Using Convex Hulls to Represent Classifier Conditions

Pier Luca Lanzi
Dipartimento di Elettronica e Informazione
Politecnico di Milano, I-20133, Milano, Italy
Illinois Genetic Algorithm Laboratory (IlliGAL)
University of Illinois at Urbana Champaign,
Urbana, IL 61801, USA
pierluca.lanzi@polimi.it

Stewart W. Wilson
Prediction Dynamics, Concord, MA 01742, USA
Illinois Genetic Algorithm Laboratory (IlliGAL)
University of Illinois at Urbana Champaign,
Urbana, IL 61801, USA
wilson@prediction-dynamics.com

ABSTRACT

This papers presents a novel representation of classifier conditions based on convex hulls. A classifier condition is represented by a sets of points in the problem space. These points identify a convex hull that delineates a convex region in the problem space. The condition matches all the problem instances inside such region. XCSF with convex conditions is applied to function approximation problems and its performance is compared to that of XCSF with interval conditions. The comparison shows that XCSF with convex hulls converges faster than XCSF with interval conditions. However, convex conditions usually do not produce more compact solutions.

Categories and Subject Descriptors

F.1.1 [**Models of Computation**]: Genetics Based Machine Learning, Learning Classifier Systems

General Terms

Algorithms, Performance.

Keywords

LCS, XCS, Representation, Convex Hulls.

1. INTRODUCTION

Effective generalization depends both on how the system partitions the problem space and on how the system models the problem solution over the resulting partition. Learning classifier systems usually partition a binary problem space using ternary conditions and map each problem subspace into a constant prediction value. More recent models such as XCSF [17] partition the problem space using typical representations (ternary [9], interval [16], and ellipsoids [1]) and model the problem solution by means of function approximators (e.g., linear [17] and polynomial [7]).

In this paper we introduce another representation of classifier conditions, based on convex hulls, that is particularly suited for real inputs (Section 3). A classifier condition is represented as sets of points in the problem space. These points correspond to a convex hull that identifies a convex region in the problem space. The condition matches all the problem instances inside such convex region. We apply XCSF with convex conditions to function approximation problems taken from the classifier system literature [1]. First, in Section 5, we compare the performance of XCSF with convex conditions with a fixed number of points (3, 5, 10, and 15) to that of the typical interval representation. We show that, when few points are used, XCSF with convex conditions tends to converge faster than XCSF with interval conditions and produces solutions that are on the average more accurate. In particular, the simplest convex conditions possible, consisting of at most three points, converges faster than all the other versions of XCSF. However, convex representation generally evolves populations that are larger than those evolved with interval conditions. We extend this comparison to convex conditions involving a variable number of points (Section 6). The experiments we present show that, in term of prediction error, convex conditions with a variable number of points always provide a tradeoff between conditions with few point (3 and 5) and conditions with many points (10 and 15). However they usually converge slower than interval conditions and, because of the typical bloating of variable sized representations, much larger populations. Then, in Section 7, we consider the problem of constraining the convex hull to the domain which potentially can introduce a generalization bias. We compare constrained and unconstrained convex conditions: on the problems considered in this paper, no significant difference is found. Finally, we show how convex conditions can be coupled with more effective prediction [7] and better prediction update [8] to further improve the performance of convex conditions.

2. XCS WITH COMPUTED PREDICTION

Computed prediction [17] replaces the usual classifier prediction with a parameter vector $\mathbf{w}$ and a prediction function $p(s_t, \mathbf{w})$, which defines how classifier prediction is computed from the current input s_t and parameter vector $\mathbf{w}$. In addition, the usual update of the classifier prediction is replaced by the update of the classifier parameter vector $\mathbf{w}$. The prediction function $p(s_t, \mathbf{w})$ is usually defined as $s_t\mathbf{w}$ but more complex functions can be used [7], as well as more simple ones. In fact, by using just one parameter (i.e., $\mathbf{w} = \langle p \rangle$) and the prediction function $p(s_t, \mathbf{w}) = p$, XCSF actually implements XCS.

At time step t, XCSF builds a *match set* [M] containing the classifiers in the population [P] whose condition matches the current sensory input s_t; if [M] contains less than θ_{mna} actions, *covering* takes place. For each action a in [M], XCSF computes the *system prediction* $P(s_t, a)$ as the fitness-weighted average of all matching classifiers that specify action a. The values of $P(s_t, a)$ form the *prediction array*. Next, XCSF selects an action to perform. The classifiers in [M] that advocate the selected action are put in the current *action set* [A]; the selected action is sent to the environment and a reward r is returned to the system together with the next input state s_{t+1} XCSF uses the incoming reward to update the parameters of classifiers in action set $[A]_{-1}$ corresponding to the previous time step. At time step t, the expected payoff P is computed as $r_{-1} + \gamma \max_{a \in A} P(s_t, a)$, where r_{-1} is the reward received at the previous time step. The expected payoff P is used to update the weight vector $\mathbf{w}$ of the classifier in $[A]_{-1}$ using a *modified delta rule* with learning rate η (see [17] for details). Then the prediction error ϵ and the fitness are updated as usual [2]. The genetic algorithm is applied as in any other XCS model [17]. The weight vectors of offspring classifiers are set to a fitness weighted average of the parents weight vectors; all the other parameters are initialized as usual [2].

