that advocated that action. In addition, Holland [2] suggests that the whole payoff amount be paid to each of [MM]'s classifiers, but Holland [14] advises sharing the payoff among them. To test whether sharing or non-sharing is better, we performed an auxiliary experiment. Because the results were clearly in favor of sharing, sharing is used in step 7b.

The experiments in this article were done not with the standard, but with variants of **Boole**, a program that generates an external action (a decision) in one time-step and has no message list; it thus differs importantly from the standard. However, **Boole** is the same as the standard in many ways, and its behavior has the experimental advantage of being simpler to understand. Our results provide information about the standard, though indirectly. The next section introduces **Boole** and describes **Boole**-1, the first variant used in the experiments and the one most closely related to the standard system.

3. Boole

In general, **Boole** sees a binary input string and in one time-step produces an action, 1 or 0, indicating whether or not it thinks the string belongs to the concept (usually a Boolean function) that it is being given feedback to learn. **Boole**'s classifiers are simple: the condition is a single taxon of the same length as the input string; the action is either 1 or 0. The particular reinforcement algorithm used initially in the experiments was as follows:

Algorithm Boole-1

1. A match set [M] is computed consisting of all classifiers whose condition is satisfied by the input string. [Analogous to step (1) of the standard.]
2. Each classifier in [M] makes a bid B equal to the product of its strength S, its specificity σ, and a small constant c like 0.1. [Same as the standard; our original work with **Boole**, as noted, did not factor specificity into the bid].

3. A subset of [M], termed [MM] or the "active set", is chosen stochastically such that inclusion in [MM] is more likely for higher bidders. [Same as the standard; the original **Boole** did not form [MM]].
4. Each classifier in [MM] has its strength reduced by the amount of its bid—termed the classifier's *payout*. [Since there is no message list here, the payouts have no recipients and so vanish from the system. This difference does not significantly affect our comparison, since the paying classifiers pay out identically under both systems.]
5. [No messages are posted, since there is no message list.]
6. The system makes its decision (1 or 0) by selecting a classifier from [MM] with probability proportional to classifier bids, and outputting the selected classifier's action as the decision. [This step is analogous to conflict resolution in the standard.]
7. If, following the decision, external payoff enters the system, it is shared by the members of [MM] that advocated that action. [Same as step 7b above.]

In the standard algorithm, an indefinite number of system cycles may occur before an external action is generated. In all such cycles but the final one, messages are posted to the message list and matched, but the messages are "internal" in the sense of not triggering any external effector. Such cycles — call them internal cycles — are absent from **Boole**, which chooses an external action on every cycle and does not post internal messages.

Despite this difference, many results with **Boole** should carry over to the standard. The reason is that an internal cycle of the standard closely resembles **Boole**'s single cycle with respect to: (1) matching, (2) formation of bids, (3) competition for activation, and (4) payout from a classifier's strength. There is a difference in that, on an internal cycle, a classifier of the standard system shares payoff that comes from another classifier, whereas in **Boole**, the shared payoff always comes from outside. The exact source of payoff, however, does not seem critical to questions about specificity and active set size.

4. Experiments with Boole

For all experiments, the initial population [P] contained 400 classifiers with alleles generated from a uniform random distribution; initial strengths were set to 100. In Step 7 of the reinforcement algorithm, the payoff for a correct answer was 1000; for a wrong answer it was 0. The constant factor c (step 2) was set at 0.1. Specificity was calculated as described, except that, to permit them to bid, the specificity of classifiers whose taxa consisted entirely of #'s was defined to be 0.1 instead of 0. The discovery component, employing the genetic algorithm, was operated essentially as described in [7], and without significant difference from the discovery component of the standard system.

It produced offspring at the rate of one per performance cycle, i.e., per input string presentation and decision. Crossover was performed between two offspring with probability 0.12. An offspring allele was mutated with probability 0.001. These rates had been found to produce good learning in the previous work.

The learning task in all experiments was the 6-bit Boolean multiplexer [15], a rather intricate function whose disjunctive normal form is as follows:

$$F = x'_0x'_1x_2 + x'_0x_1x_3 + x_0x'_1x_4 + x_0x_1x_5,$$

where $x_0 \ldots x_5$ are the left-to-right bits of the input string. In each experiment, input strings were generated randomly and presented to the system. The system's decision was correct if it equaled the value of F for that input string, otherwise not.

