The Class Imbalance Problem in Learning Classifier Systems: A Preliminary Study

[Extended Abstract]

Albert Orriols
Enginyeria i Arquitectura La Salle
Universitat Ramon Llull
Quatre Camins, 2. 08022 Barcelona, Spain.
aorriols@salleurl.edu

Ester Bernadó-Mansilla
Enginyeria i Arquitectura La Salle
Universitat Ramon Llull
Quatre Camins, 2. 08022 Barcelona, Spain.
esterb@salleurl.edu

ABSTRACT

The class imbalance problem has been said recently to hinder the performance of learning systems. In fact, many of them are designed with the assumption of well-balanced datasets. However, it is very common to find higher presence of one of the classes in real classification problems. The aim of this paper is to make a preliminary analysis on the effect of the class imbalance problem in learning classifier systems. Particularly we focus our study on UCS, a supervised version of XCS classifier system. We analyze UCS's behavior on unbalanced datasets and find that UCS is sensitive to high levels of class imbalance. We study strategies for dealing with class imbalances, acting either at the sampling level or at the classifier system's level.

Categories and Subject Descriptors

I.2.6 [**Learning**]: concept learning, knowledge adquisition

General Terms

Algorithms, Experimentation

Keywords

Evolutionary Computation, Genetic Algorithms, Machine Learning, Learning Classifier Systems, Class Imbalance

1. INTRODUCTION

Learning Classifiers Systems (LCSs) [6] are rule-based systems that are shown to perform very competitively with respect to other machine learning methods in classification problems. Nowadays, XCS [11, 12], an evolutionary online learning system, is one of the best representatives of LCSs.

This work focuses on one of the complexity factors which is said to hinder the performance of standard learning methods: the *class imbalance* problem. The class imbalance problem corresponds to classification domains for which one class is represented by a larger number of instances than other classes. The problem is of great importance since it appears in a large number of real domains, such as fraud detection [5], text classification [3], and medical diagnosis [8]. Traditional machine learning approaches may be biased towards the majority class and thus, may predict poorly the minority class examples. Recently, the machine learning community has paid increasing attention to this problem and how it affects the learning performance of some well-known classifier systems such as *C5.0*, *MPL*, and *support vector machines* [9]. The aim of this paper is to bring this analysis to the LCS's framework, and debate whether this problem affects LCSs, to what degree, and, if it is necessary, study different approaches to overcome the difficulties.

Our analysis is centered on Michigan-style learning classifier systems. We choose UCS [4] as the test classifier system for our analysis, with the expectation that our results and conclusions can also be extended to XCS and other similar LCSs. UCS is a version of XCS specifically designed for supervised classification problems. In order to isolate the class imbalance problem and control its degree of complexity, we design two artificial domains. We study UCS's behavior on these problems and identify factors of complexity when the class imbalance is high, which leads us to analyze different approaches to deal with these difficulties.

The remainder of this paper is organized as follows. Section 2 describes UCS briefly. Section 3 analyzes UCS's behavior on imbalance datasets. Then, we propose strategies for dealing with class imbalances and analyze these approaches under UCS's framework. Finally, we summarize our main conclusions.

2. DESCRIPTION OF UCS

UCS [1, 4] is a classifier system derived from XCS [11, 12] specifically designed for supervised learning problems.

UCS works as follows. UCS codifies a population of classifiers, each having a rule and a set of parameters estimating its quality. In *training* mode, UCS receives instances from the training dataset. Each instance has a set of attributes and an associated class. Then, UCS finds the matching classifiers, and builds the match set [M]. Those classifiers in [M] predicting the correct classification form the correct set [C]. If [C] is empty, then covering is triggered, creating

GECCO'05 June 25–29, 2005, Washington, DC, USA.

a new rule matching correctly the input example. Then, the parameters of the classifiers present in [M] are updated. The main parameters are *accuracy*, computed as the ratio of correct classifications to the number of covered examples, and *fitness*, which is a function of accuracy. We also compute the *experience* of each classifier as the number of times that the classifier has been active. The search mechanism is performed by a genetic algorithm (GA), which tries to guide the search towards accurate and maximally general rules. The GA is applied locally to [C] and selects two parents with probability proportional to fitness. After applying crossover and mutation, it introduces the offspring into the population, deleting other classifiers if the population is full. Subsumption is also applied to further favor the generalization of rules. In *test* mode, an input is presented and UCS must predict the correct class. In this case, after building the match set [M], UCS selects the best class from the vote (weighted by fitness) of all classifiers present in [M]. For more details, the reader is referred to [4].