3. CONDITIONS BASED ON CONVEX HULLS

Given a set of points P in an n-dimensional space, the convex hull of P, $H(P)$, is defined as the smallest convex region enclosing the points in P. In two dimensions, the convex hull is found conceptually by stretching a rubber band around the points so that all of the points lie within the band. For instance, the eight points ($P = \{p_1, \ldots, p_8\}$) in Figure 1a are enclosed in the convex hull depicted with a dashed line in Figure 1b and identified by the five solid points ($H(P) = \{p_1, p_2, p_8, p_7, p_4\}$).

The idea behind the use of convex hulls to represent classifier conditions is extremely simple. We represent classifier conditions as sets of points in the problem space; a condition matches all the states enclosed in the convex hull that the condition identifies. For instance, suppose that a classifier condition is represented by the eight points in Figure 1a; such condition will match all the points in the corresponding convex hull, depicted by the dashed lines in Figure 1b; thus, given the states s_1 and s_2 in Figure 1, the condition will match s_1 but not s_2.

Convex hull representation is a generalization of the interval representation introduced by Wilson [16]. In fact, interval conditions are a special case of convex hulls, but convex hulls allow the partitioning of the problem space into more complex regions. Convex hulls can also represent structures similar to those provided by the more recent ellipsoidal conditions [1]. In fact, ellipsoids define convex regions that may be approximated through polygons with an adequate number of points. In a different way, the simplest form of convex hulls, the triangle in two dimensions, can be used to partition the problem space producing results similar to those produced by Delaunay triangulation [3]. In the following we describe the modifications to XCSF that the introduction of convex hull representation requires.

Conditions. When representing classifier conditions with a set of points, we can either use a fixed number of points (n_p) for all the conditions in the population or a variable number of points for each condition. In the former case, we set an upper limit to the complexity of the convex region that the condition can represent but the genetic search is facilitated. In the latter case, the more general one, we allow conditions that represent arbitrarily complex convex regions but we increase the complexity of the genetic search and we also introduce the bloating phenomena that are typical with variable size representations [13, 14].

Matching. To match a condition against the current input state s we consider the convex hull that the condition represents, then we check whether s lies inside the convex hull. The first step is performed only once when the classifier is generated and given n_p points it has a complexity of order $O(n_p \log n_p)$. In the experiments considered here, limited to two dimensional problem spaces, we compute convex hulls by using the Graham's scan algorithm [6] available in the Computational Geometry Algorithms Library (CGAL) [4]. The second step can be generally performed using algorithms for convex hull computation though in two dimensions a simpler solution exists. Given a convex hull of n points $p_1, \ldots, p_n$ taken in clockwise (or counterclockwise) order to test whether s lies inside the convex region we consider all the vectors connecting p_i to p_{i+1} and the vector connecting p_n to p_1; then we test whether s lies to the right of all the vectors (to the left if the points are in counterclockwise order); if this is the case, then s lies inside the convex hull determined by $p_1, \ldots, p_n$; otherwise s lies outside the convex hull. Note that since the algorithms that compute the convex hull in two dimensions return a set of points in clockwise order the whole match operation is just $O(n_p)$.

Covering. To generate a covering condition of n_p points for an input state s, n_p random points are generated in the surroundings of s. For this purpose, we use polar coordinates with origin in s and we generate n_p angles, between 0 and 2π, and n_p radial distances between 0 and r_0. As in the interval representation, the parameter r_0 controls the size of the covering region.

Discovery Component. The genetic algorithm works as usual. With probability χ crossover is applied and with probability μ mutation is performed on each allele. When conditions consist of a variable number of points, the typical crossover and mutation for variable size representation are applied (e.g., [13]).

Subsumption Deletion. To implement subsumption for conditions based on convex hulls we need to test whether one condition (C_1) is more general than another condition (C_2). This test can be easily implemented as follows: first the two

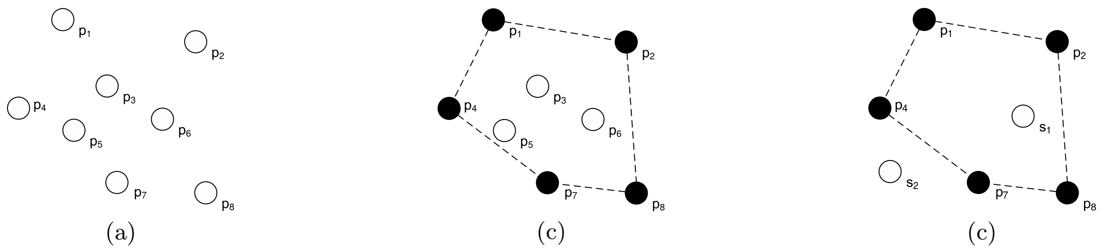


Figure 1: (a) Eight points in the Cartesian space; (b) the corresponding convex hull depicted as a dashed line and identified by the points p_1, p_2, p_8, p_7, and p_4; (c) the convex hull enclosing the points in (a).

corresponding convex hulls H_1 and H_2 are considered; then the convex hull H for $H_1 \bigcup H_2$ is computed; if H is equal to H_1, it means that C_1 is more general than C_2 since H_1 contains H_2; if H is equal to H_2, it means that C_2 is more general than C_1 since H_1 contains H_2; otherwise none of the two conditions is more general than the other one.