One measure of system performance that was used was the percentage of correct decisions over the preceding 50 trials—termed *average score*. The other measure used had more to do with the content of what was being learned. It was called the *solution count*, and measured the percentage of the population that consisted of instances of the following set of classifiers (the "/" separates taxon from action):

0	0	0	#	#	#	/	0
0	0	1	#	#	#	/	1
0	1	#	0	#	#	/	0
0	1	#	1	#	#	/	1
1	0	#	#	0	#	/	0
1	0	#	#	1	#	/	1
1	1	#	#	#	0	/	0
1	1	#	#	#	1	/	1

Examination will show that each matches exactly eight of the 64 possible input strings and recommends the right answer each time. Furthermore, the eight matched sets are disjoint—they partition the input space. There does not seem to exist a more efficient, complete set of "solution" classifiers. We named the set [S6] and used the percentage of its instances in [P] as a measure of the system's progress in evolving correct, maximal generalizations.

4.1 Experiment 1

4.1.1 Procedure and results

The first experiment used the above **Boole**-1 algorithm, that is, the algorithm deemed closest to the standard. The idea was to perform such an experiment as a baseline, then, with modifications, to proceed to further experiments that the results would indicate. Experiment 1's further purpose was to investigate the effect of active set size by varying the maximum number of classifiers allowed (in step 3 of **Boole** -1) to become members of [MM]. Starting with the same random population, separate experiments

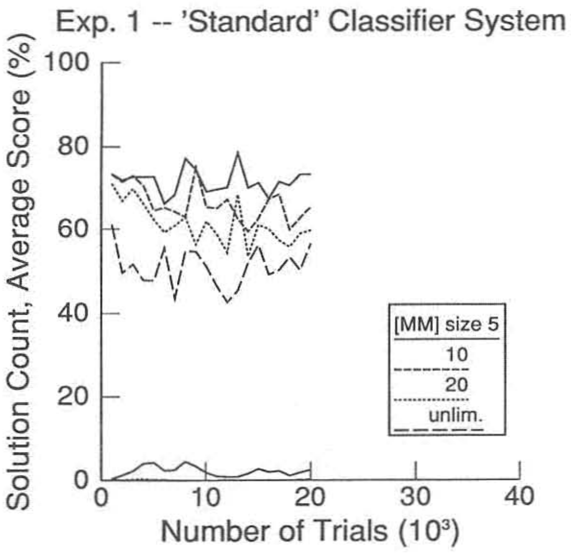

Figure 1: Results of experiments under algorithm **Boole**-1, the "standard", for different active set ([MM]) sizes. Upper four curves represent moving averages of percentage correct decisions over past 50 trials for [MM] sizes as shown. The curve at bottom is the solution count, or percentage of the population consisting of instances of the "solution set" [S6], for [MM] equal to 5. For other [MM] sizes, solution counts were negligible or zero.

were carried out in which [MM]'s maximum size was, respectively, 5, 10, 20, and unlimited. Then the series was performed again using a different initial population. Each experiment ran to $20,000$ problems (trials). For each setting of [MM]'s size, the average of the results of the two runs was plotted in Figure 1.

Learning under the **Boole**-1 algorithm was decidedly poor. The average score did not get above 80% for any setting of [MM]'s size and in fact stayed essentially flat in each plot after 1000 problems. There was a trend toward higher average scores with smaller [MM] size. In addition, a few solution classifiers were evolved at the smallest [MM] size. Examination of the final populations revealed heavy overgeneralization. Under small [MM] size, among the strongest classifiers were some that were correct three-fourths or five-eighths of the times they matched: classifiers like 0 # # 0 # # / 0 and # # # # # 0 / 0, respectively. Other classifiers were either correct half the time, such as 1 # # # # # / 0, or wrong more than half the time, such as # # 1 # # # / 0. In general, the populations contained almost no perfectly correct classifiers, either members of the solution set, or more specific than those. The two runs with unlimited active set size produced almost totally degenerate populations: one ended up with 398 instances of # # # # # # / 1 and two instances of # # # # 0 # / 1.