3. UCS IN UNBALANCED DATASETS

In order to isolate the class imbalance problem from other factors that affect UCS's performance, we designed an artificial domain. The domain has two real attributes which can take values in the interval [0,1]. Instances are grouped in two non-overlapping classes, drawing a checkerboard in the feature space. We can vary the complexity along three different dimensions: the degree of *concept complexity* (c), which defines the number of squares in the checkerboard (it has c^2 squares); the *dataset size* (s); and the *imbalance level* between the two classes (i), which specifies the ratio between the number of minority and majority class instances.

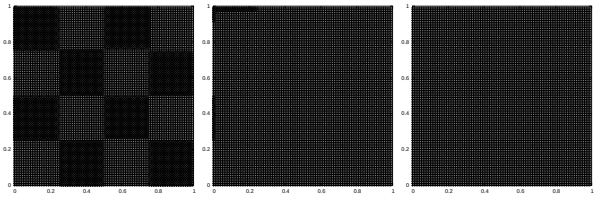
The generation process creates a well balanced dataset by randomly drawing s/c^2 points in each checkerboard square. Then, each unbalanced dataset of level i is obtained by taking out half of the minority class instances from the dataset at level $i-1$. For $i=0$ the dataset is well balanced.

In order to analyze the performance of UCS over unbalanced datasets, we ran UCS in the checkerboard (*chk*) problem with s=4096, c=4, which corresponds to sixteen alternating squares, and complexity levels from $i=1$ to $i=7$. UCS was trained with each of the datasets for 200000 learning iterations, with the following parameter settings (see [7] for the notation): N=400, α=0.1, β=0.2, δ=0.1, ν=10, θ_{del}=20, θ_{sub} = 20, acc_0=0.99, χ=0.8, μ=0.04, θ_{GA}=25, GASub = true, [A]Sub=false, Specify=true, Ns =20, Ps =0.5.

Figure 1 shows the classification boundaries obtained by UCS for imbalance levels 3, 4 and 5. Figure 1(a) shows the model evolved by UCS at imbalance level $i=3$. The model evolved for lower imbalance levels is not shown as it is mostly the same as that drawn for $i=3$. In all these cases, UCS is able to discover the optimal ruleset in spite of the class imbalances.

The problem arises with imbalance levels equal to or greater than 4. Figure 1(b) shows that the system is not able to classify correctly any of the minority class squares. Almost all the feature space is classified with the majority class. This behavior is also observed for higher imbalance levels. For imbalance levels $i=\{5,6,7\}$, UCS evolves models that classify all instances as they belonged to the majority class.

A deeper analysis of the population evolved in these high imbalance levels reveals that UCS is able to discover accurate and maximally general classifiers that predict the minority class regions. Nevertheless, their vote in the prediction array is not strong enough to encourage the correct classification of those regions with the minority class. This effect is due to the presence of overgeneral classifiers in the population predicting the majority class. These overgeneral rules cover nearly all the feature space, overcoming jointly the vote of the minority class rules. Specifically, the most numerous rules evolved by UCS at imbalance level $i=4$ are those that predict the eight minority class regions of the feature space. All of them are accurate and maximally general. An example of this type is: $x \in [0.509, 0.750]$ *and* $y \in [0.259, 0.492]$ *then class=1*, where 1 is the minority class. In addition, the population contains some overgeneral rules of type $x \in [0,1]$ *and* $y \in [0,1]$ *then class=0*, where 0 is the majority class. The population contains 32 microclassifiers[1] covering each minority class square, and more that 100 microclassifiers covering all the feature space with the majority class. Therefore, when classifiers vote for the class of a given example, the vote of such a huge number of overgenerals voting for the majority class overcomes the vote of the minority class.



(a) i=3 (b) i=4 (c) i=5

Figure 1: Boundaries evolved by UCS in the *chk* problem with imbalance levels of 3, 4 and 5. Black regions belong to the minority class and gray regions belong to the majority class.