4. DESIGN OF EXPERIMENTS

To apply XCSF to function approximation, we follow the standard experimental design used in the literature [15]. Each experiment consists of a number of problems that the system must solve. Each problem is either a *learning* problem or a *test* problem. In *learning* problems, the system selects actions randomly from those represented in the match set. In *test* problems, the system always selects the action with highest prediction. The genetic algorithm is enabled only during *learning* problems and it is turned off during *test* problems. The covering operator is always enabled, but operates only if needed. Learning problems and test problems alternate.

In function approximation problems, an example $\langle (x,y), f(x,y) \rangle$ of the target function $f(x,y)$ is randomly selected; x, y is input to XCSF which computes the approximated value $\hat{f}(x,y)$ as the expected payoff of the only available dummy action; the action is virtually performed (the action has no actual effect), and XCSF receives a reward equal to $f(x,y)$. XCSF learns to approximate the target function $f(x,y)$ by evolving a mapping from the inputs to the payoff of the only available action. All the statistics reported in this paper are averaged over 20 experiments.

5. THREE DIMENSIONAL FUNCTIONS

In the first set of experiments we apply convex conditions on the three following functions [1]:

$$f_1(x,y) = \mathrm{mod}(\lfloor 3x \rfloor, 3)/3 + \mathrm{mod}(\lfloor 3y \rfloor, 3)/3 \quad (1)$$

$$f_2(x,y) = \mathrm{mod}(\lfloor 2(x+y) \rfloor, 4)/6 \quad (2)$$

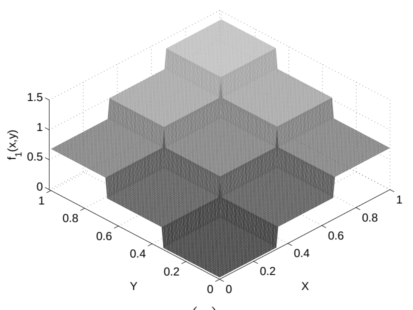
$$f_3(x,y) = \sin(2\pi(x+y)) \quad (3)$$

where $x \in [0,1]$, $y \in [0,1]$, $\lfloor \cdot \rfloor$ is the floor function, and "mod" is the common residue. Function f_1 is an axis-parallel step function (Figure 2a), f_2 is an axis-diagonal step function (Figure 2b), and f_3 is an axis-diagonal sinusoid (Figure 2c).

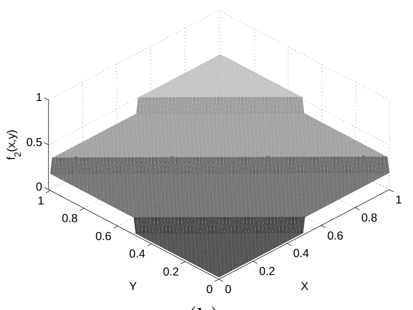
In the first experiment we apply convex hulls with conditions of 3, 5, 10, and 15 points to approximate f_1; the parameters are set as follows: $N = 6400$; $\eta = 0.5$; $\beta = 0.5$; $\alpha = 0.1$; $\nu = 5$; $\chi = 1.0$, $\mu = 0.05$, $\epsilon_0 = 0.01$; $\theta_{del} = 20$; $\theta_{GA} = 20$; $\delta = 0.1$; GA-subsumption is on with $\theta_{sub} = 50$; while action-set subsumption is off; $r_0 = 1.0$, and $x_0 = 1$. The mutation of convex conditions in this case is implemented as usual, i.e., with probability μ each allele is mutated using $m_0 = 0.2$ [17]. Figure 3a compares the performance of interval conditions with that of convex hulls using different number of points. Figure 3a shows that convex conditions converge faster only when they are limited to three points, i.e., only when conditions can at most represent triangles. As the number of points increases the convergence becomes slower and slower: when conditions consists of 10 or 15 points, XCSF cannot converge below the required error threshold ϵ_0. We performed a statistical analysis on the curves in Figure 3a to test whether the differences in the performances are statistically significant. For this purpose we followed the approach introduced in [12] which is based on a one-way analysis of variance or ANOVA. The analysis shows that convex conditions with three points are significantly faster than all the others versions at a 99.99% confidence level. The following post-hoc procedures we applied (SNK, Scheffé, Bonferroni, and Tukey [5]) showed that the performance of interval conditions and convex conditions with 5 points are similar, i.e., they are not statistically different at a 99.99% confidence level.

