Discovery in Hydrating Plaster Using Multiple Machine Learning Methods

Judith E. Devaney, John G. Hagedorn

National Institute of Standards and Technology, Gaithersburg MD, 20899-8951, USA,
{judith.devaney,john.hagedorn}@nist.gov,
WWW home page: http://math.nist.gov/mcsd/savg

Abstract. We apply multiple machine learning methods to obtain concise rules that are highly predictive of scientifically meaningful classes in hydrating plaster over multiple time periods. We use three dimensional data obtained through X-ray microtomography at greater than one micron resolution per voxel at five times in the hydration process: powder, after 4 hours, 7 hours, 15.5 hours, and after 6 days of hydration. Using statistics based on locality, we create vectors containing eight attributes for subsets of size 100^3 of the data and use the autoclass unsupervised classification system to label the attribute vectors into three separate classes. Following this, we use the C5 decision tree software to separate the three classes into two parts: class 0 and 1, and class 0 and 2. We use our locally developed procedural genetic programming system, GPP, to create simple rules for these. The resulting collection of simple rules are tested on a separate 100^3 subset of the plaster datasets that had been labeled with their autoclass predictions. The rules were found to have both high sensitivity and high positive predictive value. Using multiple machine learning methods we were able to go from unlabeled data to simple rules in a very straightforward manner.

1 Introduction

Plaster of paris is a widely used material of economic importance [1]. For example, the porcelain industry maintains large numbers of molds whose strength, durability, and ability to absorb water impact the industry's costs [2]. Plaster powder is formed by calcining gypsum (calcium sulfate dihydrate, $CaSO_4{\cdot}2H_2O$) to form calcium sulfate hemihydrate ($CaSO_4 \cdot \frac{1}{2}H_2O$). The solid plaster is then formed by adding water (hydration) to the powder and allowing the mixture to set. The equations are [1]:

Calcination: $CaSO_4 \cdot 2H_2O = CaSO_4 \cdot \frac{1}{2}H_2O + \frac{3}{2}H_2O$

Hydration: $CaSO_4 \cdot \frac{1}{2}2H_2O + \frac{3}{2}H_2O = CaSO_4 \cdot 2H_2O$

During hydration, an interlocking network of gypsum crystals forms. See Figure 1 for a scanning electron micrograph (900X) [3] of precipitated gypsum crystals ($CaSO_4{\cdot}2H_2O$). This crystalline network is the foundation of the strength, durability, and absorptivity of the plaster [1]. However, the final properties of the set plaster are dependent on many things such as the water-solid ratio in hydration, impurities in the plaster, additives, temperature, and production conditions [3]. Moreover, these interact. For example, as the water solid-ratio in hydrating the plaster increases, the volume fraction of porosity increases, absorptivity of the plaster increases, but the strength and durability decrease [1]. There is much to learn about plaster; even the form of the kinetic equations (fraction of plaster reacted versus time) is not agreed upon [4][5][6]. Understanding the process of setting plaster as well as being able to predict its final properties is of scientific as well as economic interest.

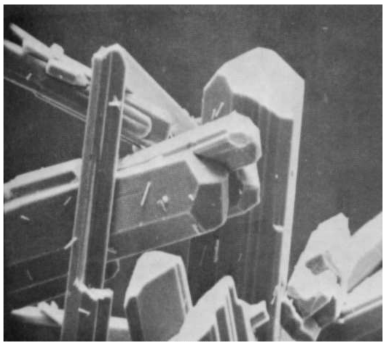

Fig. 1. A scanning electron micrograph (900X) [3] of precipitated gypsum crystals ($CaSO_4 \cdot 2H_2O$).

2 Methodology

Recently, X-ray microtomography has been used to obtain the unprecedented resolution of 0.95μ per voxel in three dimensional images of hydrating plaster. Commercial grade plaster of paris was mixed with a water-to-solids mass ratio of 1.0 and viewed with X-ray microtomography after 4, 7, 15.5 hours and 6 days. Additionally, a sample of plaster powered was imaged. This resulted in five images of plaster of size 1024^3. This is gray scale data with each pixel varying from 0 to 255 [7][8].