We hypothesize that the presence of these overgeneral rules is due to the generalization pressure induced by the GA. Once created, these rules are activated in nearly any action set, because of their overgeneral condition. In balanced or low-unbalanced datasets, these overgeneral rules tend to have low accuracy and consequently, low fitness. For example, the most general rule has a 0.50 of accuracy in a balanced dataset. Thus, overgeneral rules with low fitness tend to have low probabilities of participating in reproductive events and finally, they are removed from the population. The problem comes out with high imbalance levels, where overgeneral rules have a high tendency to be maintained in the population. The reason is that the data distribution does not allow to penalize the classifier's accuracy so much, as long as the minority class instances are sampled in a lower frequency. So, the higher the imbalance level, the more accurate an overgeneral classifier is considered (and also the higher fitness it has). Consequently, in high imbalance levels overgeneral rules tend to have higher

[1]A *microclassifier* refers to the regular classifier. We distinguish it from the term *macroclassifier*, which refers to a classifier containing several copies of microclassifiers, sharing identical conditions, actions and parameters.

accuracies, presenting more opportunities to be selected by the GA, and also lower probabilities of being removed by the deletion procedure.

4. STRATEGIES FOR DEALING WITH CLASS IMBALANCES

In the literature, several strategies for dealing with class imbalances have been proposed [9]. Some of them are based on resampling the training dataset so that the classifier system receives the same proportion of examples per class, either by *oversampling* the minority class examples or *undersampling* the majority class examples. Other strategies alter the relative costs of misclassifying each of the classes. We have designed and adapted these strategies for UCS. We describe and analyze them in the following.

4.1 Random Oversampling

Random oversampling resamples at random the minority class examples until their number is equal to the number of instances in the majority class. In this case, we only modify the training dataset so that UCS perceives a balanced dataset.

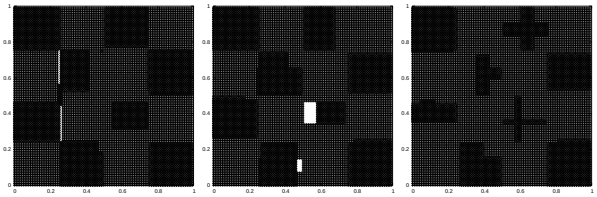
We tested UCS in the checkerboard problem for imbalance levels from i=0 to i=7, using the same parameter settings as in the previous section.

Figure 2 shows the boundaries evolved by UCS under random oversampling with imbalance levels from 5 to 7. The results for the lower imbalance levels are not shown for brevity. They correspond to cases where the model evolved by UCS is accurate (i.e., both classes are predicted correctly). For the highest imbalance levels, note that UCS has been able to evolve some of the boundaries belonging to the minority class examples. In many cases, these boundaries do not reach the real boundaries of the original balanced dataset. But this result is reasonable since the distribution of training points has changed with respect to the original dataset.

Under oversampling, UCS works as the problem was a well balanced dataset, because the proportion of minority class examples has been adjusted a priori. UCS sees a dataset with the same number examples per class, but with some gaps in the feature space that are not represented by any example. These gaps are mostly covered by rules predicting the majority class rather than by minority class rules. In fact, rules from both classes tend to expand as much as possible into these gaps until they reach points belonging to the opposite class. That is, rules tend to expand as long as they are accurate. Thus, there are overlapping rules belonging to different classes in the regions that are not covered by any example. When we test UCS in these regions, the majority class rules have higher numerosities and their vote into the prediction array is higher. Consequently, these regions get classified by the majority class.

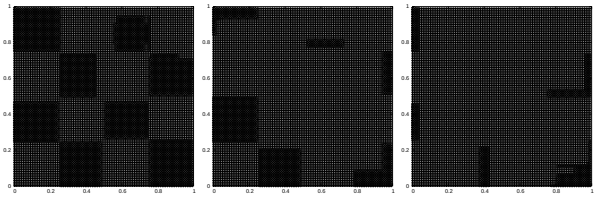
4.2 Adaptive Sampling

This method consists of adapting the sampling probabilities of the training examples according to their classification [2]. To be exact, the method maintains a weight for each instance, which estimates the sampling probability of the instance. Weights are updated incrementally when the system performs a prediction under test mode. If the instance is well classified, then the weight is decreased by a proportion α of its value; otherwise, the weight is increased by the same proportion α. In these experiments, α was set to 0.1.