In terms of size of the solutions evolved, interval conditions perform better than convex conditions. Figure 3b shows that the solutions evolved by XCSF with interval conditions are much smaller than those evolved by XCSF with convex hulls. Figure 3c reports the average generalization in the current action set measured as the average perimeter of the convex hulls that conditions in the action set represent. As can be noted for three of the four settings (with 3, 5, and 10 points) the average generality tends to converge to a value just above 1, i.e., near to the perimeter of the nine areas that allow an accurate maximally general representation of f_1. Overall the results for the function f_1 suggest that convex conditions may converge faster to an accurate approximation, given an adequately small number of condition points; nevertheless, when the problem space is favorable to the interval representation, convex conditions may perform worse in terms of generalization.

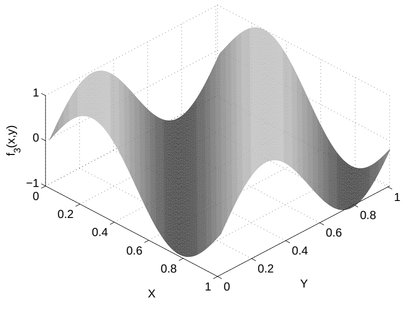
In the next experiment, we apply convex conditions to f_2: being f_2 an axis diagonal step function we should expect an improvement in the performance of convex conditions which can easily represent oblique regions. Figure 4a compares the prediction error of interval conditions and convex conditions on f_2. Both convex conditions with 3, 5, and 10 points show a faster convergence than interval conditions; convex conditions with 10 points are still slower than conditions of 3 and 5 but in this case they can reach a prediction error below the target threshold ϵ_0 which is not reached by convex condi-



(a)



(b)



(c)

Figure 2: (a) f_1, an axis parallel step function; (b) f_2, an axis diagonal step function; (c) f_3, an axis diagonal sinusoidal function.

tions with 15 points. Also for f_2, we performed a statistical analysis on the curves in Figure 4a following the approach in [12]. The analysis shows that all the versions of XCSF reported in Figure 4a perform significantly different: again convex hulls significantly improve the learning capabilities of XCSF with respect to the interval conditions. In function f_2 convex conditions improve in terms of the size of the populations evolved. Figure 4b shows that convex hulls with 3, 5, and 10 points reach a population size that is just slightly higher than that evolved by interval conditions. Conditions with 15 points initially tend to have populations consisting mainly of macroclassifiers then, they converge toward much smaller populations as evolution proceeds. Again, the average generality of the classifiers in the current action set (Figure 4c) tends to the same value for all the four settings.

Finally, we compare interval conditions and convex conditions on the diagonal sinusoidal function f_3. The results reported in Figure 5 confirm what found for the function f_2. Convex conditions with 3 and 5 points converge faster than interval conditions to a prediction error below ϵ_0; conditions with 10 and 15 points converge slower but, as in the previous experiment, at the end the prediction error is smaller than that obtained with interval conditions. The usual statistical analysis [12] applied to the curves in Figure 5 confirms the previous analyzes: all the XCSF versions considered perform significantly different, i.e., convex hulls significantly improve the learning capabilities of XCSF. In terms of generalization, on this particular problem, convex conditions can evolve solutions than are slightly more compact than those evolved by interval conditions (Figure 5b) both for 3, 5, and 10 points. As in the previous experiments, with 10 and 15 points the populations initially contain microclassifiers but they become much more compact at the end. Also in this case, the average generality of the classifiers in the current action set (Figure 5c) converges to the same value for all the four settings.

6. VARIABLE SIZE REPRESENTATION

We apply XCSF to convex conditions with a variable number of points. Conditions are represented directly as convex hulls, instead of as set of points; covering conditions are created by generating 10 random points around the current input, then computing the corresponding convex hull; conditions are recombined using the usual one point crossover for variable size representations [13] while mutation works as before.

Figure 6 compares the performance of variable size convex conditions with that of fixed size ones on function f_1. Variable size conditions converge faster than conditions with 10 and 15 points but are slower than the simpler conditions with 3 and 5 points. As anticipated in Section 3 variable size representation introduces a bloating effect: the number of macroclassifiers in the population tend to grow as much as possible as in the case of symbolic conditions [10]. However, in this case the number of macroclassifiers does not grow as much as for the case with 15 points. These results are confirmed both for f_2 and f_3 (see [**?**] for details): XCSF with variable size conditions converges faster than with 10 and 15 points but slower than with 3 and 5 points. There is a bloating effect, the number of classifier in the population is between 75% and 80% of N and, in contrast to the case of 10 and 15 points, in this case the population does not shrink. Also in this case, the average generality of classifiers in the current action set tend to the same value as in the previous experiments with fixed size conditions.