We seek simple rules to describe and predict states in hydrating plaster. We use multiple methods from machine learning to obtain the rules. We use a combination of unsupervised classification, decision trees, and genetic programming to obtain these rules. The rules are developed on a 100^3 subset of the data taken at the same place in each dataset. The rules are tested on a completely separate 100^3 subset of the data taken at the same place in each dataset.

2.1 Unsupervised Classification

Since the data is unlabeled, we start the discovery process with an unsupervised classifier. We use autoclass [9][10][11] which has been used successfully for discovery. In the creation of attributes for input to autoclass, we follow the *Principle of locality* [12], wherein natural laws are viewed as the consequence of small-scale regularities. Since the particle sizes may be as small as a few microns [8], we choose as our scale a 3^3 cube centered on each pixel in the image. Using simple statistics on these small cubes, we create eight attributes for each pixel as input vectors to autoclass, as described in the following table.

Attribute Name	Attribute Definition
A0	gray level value of pixel itself
A1	neighborhood midrange
A2	neighborhood variance about midrange
A3	neighborhood range
A4	neighborhood minimum
A5	neighborhood maximum
A6	neighborhood median
A7	neighborhood mean

Hence, for the 100^3 training subcube, this results in $1,000,000$ vectors. Since we are interested in the science and materials scientists are interested in three classes [13], we constrain autoclass to seek three classes. Classification runs were performed for each training subcube for powder, 4 hours, 7 hours, 15.5 hours, and 6 days. Since this data is at a new resolution, we do not have pre-labeled data to compare it with, or experts who can label it. We validate the classification through a visual comparison with the classes obtained. The 100^3 dataset used for training is small enough to look at all one hundred 100^2 images in the dataset and in the classification. A hundred 100^2 images can be printed on an $8\frac{1}{2}$ by 11 page in a ten by ten array making comparisons straightforward. Due to space considerations, we reproduce three images from each array here. Each image is taken at $\mathbf{z} = 0, 30, 60$ in the image array. Figure 2 shows the data and classes for the plaster powder. The data is on the left, with $\mathbf{z} = 0$ at the bottom, $\mathbf{z} = 30$ in the middle, and $\mathbf{z} = 60$ at the top. The corresponding classification for each plane is to the right of the data. Class 0 is black, class 1 is gray, and class 2 is white. It is immediately obvious that autoclass has picked up the basic structure of the data. The plaster particles are class 1. Figure 3 shows the equivalent images for 4, 7, 15,5 hours and 6 days of hydration. Again the structure in the data matches the structure of the classification. Class 1 is the crystalline network and unhydrated plaster. Class 2 is the porosity (voids), and class 0 is the boundary region. Figure 4 shows this difference clearly in renderings of the individual classes in their three dimensional configuration. Plaster hardens with little change in external volume, but since the volume of the hydration products is smaller than the original material plus water, voids occur inside [14].

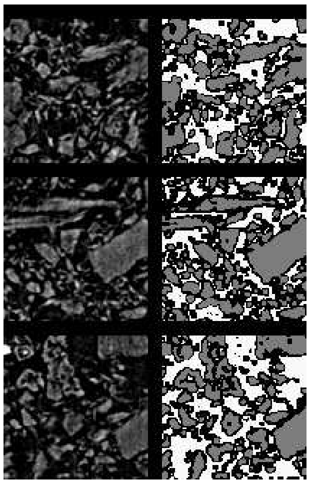

Fig. 2. Comparison of three 100^2 slices of original data with corresponding classification for powder. The data is in the left column, with $\mathbf{z} = 0$ at the bottom, $\mathbf{z} = 30$ in the middle, and $\mathbf{z} = 60$ at the top. The corresponding classification for each plane is to the right of the data.

2.2 Decision Tree

In order to gain better insight into the classifications for each time period, we seek comprehensible representations of the classification algorithms. That is, we wanted to find relatively simple rules to determine each element's class based on its eight attributes.