4.1.2 Discussion

Under our assumption that **Boole**-1 can tell us about the standard algorithm, the results of this experiment suggest that rampant overgeneralization and consequent mediocre performance may occur under the standard algorithm. There has been some informal indication (e.g., [13]), that this can be the case. Here, we should like to pin down just what factor is causing the overgeneralization and find the simplest way to correct it.

Goldberg [5] pointed out that under steady-state payoff conditions, the strength of a classifier would approach (in the present notation) $S = R/c\sigma$, where R is the mean payoff to the classifier. This suggests that if two classifiers C1 and C2 receive equal mean payoffs, their steady-state strengths will nevertheless be unequal if their specificities differ. In particular, the more general classifier will be stronger by the ratio of the two specificities. While that implies a bias in favor of generalists, it is still not clear why the bias should tend to produce a rout.

The explanation may be as follows. Suppose that C2 is a more specific version of C1 (that is, C1 with some #'s switched to 1 or 0). There will then be situations in which C1 and C2 share payoff. Suppose, for simplicity, that no other classifiers participate in these situations. Then C1 and C2 will each get half of the payoff, i.e., R is the same for both of them. From the last paragraph, however, C1 will tend toward a higher steady-state strength, which must result in C1's having more offspring than C2. Consequently, on the next occasion that C1 and C2 share payoff, it is more likely they will be joined by a second copy of C1 than by a second copy of C2. But this means that C2 will get only one-third of the available payoff, instead of half of it, while the two copies of C1 together get two-thirds. The change in the payoffs further alters the strengths in favor of the more general concept, and the cycle repeats. Eventually C1 and its offspring drive out C2 and its. C1 can even afford to be overgeneral — that is, wrong in some (other) situations — if the additional #'s increase its margin over C2 by more than the cost of the errors.

We hypothesize that a competitive mechanism of this sort is responsible for the overgeneralization in Experiment 1. The basic cause is that specificity, a formal factor, intervenes between a classifier's mean payoff and its steady-state strength — so that, as a result of reproduction, payoff and strength cannot form a stable relationship. The extent of the overgeneralization will depend on the degree to which a gain in numbers for the C1 concept can take payoff away from the C2 concept. Note, however, that if the size of the active set [MM] is N, the payoff ratio in favor of C1 cannot exceed N to one. Consequently, less overgeneralization should occur when [MM]'s size is small — as indeed the experiment indicates.

What is the remedy for the overgeneralization? Certainly it is not to restrict [MM]'s size, for learning is still poor when the size is 5 (further experiments showed no improvement for smaller values). In addition, classifier systems' emphasis on concurrent processing would not be served by reducing the size. The right solution would seem to be to change the way specificity is handled. In particular, we shall retain specificity in the bid calculation (step 2), but eliminate it from the calculation of a classifier's payout (step 4). The bid *per se* affects the probability of entry into the active set and has no direct effect on strength. In contrast, if specificity is eliminated from a classifier's payout, then Goldberg's equation becomes $S = R/c$ and strengths will be stably related to payoffs. The changed algorithm is as follows:

Algorithm Boole-2

Same as **Boole**-1, except step 4 now reads:

4. Each classifier in [MM] has its strength reduced by a payout equal to c times the strength (where c, as before, is a small constant like 0.1).

4.2 Experiment 2

4.2.1 Procedure and results

This experiment used **Boole**-2. All parameter values were the same as in Experiment 1. Five runs for each setting of [MM]'s size were performed (each run used a different random initial population); the results were averaged and are plotted in Figure 2. There is a striking difference from Figure 1. The upper set of curves plots average score, which, except when [MM] size is 5, rises within 10,000 problems above 90%. The lower set of curves shows strong rises in the solution count, in contrast to no such rise in Experiment 1. Clearly, leaving specificity out of the payout calculation — the sole difference between the two experiments — has had a major effect.