7. CONSTRAINED VS. UNCONSTRAINED

Interval conditions are usually constrained to the input domains by applying heuristics after crossover and mutation [16]. However, with convex conditions such heuristics may pose issues. Consider the condition depicted with a dashed line in Figure 7a, the problem space is represented by the square gray area. The simplest heuristic to constrain such condition consists of projecting the points outside the domain on the domain border – as done with interval conditions [16]. This implies changing the original condition to that in Figure 7b. However, the new condition is less general in that it matches less instances leaving out many border points. This phenomenon does not happen with interval conditions because their shape fit the typical hyperrectangular input space well. To evaluate the influence of such phenomenon on the performance of convex conditions, we compare constrained and unconstrained convex conditions on the previous problems.

We apply XCSF with unconstrained convex hull to the three functions f_1, f_2, and f_3. In these experiments we apply one point crossover instead of uniform crossover. Figure 8 reports the prediction error and the number of classifiers for constrained and unconstrained convex conditions for f_2. In contrast to what we may expect, we find no difference in performance between constrained and unconstrained convex hulls in terms of prediction error. Instead, in terms of number of macroclassifiers in the population, unconstrained convex conditions evolve larger solutions. Noticeably, all the versions of XCSF with convex conditions converge more or less to the same number of classifiers, 50% of N. The little difference in the performance disappears as the number of points increases. The same type of behavior is confirmed for f_1 and for f_3 (see [11] for details). Finally, we note that overall with one point crossover XCSF converges slightly slower than with uniform crossover (Figure 8) and this also happens in f_1 and f_3.

8. IMPROVING PERFORMANCE

In [8] it has been shown that generalization in XCSF can be improved by speeding up the convergence of classifier prediction and by using polynomials instead of linear prediction (see also [7]). We can combine the two proposed solutions to improve the performance of XCSF with convex conditions. Figure 9 reports the prediction error and the number of classifiers for XCSF with recursive least squares with quadratic approximators in f_3 with the same parameters setting used in the previous experiments. In this case, recursive least squares and quadratic approximation improve the convergence toward a slightly more accurate solution, but the solutions evolved are comparable to the previous experiments.

9. CONCLUSIONS

We have introduced a new representation for classifiers conditions that exploits points in the problem space to represent convex regions in the same space. The experimental results we report show that convex conditions can converge faster than interval conditions when few points are involved. The statistical analysis of the reported results show that such an improvement is significant at a 99.99% confidence level. As the number of points increases, i.e., the size of the conditions increases, convex representation converges slower. The experiments also show that convex conditions with a variable number of points can actually provide a tradeoff between the convergence speed of conditions with few and many points. However, convex conditions tend to evolve solutions that are on the average at least as complex as those produced by interval conditions. Further investigations include application to multistep and supervised classification tasks.

10. REFERENCES

[1] M. V. Butz. Kernel-based, ellipsoidal conditions in the real-valued XCS classifier system. In H.-G. Beyer, U.-M. O'Reilly, D. V. Arnold, W. Banzhaf, C. Blum, E. W. Bonabeau, E. Cantu-Paz, D. Dasgupta, K. Deb, J. A. Foster, E. D. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G. R. Raidl, T. Soule, A. M. Tyrrell, J.-P. Watson, and E. Zitzler, editors, *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, volume 2, pages 1835–1842, Washington DC, USA, 25-29 June 2005. ACM Press.

[2] M. V. Butz and S. W. Wilson. An algorithmic description of XCS. *Journal of Soft Computing*, 6(3–4):144–153, 2002.

[3] M. de Berg, O. Schwarzkopf, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2000. Second Edition.

[4] A. Fabri, E. Fogel, B. Gartner, M. Hoffmann, M. Karavelas, L. Kettner, S. P. M. Teillaud, R. Veltkamp, and M. Yvinec. Computational geometry algorithms library: User and reference manual. release 3.1, 2004. Available at `http://www.cgal.org`.

[5] S. A. Glantz and B. K. Slinker. *Primer of Applied Regression & Analysis of Variance*. McGraw Hill, 2001. second edition.

[6] R. L. Graham. An efficient algorithm for determing the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972.

[7] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. Extending XCSF beyond linear approximation. In *Genetic and Evolutionary Computation – GECCO-2005*, Washington DC, USA, 2005. ACM Press.

[8] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. Generalization in the xcsf classifier system: Analysis, improvement, and extension. Technical Report 2005012, Illinois Genetic Algorithms Laboratory – University of Illinois at Urbana-Champaign, 2005.

[9] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. XCS with Computed Prediction for the Learning of Boolean Functions. In *Proceedings of the IEEE Congress on Evolutionary Computation – CEC-2005*, Edinburgh, UK, 2005. IEEE.

[10] P. L. Lanzi and A. Perrucci. Extending the Representation of Classifier Conditions Part II: From Messy Coding to S-Expressions. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 99)*, pages 345–352, Orlando (FL), July 1999. Morgan Kaufmann.