Autoclass operates in a "black-box" fashion. The algorithm by which it classifies and predicts elements is opaque to the user. To derive more transparent statements of the classification schemes, we used a decision tree, C5 [15][1]. C5 is the commercial successor to C4.5 [16], which has been used extensively for learning.

Runs of C5 on the autoclass labeled attributes produced incomprehensible trees with thousands of nodes. However, the component classes in the brightness histograms in Figure 5 indicated that class 2 and class 1 might be easily separable. Ten fold cross validation on the combined class 1 and 2 showed that this was the case in four of the five datasets. Three of the datasets yielded single node decision stumps with less than five misclassifications over hundreds of thousands of cases for powder, 4 hours, and 15.5 hours. All of these branched on attribute A1. The fourth simple classification was for the 7 hour dataset. This also yielded a single node decision stump; however, this branched on A7. For uniformity in

[1] The identification of any commercial product or trade name does not imply endorsement or recommendation by either the National Institute of Standards and Technology.

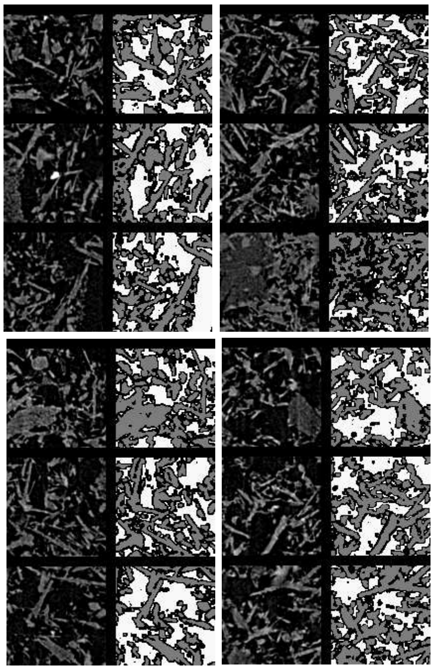

Fig. 3. Comparisons of original data with classification for 4 hours (top left), 7 hours (top right), 15.5 hours (bottom left), and 6 days (bottom right). For each time, three 100^2 slices of original data are in the left column, with $\mathbf{z} = 0$ at the bottom, $\mathbf{z} = 30$ in the middle, and $\mathbf{z} = 60$ at the top. The corresponding classification for each plane is to the right of the data.

the final rules across the hydration times, the 7 hour case was rerun requiring C5 to use only A1 to get a single best split on this attribute. The six day dataset did not yield a simple decision tree for the combined class 1 and 2. So it was also run with A1 as the only attribute to get the best split for input into the next phase, which was genetic programming to obtain complete and simple rules.

The attribute A1 is the local midrange. The midrange is a robust estimator of the center of a short tailed distribution [17]. Since the range is limited for each neighborhood to $0 - 255$, this is the situation here. So all the rules are now of the form: *if the center of the local distribution is* $\leq x$ *....*

2.3 Genetic Programming

Genetic Programming [18][19][20] is a technique that derives a sequence of operations (chosen from a user defined set) that have high fitness (as defined by the user) using techniques inspired by evolution. We use GPP (Genetic Program-

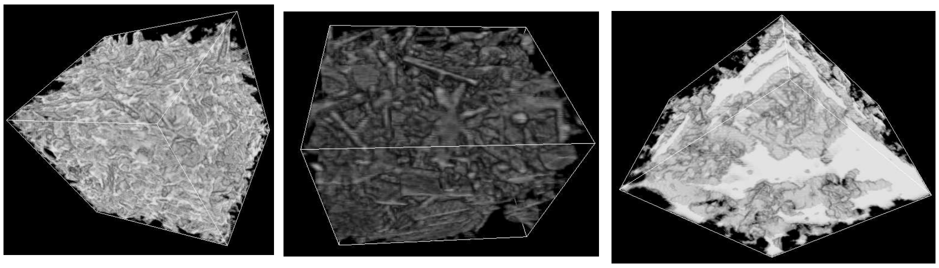

Fig. 4. Three dimensional renderings of individual classes (class 0 is on the left, class1 is in the middle, and class 2 is on the right) at 4 hours of hydration.

ming - Procedural) [21][22], a procedural genetic programming system that we have developed.