(a) i=5 (b) i=6 (c) i=7

Figure 2: Class boundaries evolved by UCS with random oversampling in the chk problem with imbalance levels from 5 to 7.



(a) i=5 (b) i=6 (c) i=7

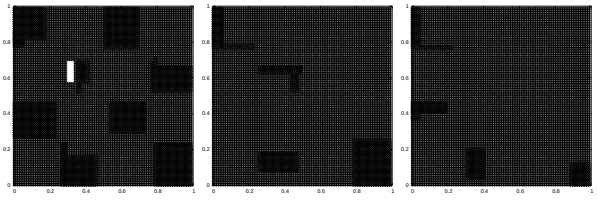
Figure 3: Class boundaries evolved by UCS with adaptive sampling in the chk problem with imbalance levels from 5 to 7.

Figure 3 shows the results obtained by UCS under the adaptive sampling strategy with imbalance levels from 5 to 7. The results for the lowest imbalance levels are not shown for brevity. They correspond to very accurate approximations of the real boundaries. For i=5, the boundaries evolved by UCS are fairly accurate. For higher imbalance levels, UCS has found more difficulties in finding good approximations for the minority class squares. In these cases, the result achieved under the adaptive sampling strategy is worse than that achieved by UCS under oversampling (see figure 2).

Analyzing the behavior of UCS under these two strategies (not detailed for brevity), we found that under adaptive sampling there is less generalization pressure towards the minority class rules than with oversampling. The reason is that, with adaptive sampling, once all instances are well classified, weights stabilize and then, all instances are sampled as the original a priori probabilities. Under oversampling, minority class instances are always sampled at the same a priori probability as majority class instances, keeping the same generalization pressure towards both classes. This may justify why, under adaptive sampling, UCS finds more difficulties to generalize the minority class rules, especially for the highest imbalance levels.

4.3 Class-Sensitive Accuracy

Class-sensitive accuracy modifies the way in which UCS computes accuracy so that each class is considered equally important regardless of the number of examples representing each class. In the UCS's original design, accuracy is computed as the ratio of the number of correctly classified examples to the number of covered examples. Herein, we compute the accuracy for each class separately and average the result. Additionally, we smooth this computation by a



(a) i=5 (b) i=6 (c) i=7

Figure 4: Class boundaries evolved by UCS with class-sensitive accuracy in the chk problem with imbalance levels from 5 to 7.

function of class experience to give opportunities to rules approaching class boundaries (see [10] for the details).

Figure 4 shows the results of UCS under class-sensitive accuracy. Note that the boundaries evolved in all imbalance levels are better at discovering minority class regions than those evolved by raw UCS.

For imbalance level i=5, UCS with class-sensitive accuracy clearly improves raw UCS in terms of the boundaries evolved. Furthermore, the tendency of evolving overgeneral rules, as mentioned in section 3, has been restrained. The population evolved by UCS (not detailed for brevity) contains accurate and maximally general rules predicting each of the squares of the checkerboard problem, both the minority class and the majority class squares.

Finally, figures 4(b) and 4(c) show the models evolved with imbalance levels i=6 and i=7 respectively. As the imbalance level increases, the system finds it harder to evolve the minority class regions. For the highest class imbalance, UCS can only draw partially four of the minority class regions. Looking at the evolved population, not shown for brevity, we confirm that the problem is not attributable to the evolution of overgeneral rules but to the fact that the imbalance ratio is so high (1:128) that the imbalance problem almost becomes a sparsity problem. There are so few representatives of the minority class regions that we may debate whether these points are representative of a sparse class region or whether they belong to noisy cases. In the latter case, we would agree that UCS should not find any distinctive region.

4.4 Further Results

To further test the strategies for dealing with class imbalances, we ran these strategies with another kind of class imbalance problem. Particularly, we select the position problem (denoted as *pos*) from [4] for presenting multiple classes and different proportions of examples per class. The problem is defined as follows. Given a binary input x of fixed length l, the output class corresponds to the position of the leftmost one-valued bit. If there is not any one-valued bit, the class is zero. The length of the string l determines both the complexity of the problem and the imbalance level.