Figure 2 also exhibits a reversal in the trend with [MM] size: the worst curves occur for size 5, the best for the case when the active set size is not limited. The implication, at least from this experiment, is that in terms of performance and discovery of solution classifiers, step 3 of the reinforcement algorithm does not buy anything. That is, no purpose seems to be served by restricting the system's controlling classifiers to a high-bidding subset of the match set; instead, it appears better to eliminate the bid competition and let all members of the match set participate in the decision. Extended to the standard classifier system, this result suggests that all matching classifiers should get to post their messages.

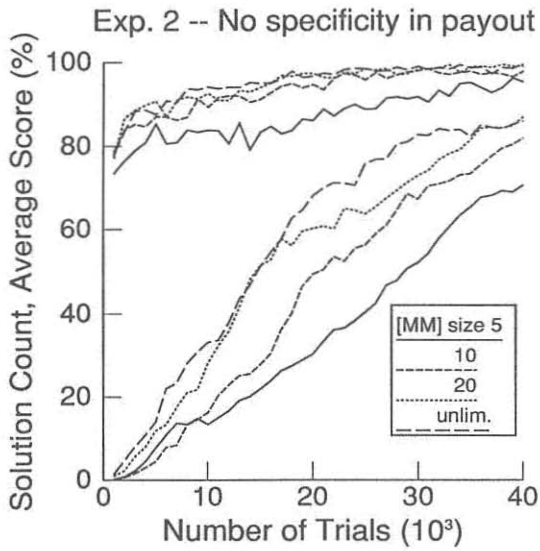

Figure 2: Results of experiments under algorithm **Boole**-2, which omitted specificity from payout, for different active set sizes. As in Figure 1, upper four curves represent average score; lower curves are solution count.

4.2.2 Discussion

Why does small [MM] size slow down learning? Note that only classifiers that get into [MM] pay out from their strengths and receive payoff. That is, they are *evaluated*, whereas the remaining members of [M] are not. Consequently, if the size of [MM] is small, the overall rate of evaluation of the classifier population is reduced. Each time a classifier is evaluated, its strength tends toward a more accurate estimate of the classifier's typical payoff (divided by c). High evaluation rates thus mean that the genetic algorithm has more accurate fitness information to work with. Conversely, low rates can mean inaccurate information, so that, for example, poor classifiers over-reproduce and good ones don't reproduce enough, slowing down the discovery process.

Of course, there may be good reasons for conducting the bid competition and forming a restricted message list that outweigh any attendant slowing of learning. For example, a bid competition would allow the system to focus on the most significant aspects of a situation, as indicated by the high-bidding classifiers [13]. However, the most important suggested benefit of the bid competition is the formation of *default hierarchies* of classifiers [14]. In the simplest default hierarchy, there is a general, *default* rule that is correct in most situations that satisfy its condition, but in a few situations it is not correct. To cover these situations, the system has one or more *exception* rules and some automatic mechanism to make sure that the exception rules then control the system and not the default. If the default rule is indeed broadly correct and the exceptions quite rare, considerable representational economy may be achieved compared with a set of completely accurate rules covering the same set of situations—that is, compared with rules that logically partition the situations [13].

In a classifier system, the default rule would be a classifier with many #'s in its condition; a related exception classifier would match in a subset of the situations matched by the default. For these rules to control the system accurately, it should be the case that the default controls when it is correct, and the exceptions control when they, and not the default, are correct. The dependence of the bid competition on specificity is designed to accomplish this. For the broad set of cases when only the default matches, and in which it is by hypothesis correct, only the default will go into [MM], and its action will control. In an exception situation both the default and the appropriate exception rule will match. But due to its higher specificity (assuming equal strengths) the exception will have a correspondingly higher chance of getting into [MM], in preference to the low-specificity default. Thus the exception classifier will tend to control, and the default, which is wrong in the exception situation, will not cause the system to make a mistake. At the same time, the default is prevented from paying the cost of error, namely, the strength reduction of step 4 had it entered [MM], plus any penalty that might have come in step 7. In this sense, the higher-bidding exception may be said to "protect" the default from loss.

4.2.3 Are there default hierarchies?

Since the bid competition was in effect in Experiment 2, we should expect evidence of default hierarchies in the results, especially for small active set size. Examination of the classifier populations showed, however, little sign that default hierarchies had evolved. Instead, the system quite clearly tended to evolve the partition represented by the solution set [S6], nearly independent of [MM] size.