GPP was used to derive simple, understandable formulae that closely match the original classifications provided by autoclass. Before GPP is used, the class 1/2 decision algorithm is determined by C5 as described above. GPP is then used to derive the class 0/1 and the class 0/2 decision algorithms. The method for using GPP in this problem followed the following steps for each desired classification:
- Prepare training data sets from the classified data sets.
- Select parameters, such as the operator set, for the GPP runs.
- Construct a fitness function to measure algorithm effectiveness.
- Execute a set of GPP runs.
- Select the run with results that most closely match the original classification.
- Simplify the GPP-produced algorithm to a succinct form.

Training Data Set Preparation: The decision algorithms to be derived by GPP are be used as a second-tier decision after the application of the class 1/2 decision algorithm derived by C5. The class 1/2 decision algorithm classifies elements from classes 1 and 2 very accurately, but the algorithm also assigns elements from class 0 to either class 1 or 2. We create training sets for the GPP runs based on the actual input that the decision algorithms will receive. For example, to create the training data set for the class 0/1 decision algorithm we follow these steps:
- Apply the class 1/2 decision algorithm to class 0.
- Select a subset of class 0 elements classified as class 1 for inclusion in the training set.
- Select a subset of all class 1 elements for inclusion in the training set.

We follow the corresponding procedure for the class 0/2 training sets. For all runs the training sets have 200 elements: 100 elements from each of the two classes being considered.

GPP Run Parameters and Fitness Function: A variety of operating pa-

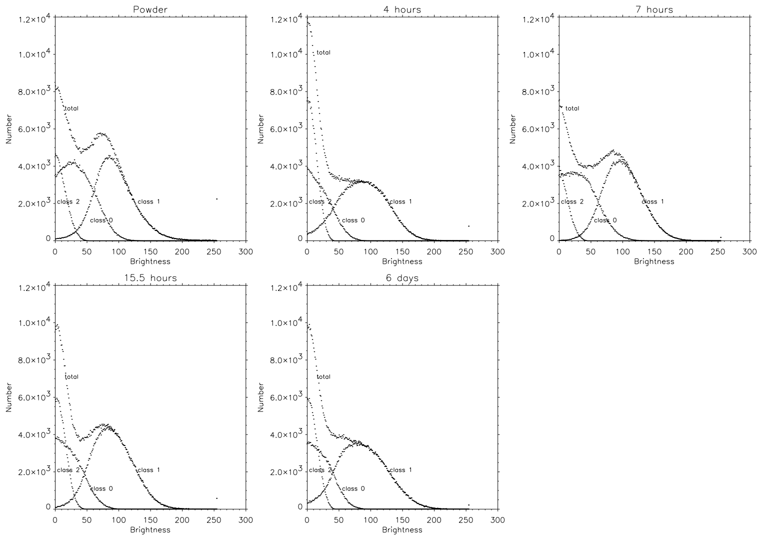

Fig. 5. Brightness histograms for whole datasets as well as component classes for powder, 4 hours, 7 hours, 15.5 hours, and 6 days.

rameters must be selected for the GPP runs. All runs were done with the same set of parameters. The parameters were selected based on knowledge of the problem to be solved, prior experience with GPP, and a few preliminary test runs. The primary parameters that were used for all runs are:
- Operators allowed in the programs: $+,-,*,/,<$,negate,mod, conditional assignment,and,or,not
- Population size: 500
- Maximum number of generations: 50
- Rates for genetic operations: (crossover : 10%), (mutation : 60%), (prune : 10%), (repair : 10%), (survival : 5%), (new individual : 5%)

The fitness function is based on the correlation between the algorithm's classifications with the actual classifications. The fact that this is a two-valued problem simplifies the calculation of the correlation. We use the formulation by Matthews[23], which has been used in the context of genetic programming by Koza[19]. The correlation is given by:

$$\frac{T_p T_n - F_n F_p}{\sqrt{(T_n + F_n)(T_n + F_p)(T_p + F_n)(T_p + F_p)}}$$

where: T_p is the number of true positives, T_n is the number of true negatives, F_p is the number of false positives, F_n is the number of false negatives