We ran UCS with imbalance levels from l=8 until l=15. Figure 5 depicts the percentage of the optimal population[2] achieved by UCS during training. Curves are averages over five runs. Note that UCS has difficulties in learning all the optimal classifiers as the condition length grows. Particularly, UCS can not discover the most specific rules, because they cover very few examples.

We tested whether the strategies for dealing with class imbalances could improve this result and helped UCS discover the minority class rules. Figures 6, 7 and 8 show the results of UCS under oversampling, adaptive sampling and class-sensitive accuracy respectively.

Figure 6 shows that under oversampling, there are high oscillations in the learning curves of UCS. Making a deeper analysis, some harmful traits of oversampling, which were not observed in the *chk* problem, come out. In the *pos* problem, changing the a priori probabilities of examples makes accurate generalizations very hard to be evolved. UCS learns accurate and maximally general rules by the presence of the appropriate examples and counter-examples. While the presence of numerous examples favor the generalization of rules, counter-examples set the limit for these generalizations. If rules overgeneralize, the presence of counter-examples makes the rule inaccurate. In the *pos* problem, we oversample the minority class examples. Thus, the system gets a higher number of examples for the minority class rules, but on the contrary receives few proportion of counter-examples for these rules. The result is that UCS tends to overgeneralize the rules covering the minority class examples. And the discovery of specific rules for the minority class examples remains unsolved. This problem did not arise in the *chk* problem, because this problem has originally the same generalization for each of the rules. Oversampling makes each rule to receive the same proportion of examples and counter-examples. So in this case it is easier to find accurate generalizations.

Figure 7 shows the percentage of optimal population achieved by UCS under adaptive sampling. See that the oversampling effect does not appear here. If a rule is inaccurate because it does not classify properly an example, the probability of sampling that example is increased. Thus, this example will serve as a counter-example for overgeneral rules, and as an example to help discover rules covering it accurately. Note that the learning curves have improved with respect to the original problem, although there is still a high difficulty in discovering the optimal populations for the highest complex levels.

Figure 8 shows the percentage of optimal population evolved by UCS with class-sensitive accuracy. The figure does not reveal significant improvements with respect to raw UCS. The problem here is that UCS is receiving very few instances of the most specific classes. Thus, even though UCS weighs the contribution of each class equally, if the minority class instances come very sparsely, rules covering them have few reproductive events.

5. CONCLUSIONS

We analyzed the class imbalance problem in UCS classifier system. We found that UCS has a bias towards the majority class, especially for high degrees of class imbalances. Isolating the class imbalance problem by means of artificially designed problems, we were able to explain this bias in terms of the population evolved by UCS.

In the checkerboard problem, we identified the presence of overgeneral rules predicting the majority class which covered almost all the feature space. For a given imbalance level, these rules overcame the vote of the most specific ones

[2]The *optimal population* for a classification problem is the ruleset formed by accurate and maximally general rules.

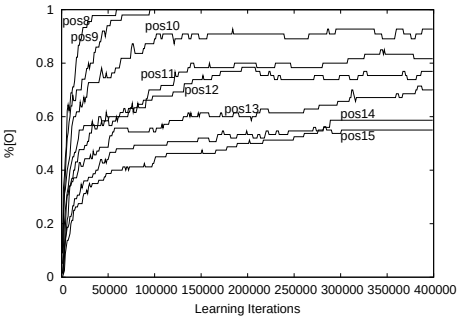


Figure 5: Percentage of optimal population evolved by UCS in the *pos* problem, ranging from l=8 to l=15.