Classifiers in a population can be sorted in the form of a *macrostate* list [7] in which the classifier with the most copies heads the list, the one with the next most copies is next, etc. Or, the listing may be arranged in order of the total strength possessed by each set of copies. The two methods produce very similar orderings. Macrostates of the populations at 40,000 problems were formed for all 20 runs of Experiment 2. Upon examination, the main impression was that the first 8 to 10 classifiers in each of these lists had most of the numerosity and strength, and nearly always included the members of [S6]. An example of a very clear partition, from a run in which [MM] size was 5, was

0	0	1	#	#	#	/	1	56
0	1	#	1	#	#	/	1	51
1	0	#	#	1	#	/	1	51
0	0	0	#	#	#	/	0	48
1	0	#	#	0	#	/	0	48
1	1	#	#	#	0	/	0	48
1	1	#	#	#	1	/	1	35
0	1	#	0	#	#	/	0	20
#	1	#	0	#	#	/	0	3
0	1	#	#	#	#	/	0	3

for the top ten classifiers (number of copies in the right-hand column). About the clearest example of a default hierarchy was the following, where [MM] size was 10:

0	1	#	0	#	#	/	0	54
0	0	0	#	#	#	/	0	46
1	0	#	#	0	#	/	0	43
1	0	#	#	1	#	/	1	43
1	1	#	#	#	1	/	1	43
#	1	#	1	#	#	/	1	42
1	1	#	#	#	0	/	0	38
#	0	1	#	#	#	/	1	24
0	0	1	#	#	#	/	1	16
#	0	1	#	1	#	/	1	6

Note that the sixth classifier will be wrong for any input of the form $11*1*0$ ("$*$" means either input value). However, the seventh classifier matches in

[MM] size	σ	σ^2
5	6.6.	6.6
10	7.4	7.2
20	7.8	7.8
unlimited	7.6	8.0

Table 1: Mean number of solution classifiers in the top 8 versus active set size at two specificity powers.

all such situations and is correct; since it is more specific it stands to outbid, and thus protect, the other. A similar relation holds between the eighth and third classifiers, respectively. In all 20 runs, these were about the most prominent examples of default hierarchies. To get a numerical picture, the mean number of solution classifiers in the top eight macrostate classifiers was calculated for each [MM] size. In addition, the experiment was repeated but with σ^2, instead of just σ, factored into the bid (so as presumably to increase a more specific classifier's protective ability [10]). The results were as seen in Table 1. A value of 8.0 corresponds to a partition solution, so that even with [MM] size of 5, the solution classifiers dominate. Increasing the power of specificity does not seem to aid default hierarchies. However, it is not yet clear just what a default hierarchy for the multiplexer problem should look like. The next experiment produced one that was quite striking. For the moment, let us draw the working conclusion that, on balance, the bid competition does not promote significant default hierarchies—and propose an explanation why.

Our hypothesis is that, while the bid competition protects defaults when they are wrong, it tends to *starve them* when they are right. Earlier, we assumed that, in non-exception cases, the default would get the payoff because the exceptions would not match and therefore would not compete with the default. However, we did not also ask whether some other classifier might not sufficiently compete with the default to prevent it from entering [MM]. In particular, consider a classifier having the same action as the default, but a more specific version of its taxon (different from any exception's taxon). Such a classifier would tend to beat the default out of [MM] and thus get its payoff. Getting more payoff, it would tend to proliferate in the population. The default, on the other hand, because it participates in relatively few payoffs, would have trouble maintaining its numbers in the face of deletion and could well disappear (see the appendix of [7] for similar deletion effects).

This reasoning suggests that the bid competition will support exception classifiers and more-specific versions of the default, but not the default. The implication is that the system will tend to evolve partition solutions — as indeed occurred in Experiment 2 — not default hierarchies. The results in Table 1 indicate little effect of different [MM] sizes. This is reasonable if the bid competition both protects and starves defaults. Then the additional protection from a smaller [MM] would be accompanied by additional starvation, with no net advantage to the default. To the extent it applies to the standard classifier system, Experiment 2 suggests default hierarchies will be rare.