In this context, a positive determination refers to the classification of an element as class 1 or 2 depending on whether the run is for the class 0/1 decision or the class 0/2 decision. The correlation is evaluated based on the execution of an algorithm on the appropriate training set. This correlation value varies from -1 to 1, where 1 indicates perfect performance on the training set and -1 indicated a perfectly incorrect performance on the training set. Because these decision algorithms are binary in nature, we can turn a very bad algorithm into a very good algorithm simply by inverting each decision. In terms of the correlation value, this means that we can regard a decision algorithm with a correlation of -0.6 to be just as good as an algorithm with correlation +0.6. So our fitness function is the absolute value of the correlation value given above. Each GPP run seeks to evolve an algorithm that maximizes this fitness value.

Execution of the GPP Runs and Selection of the Best Run: For each decision algorithm to be derived, five hundred GPP runs were made. Each of these five hundred runs differed only in the seed to the random number generator. On a single processor a set of five hundred runs would typically complete in approximately 5 hours. Given the resources available to us, we easily ran all 12 sets of runs over a single night. After completing a set of five hundred runs we then have to select the best run for further consideration. Each of the runs produces an output file that contains the evolved decision algorithm in C++ form (although C++ is not the internal representation of the evolved programs). We implemented a post-processing procedure to aid us in finding the GPP run that produces the best result. For each of the five hundred runs, the procedure executes these steps:

- Extract the C++ version of the algorithm from the output file.
- Compile the C++.
- Run the program on the appropriate full data sets.
- Write out brief performance statistics for that program.

The performance statistics are then easily scanned to select the best run.

Simplification of the GPP Algorithms:

Once the best GPP run has been selected, we would like to express the evolved algorithm in a simple form. The GPP produced programs can be a bit obscure and often include elements that do not contribute to the final decision. For this reason, we seek to simplify the GPP evolved algorithms. This simplification process is, in part, a manual procedure, but it is aided greatly by the use of Maple [24], a symbolic computation system. Maple has many capabilities, but we use it in this project to help reduce algebraic expressions to simpler forms. As mentioned above, each GPP run produces an output file that contains a representation of the evolved program in C++ form. In addition to this C++ representation, GPP also writes out the algorithm in a Maple representation. Maple can operate on this representation of the algorithm and simplify to a single compact formula. There are some issues in this scenario because some of our internal program representations cannot be easily expressed in Maple's representation. In these cases, some manual intervention is required to complete the simplification. But this is not overly burdensome and Maple has proven to be invaluable in understanding and simplifying the GPP-evolved programs.

3 Results

Rules: The final classification rule are:

Powder: if $A1 \leq 42.$
    then *(class is either 2 or 0)*
        if $\lfloor .518494A3 + .019318A6 \rfloor = 2$
            then class = 0
            else class = 2
    else *(class is either 1 or 0)*
        if $(7235./A7) \leq (A7 + A4 + A1)$
            then class = 1
            else class = 0

4 hours: if $A1 \leq 27.$
    then *(class is either 2 or 0)*
        if $A6 = 0$
            then if $\lfloor 0.5 + .06145A1 \rfloor = 2$
                then class = 0
                else class = 2
            then if $\lfloor 0.5 + .06145A1 + .003373A3 \rfloor = 2$
                then class = 0
                else class = 2
    else *(class is either 1 or 0)*
        if $(6323.0/A1) < (A0 + A4 + 0.692129A6 + A7)$
            then class = 1
            else class = 0

7 hours:if $A1 \leq 42.5$
    then *(class is either 2 or 0)*
        if $\lfloor .527467A1 + .027467A7 \rfloor = 2$
            then class = 0
            else class = 2
    else *(class is either 1 or 0)*
        if $\lfloor 0.5 + 0.008515A7 \rfloor = 1$
            then class = 1
            else class = 0