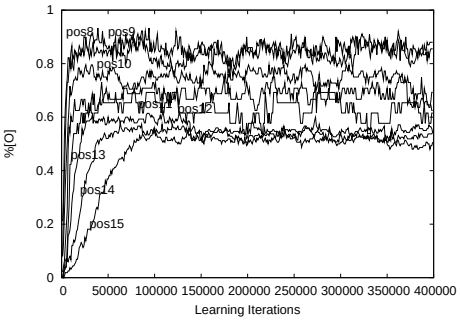


Figure 6: Percentage of optimal population evolved by UCS under oversampling in the *pos* problem, ranging from l=8 to l=15.

and thus, UCS predicted all instances as belonging to the majority class. Different strategies were analyzed to prevent the evolution of these overgeneral rules. We found that all tested strategies (oversampling, adaptive sampling, and class-sensitive accuracy) prevented UCS from evolving these overgeneral rules, and class boundaries, for both the minority and majority classes, were approximated fairly better than with the original setting. However, the analysis on the position problem revealed many inconveniences in the oversampling strategy which make UCS's learning unstable. This leads us to discard this method for real-world datasets.

The study would be much enhanced with the analysis of the class imbalance problem on other classifier schemes. Also, we would like to extend this analysis to other artificial problems, as well as to real-world datasets.

Acknowledgements

The authors thank the support of *Enginyeria i Arquitectura La Salle*, Ramon Llull University, as well as the support of *Ministerio de Ciencia y Tecnología* under project TIC2002-04036-C05-03, and *Generalitat de Catalunya* under Grant 2002SGR-00155.

6. REFERENCES

[1] Ester Bernadó. *Contributions to Genetic Based Classifier Systems*. PhD thesis, Enginyeria i Arquitectura la Salle, Ramon Llull University, Barcelona, 2002.

[2] Ester Bernadó-Mansilla. Complejidad del Aprendizaje y Muestreo de Ejemplos en Sistemas Clasificadores. In *Tercer Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'2004)*, pages 203–210, 2004.

[3] N. Chawla, K. Bowyer, L Hall, and W. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[4] Ester Bernadó and Josep M. Garrell. Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks. *Evolutionary Computation*, 11(3):209–238, 2003.

[5] R.E. Fawcett and F. Provost. Adaptive Fraud Detection. *Data Mining and Knowledge Discovery*, 3(1):291–316, 1997.

[6] John H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[7] Martin V. Butz and Stewart W. Wilson. An algorithmic description of XCS. In *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop*, volume 1996 of *Lecture Notes in Artificial Intelligence*, pages 253–272. Springer, 2001.

[8] P.M. Murphy and D.W. Aha. UCI Repository of machine learning databases, University of California at Irvine, Department of Information and Computer Science, 1994.

[9] Nathalie Japkowicz and Shaju Stephen. The Class Imbalance Problem: A Systematic Study. *Intelligent Data Analysis*, 6(5):429–450, November 2002.

[10] Albert Orriols and Ester Bernadó-Mansilla. Data Imbalance in UCS Classifier System: Fitness Adaptation. In *2005 IEEE Congress on Evolutionary Computation*, submitted.

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

[12] Wilson, Stewart W. Generalization in the XCS Classifier System. In *Genetic Programming: Proceedings of the Third Annual Conference*, pages 665–674. Morgan Kaufmann, 1998.

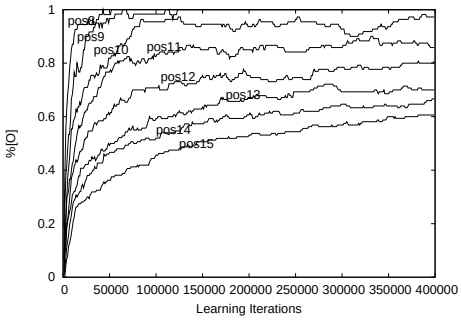


Figure 7: Percentage of optimal population evolved by UCS under adaptive sampling in the *pos* problem, ranging from l=8 to l=15.

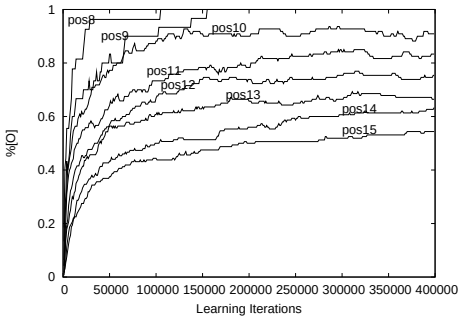


Figure 8: Percentage of optimal population evolved by UCS with class-sensitive accuracy in the *pos* problem, ranging from l=8 to l=15.