[11] P. L. Lanzi and S. W. Wilson. Classifier conditions based on convex hulls. Technical Report 2005024, Illinois Genetic Algorithms Laboratory – University of Illinois at Urbana-Champaign, 2005.

[12] J. H. Piater, P. R. Cohen, X. Zhang, and M. Atighetchi. A Randomized ANOVA Procedure for Comparing Performance Curves. In *Machine Learning: Proceedings of the Fifteenth International Conference (ICML)*, pages 430–438, Madison, Wisconsin, July 1998. Morgan Kaufmann, San Mateo, CA, USA.

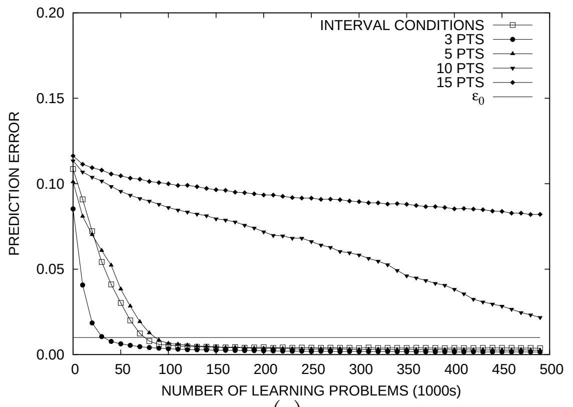
[13] T. Soule. Operator choice and the evolution of robust solutions. In R. L. Riolo and B. Worzel, editors, *Genetic Programming Theory and Practice*, chapter 16, pages 257–270. Kluwer, 2003.

[14] T. Soule and R. B. Heckendorn. An analysis of the causes of code growth in genetic programming. *Genetic Programming and Evolvable Machines*, 3(3):283–309, Sept. 2002.

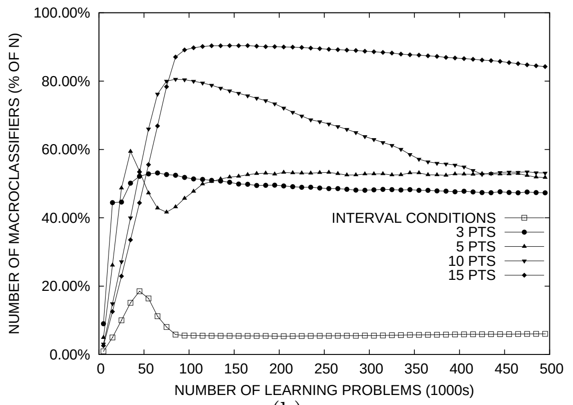
[15] S. W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[16] S. W. Wilson. Mining Oblique Data with XCS. volume 1996 of *Lecture notes in Computer Science*, pages 158–174. Springer-Verlag, Apr. 2001.

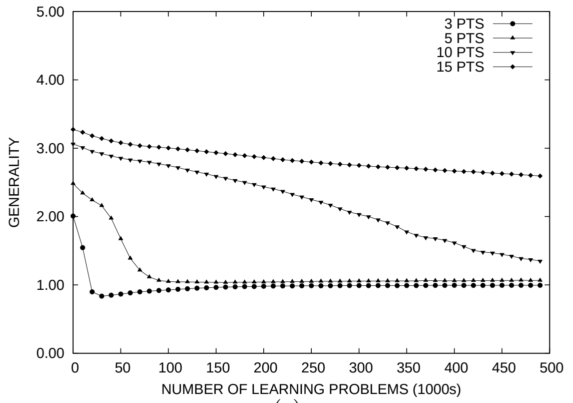
[17] S. W. Wilson. Classifiers that approximate functions. *Journal of Natural Computing*, 1(2-3):211–234, 2002.



(a)

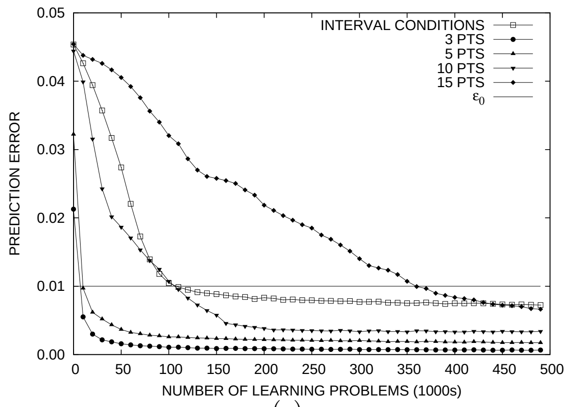


(b)

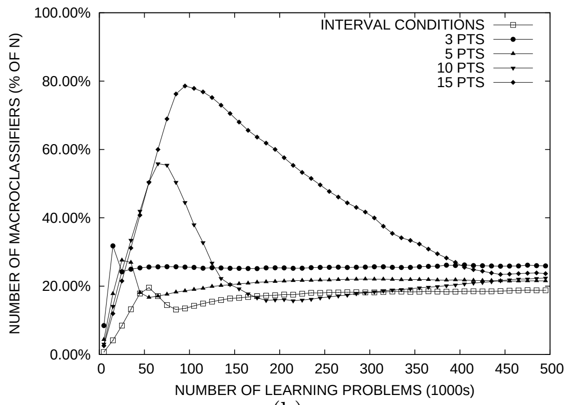


(c)