4.2.4 What does it take to get them?

It would be interesting to find a variant of the **Boole** algorithm that makes default hierarchies. The key seems to be to achieve protection without attendant starvation. Let us compare the following algorithm with **Boole**-2.

Algorithm Boole-3

1. A match set [M] is computed consisting of all classifiers whose condition is satisfied by the input string.
2. Each classifier in [M] makes a bid B equal to the product of its strength S, its specificity σ, and a small constant c like 0.1.
3. —
4. —
5. —
6. The system makes its decision (1 or 0) by selecting a classifier from [M] with probability proportional to classifier bids, and outputting the selected classifier's action as the decision.
7. If, following the decision, external payoff enters the system, it is shared by the members of [M] that advocated that action.
8. Each classifier in the "advocate set" [AD] of step 7 has its strength reduced by a payout equal to c times the strength.

Steps 1 and 2 are identical to **Boole**-2; elimination of steps 3, 4, and 5 removes [MM]; steps 6 and 7 are just like **Boole**-2 except they now refer to [M]. Finally, step 8 reintroduces the payout step eliminated with step 4, but in a different way: from among the members of [M], it is the advocates of the system's decision that pay out, not the high bidders (i.e., [MM]).

Under **Boole**-3, a default will be protected by an exception because, in step 6, the exception will control the decision, and therefore the default will not belong to [AD] and so will not pay out in step 8. Now consider the case when the default is correct. It and any more-specific versions of itself will control the decision in step 6, receive payoff in step 7, and do their payout in step 8. No starvation can occur because all classifiers in [M] that agree with the system's decision receive payoff. This was not the case under **Boole**-2.

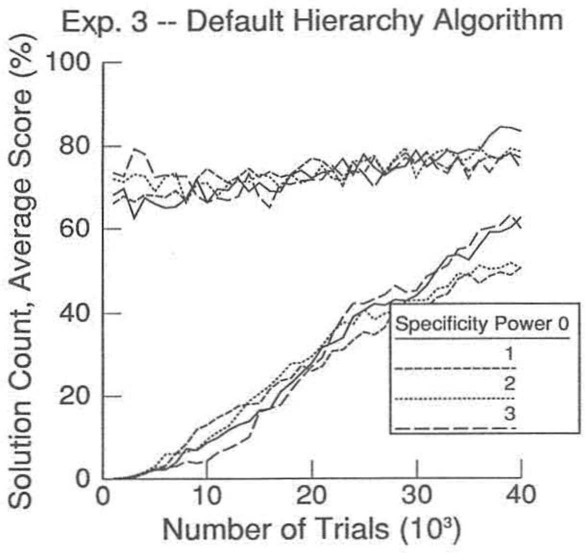

Figure 3: Results of experiments under algorithm **Boole-3**, which produced default hierarchies, for different specificity powers. Curve sets represent same quantities as in previous figures.

4.3 Experiment 3

4.3.1 Procedure and results

This experiment used **Boole-3**. The objective was to see whether default hierarchies would form. To investigate the effect of various degrees of emphasis of specificity in step 2, separate runs were conducted with: specificity not factored into the bid, and specificity raised to the first, second, and third powers, respectively, before being factored in. Each of these runs was repeated five times using different initial populations and the results were averaged and plotted in Figure 3. Parameter values were the same as in previous experiments.

The experiment produced extremely modest performance, with average score quickly rising to, then staying in, the 70% range. Solution counts, however, rose rather strongly to about 50% by 40,000 problems. Interestingly, both average score and solution count showed little dependence on specificity power.

Examination of the macrostates at 40,000 problems for each of the 20 runs showed a strong pattern of default hierarchy in runs where specificity entered the bid calculation. For example, the top eight classifiers for a run with specificity to the first power were:

0	0	1	#	#	#	/	1	56
1	1	#	#	#	1	/	1	56
0	1	#	1	#	#	/	1	48
1	0	#	#	1	#	/	1	39
#	#	#	#	#	#	/	0	23
1	#	#	#	1	#	/	1	23
#	#	#	#	0	#	/	0	18
0	#	#	0	#	#	/	0	17

The first four classifiers are those of [S6] having action 1. They and the fifth classifier form a default hierarchy that says, in effect, "decide 0, except decide 1 in cases matched by one of the first four." Note that the general classifier, # # # # # # / 0, is wrong in 32 out of 64 cases, yet has a strong position in the macrostate — showing that it is both well protected and "nourished."