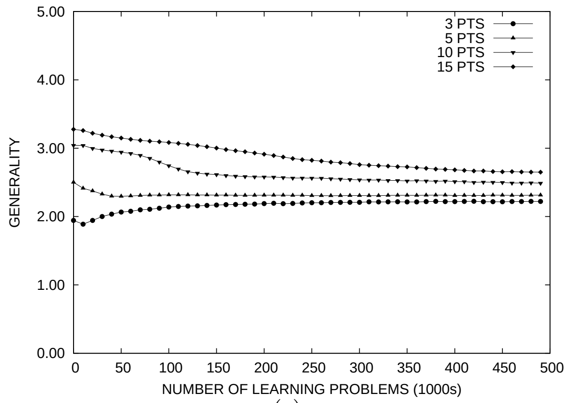
Figure 3: XCSF with convex conditions of different number number of points applied to f_1: (a) prediction error; (b) number of classifiers in the populations; (c) average generalization in the current action set. Curves are averages over 20 runs.



(a)

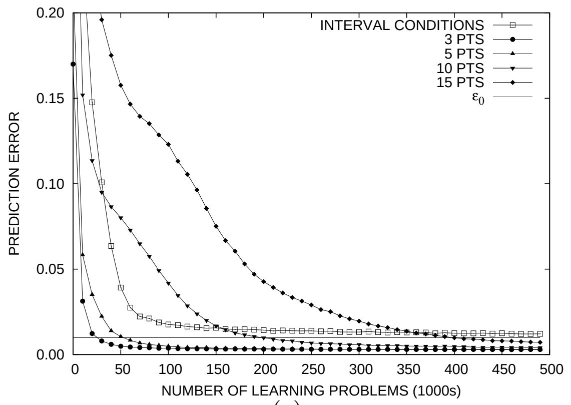


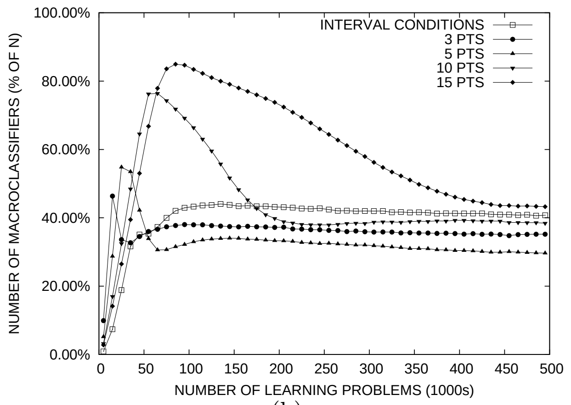
(b)



(c)

Figure 4: XCSF with convex conditions of different number number of points applied to f_2: (a) prediction error; (b) number of classifiers in the populations; (c) average generalization in the current action set. Curves are averages over 20 runs.


(a)


(b)

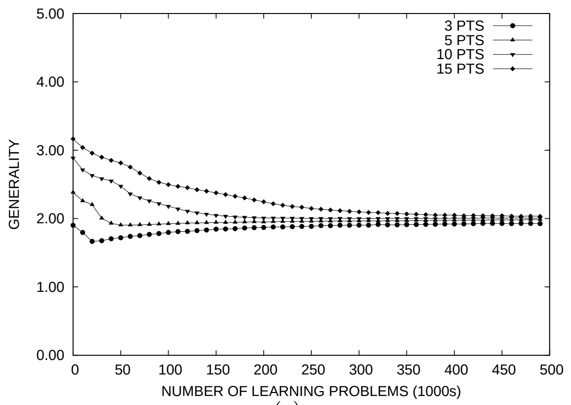
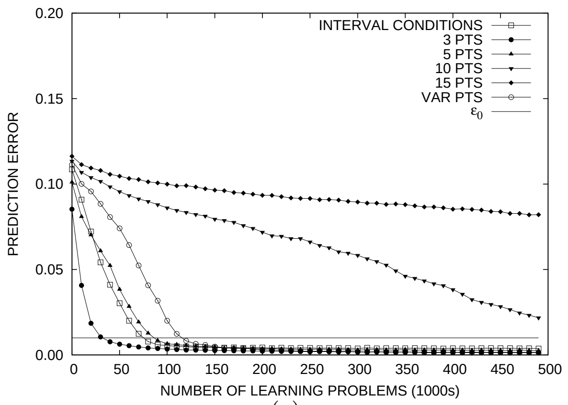

(c)

Figure 5: XCSF with convex conditions of different number number of points applied to f_3: (a) prediction error; (b) number of classifiers in the populations; (c) average generalization in the current action set. Curves are averages over 20 runs.