This pattern of five classifiers, or its complement with opposite action bits, occurred in the top eight macrostate positions in 11 out of the 15 runs that had specificity in the bid. In those 11 runs, no other member of [S6] ever appeared in the top eight positions, indicating a default hierarchy was decisively chosen over a partition solution. In the other four of those 15 runs, the picture was one of incomplete formation of one or the other hierarchy. In general, there was no correlation between specificity power and the likelihood of getting a hierarchy. In the five runs that did not have specificity in the bid, the partition solution was evident, as would be expected; the average number of solution classifiers in the top eight was 6.4.

4.3.2 Discussion

Experiment 3 confirms Holland's insight that default hierarchies can occur in classifier systems, but a reinforcement regime different from the standard appears to be required. Instead of the standard's active set bid competition, under which payoff can never go to the low bidders, the experiment suggests that the correct principle is for payoff to go to all matching classifiers that agree with the system's decision. Unfortunately, it is not obvious how to apply this principle in the standard classifier system. The reason is that on each internal cycle of the standard, messages are simply posted: there is no clear notion of agreement and disagreement with a system decision. Even on a cycle containing an external action it would be hard to apply the principle since many classifiers in the match set are not, in general, involved with the action decision.

Because it seems important to be able to evolve default hierarchies in the standard system, we shall offer a suggestion. Since it involves a rather deep change, the suggestion is offered mainly as food for thought. The **Boole-3** algorithm generates default hierarchies because there is a clear notion of system decision with which matching classifiers either agree or disagree. Consider a standard system in which on each cycle a *single message is computed collectively using all the matching classifiers.* Suppose, for concreteness, that messages are of length 6 and each classifier has just one condition. An example classifier might look like this:

```
0 0 1 # # 0 / 43:1 4:0 15:0 27:1 2:0 9:0.
```

It says: "if my taxon matches the current (single) message (for simplicity, ignore the environmental message), then in computing the next message I vote 43 for 1 on the first bit, 4 for 0 on the second, etc." Each matching classifier would make a similar statement. The system would decide the bits of the next message by adding up the votes for each bit and deciding 1 or 0 depending on which had the majority. Notice that for each bit decision there will in general be some classifiers that agree and some that disagree. Those that agree will, in analogy with **Boole-3**, share one-sixth of the available payoff. Those that disagree with a particular bit's decision will get nothing; but they may be in the agreement sets for other bits. The payoff shares to a given classifier would be added to the corresponding "vote" weights, and a payout consisting of a small fraction of each weight would be removed from those vote weights that had been adjusted. The fitness of a classifier for the genetic algorithm would equal the sum of its weights, which form a sort of strength vector in contrast to the scalar strength of the standard system.

This approach seems likely to yield default hierarchies. The problem is that the "message list" has been reduced to a single message, apparently drastically reducing the system's concurrency. Two comments can be made. First, it is at least worth noting that the message could be regarded as bearing the concurrency in its substrings. That is, substrings could be thought of as analogous to the separate messages of the standard system. Second, a multistep classifier system has been proposed [8], in which, though the message list can have many messages, only one message is posted on each cycle; it could be computed as above. That system has not been investigated experimentally.

5. Summary

The results in this paper point to several conclusions with respect to classifier system mechanisms:

1. Factoring specificity into a classifier's payout introduces a significant tendency toward overgeneralization and poor performance.

2. Restriction of system control to a high-bidding ("active") subset of the match set is not, in general, desirable:

   (a) Performance and learning are better using the whole match set, without restriction (except perhaps in the presence of strong overgeneralization forces);

   (b) The bid competition does not encourage default hierarchies: defaults are indeed protected when they are wrong, but they are "starved" when they are right.

3. One method of promoting default hierarchies is to distribute payoff to, and take payout from, all matching classifiers that agree with the system's decision, leaving alone the others. However, "system decision", as well as agreement and disagreement therewith, must be well defined.

A caveat on these conclusions is that they were drawn from experiments with **Boole**, a specialized, one-step classifier system that allowed easier interpretation of the results than would have been the case with the standard system. Analogous experiments should be performed with the standard system using well understood tasks.

6. Related work

In one part of a wide-ranging article on genetic algorithms and classifier systems, Holland [13] proposes changes in the use of specificity that parallel those investigated here, but appear to be more complicated. In our terminology, Holland forms a classifier's (bid competition) bid "by reducing [the quantity σS] in proportion to the generality of the classifier." The simplest interpretation of this would make the actual ("effective") bid equal to $\sigma^2 S$. Next, a classifier's payout is made equal to σS. Finally, a classifier's probability of reproduction under the genetic algorithm is also made σS. Notice that if σ is divided out of all three quantities, one obtains the values used in **Boole**-2 (or -3). As a check, an experiment was performed using Holland's regime; the results were similar to Experiment 2, including the absence of default hierarchies.

In careful simulations, Riolo [10] shows that if the bid competition is sufficiently biased against generalists, default hierarchies, once formed, will persist stably. However, the simulations included only the performance and reinforcement components and not the genetic algorithm. Since Riolo advocates factoring specificity into a classifier's payout, we would predict that a complete experiment would show a tendency toward overgeneralization, but offset by starvation of the overgenerals due to the bias against them. Whether this opposition would lead to interesting default hierarchies is not clear.

Acknowledgments

The author acknowledges stimulating debates on these questions with John Holland. I appreciate the able assistance of Lesley Pew in the preparation of this and previous papers.

References

[1] J.H. Holland, "Adaptation", in *Progress in Theoretical Biology*, **4**, eds. R. Rosen and F. M. Snell (Plenum, New York, 1976).

[2] J.H. Holland, "Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems", in *Machine Learning, an Artificial Intelligence Approach II*, eds. R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Los Altos, California: Morgan Kaufman, 1986).

[3] J.H. Holland and J.S. Reitman, "Cognitive systems based on adaptive algorithms" in *Pattern-Directed Inference Systems*, eds. D.A. Waterman and F. Hayes-Roth (New York: Academic Press, 1978).

[4] L. Booker, *Intelligent Behavior as an Adaptation to the Task Environment*, Ph. D. Dissertation, Computer and Communication Sciences, University of Michigan, 1982.

[5] D.E. Goldberg, *Computer-aided pipeline operation using genetic algorithms*, Ph. D. Dissertation, University of Michigan, 1983.

[6] S.W. Wilson, "Knowledge growth in an artificial animal", *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Hillsdale, New Jersey (1986) 16–23.

[7] S.W. Wilson, "Classifier systems and the animat problem", *Machine Learning*, **2** (1987) 199–228.

[8] S.W. Wilson, "Hierarchical credit allocation in a classifier system", *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, (Los Altos, California: Morgan Kaufman, 1987) 217–220.

[9] R.L. Riolo, "Bucket brigade performance: I. Long sequences of classifiers", *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1987) 184–195.

[10] R.L. Riolo, "Bucket brigade performance: II. Default hierarchies", *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1987) 196–201.

[11] G.G. Robertson and R.L. Riolo, "A tale of two classifier systems", *Machine Learning*, **3** (1988) 139–159.

[12] J.H. Holland, K.J. Holyoak, R.E. Nisbett, and P.R. Thagard, *Induction: Processes of Inference, Learning, and Discovery* (Cambridge, MA: MIT Press, 1986).

[13] J.H. Holland, "Genetic algorithms and classifier systems: foundations and future directions", *Genetic Algorithms and Their Applications: Proceedings*

of the Second International Conference on Genetic Algorithms (Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1987) 82–89.

[14] J.H. Holland, "Properties of the bucket brigade algorithm", *Proceedings of the First International Conference on Genetic Algorithms and Their Applications* (Hillsdale, NJ: Lawrence Erlbaum Associates, 1985) 1–7.

[15] A.G. Barto, "Learning by statistical cooperation of self-interested neuron-like computing elements", COINS Technical Report, 85-11 (1985) (available from Dept. of Computer and Information Science, University of Massachusetts, Amherst, MA 01003).