(a)

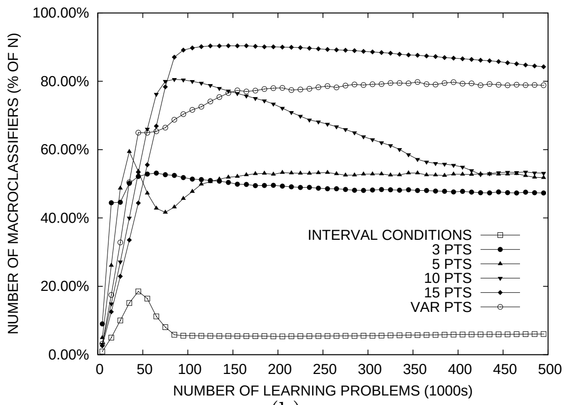
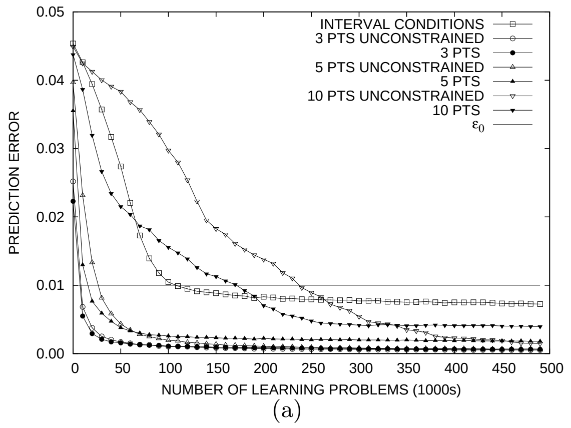
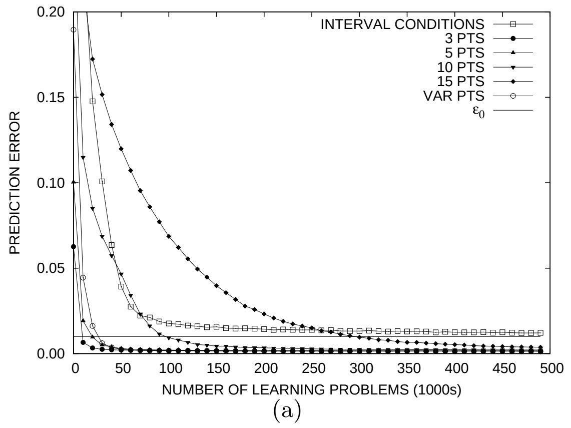

(b)

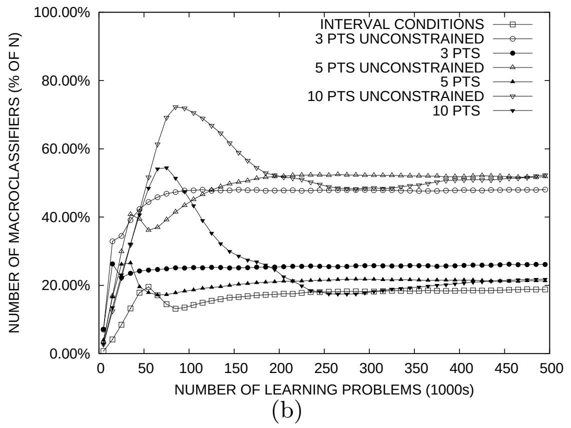
Figure 6: XCSF with variable size convex conditions applied to f_1 compared to fixed size convex conditions: (a) prediction error; (b) number of classifiers in the populations. Curves are averages over 20 runs.

(a) (b)

Figure 7: (a) Unbounded convex conditions allow a better covering of the problem space border, while (b) constrained conditions may make such a covering more difficult.







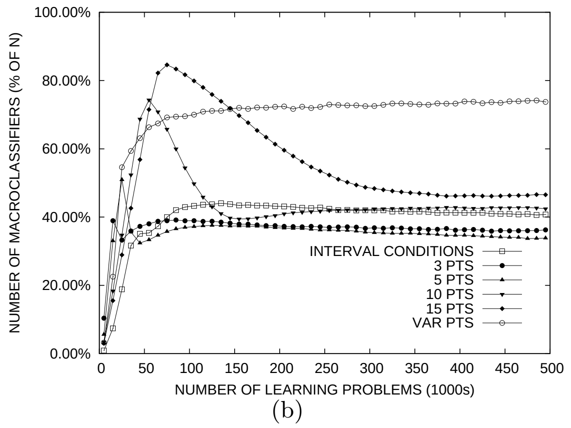


Figure 8: XCSF with constrained and unconstrained convex conditions applied to f_2: (a) prediction error; (b) number of classifiers in the populations. Curves are averages over 20 runs.

Figure 9: XCSF with convex conditions and quadratic approximation applied to f_3: (a) prediction error; (b) number of classifiers in the populations. Curves are averages over 20 runs.