15.5 hours: if $A1 \leq 30.$
    then *(class is either 2 or 0)*
        if $\lfloor .525849(A6 + A3) \rfloor = 2$
            then class = 0
            else class = 2
    else *(class is either 1 or 0)*
        if $A0 \neq 0$
            then if $A4 < (5321.0/A7) - (a0 + a7)$

then class = 0
else class = 1
else if $A4 < (5321.0/A7) - a3$
then class = 0
else class = 1

6 days: if $A1 \leq 28$.
then class = 2
else *(class is either 1 or 0)*
if $A0 + (A1 - (4643/A7)) > 0$
then class = 0
else class = 1

Evaluation of the Rules: We use *sensitivity* and *positive predictive value* [25] as metrics to evaluate our rules. A rule can be optimal with respect to a particular classification in two ways. The rule can be very successful at seeing a class when it is there. This is called its *sensitivity.* And the rule can be very successful at identifying the class in the presence of other classes. This is called its *positive predictive value.* Let T_p be the true positives. Let F_p be the false positives. Let F_n be the false negatives. Then:

$$\begin{aligned} \textit{Sensitivity} &= \tfrac{T_p}{T_p+F_n} \\ \textit{Positive-Predictive-Value} &= \tfrac{T_p}{T_p+F_p} \end{aligned}$$

In a confusion matrix, *sensitivity* is accuracy across a row; *positive predictive value* is accuracy down a column.

We test our classification rules with a completely different 100^3 subcube of data from each of the five time periods. To test the rules we first compute the same attribute vectors for the new dataset. Then we use the prediction capability of autoclass to label the vectors. Finally, we use the above rules to create confusion matrices of the predictions for each of the time periods. We derive the *sensitivity* and *positive predictive values* for class and time period. We also do a prediction for the rules derived for the split on A1 and on A7 for 7 hours of hydration for comparison purposes. The derived rules are all highly predictive as shown in the following table.

Dataset	Sensitivity			Positive-Predictive-Value		
	class 0	class 1	class 2	class 0	class 1	class 2
Powder	0.94	0.88	0.97	0.86	0.97	0.95
4 hours	0.97	0.96	0.97	0.93	0.98	0.997
7 hours	0.95	0.95	0.99	0.95	0.95	0.99
15.5 hours	0.96	0.93	0.98	0.93	0.96	0.996
6 days	0.97	0.96	0.996	0.99	0.95	0.98

4 Conclusions and Future Work

We have taken unclassified data and created a set of simple rules that accurately predict the class of unseen data for plaster powder, and plaster after 4, 7, 15.5 hours, and 6 days of hydration. Combining multiple machine learning methods enabled us to go from unlabeled data to simple rules in a very straightforward manner. Every learning algorithm has its strengths. By using them for different tasks, the end result can be achieved more easily.

The work on this data set has just begun, however. We would like rules to separate the unhydrated plaster from the gypsum crystals. Our plan is to use our genetic programming system to accomplish this. Then our next task is get an expert to label small subsets of our data. We would like to compare these labels to our classifications. This may result in some refinement of our rules. Next we would like to develop equations that accurately predict the class regardless of the time of hydration, i.e. that work over the whole hydration period. We will need additional data to include variations with respect to the parameters that can influence the setting process and the resultant properties of plaster. Finally, we would like to predict physical characteristics of classes with equations, instead of predicting classes. Plaster is an interesting and exciting topic for automated discovery methods. We look forward to extending our study.

Acknowledgements: We would like to thank Dale Bentz, Barbara Cuthill, and John Hewes and the NIST Advanced Technology Program for their support.

References

1. Kingrey, W.D., Bowen, H.K., Uhlmann, D.R.: Introduction to Ceramics. John Wiley and Sons, New York (1976)
2. Bullard, J.W.: Personal communication (2002)
3. Clifton, J.R.: Some aspects of the setting and hardening of gypsum plaster. Technical Note 755, NBS (1973)
4. Hand, R.J.: The kinetics of hydration of calcium sulphate hemihydrate: A critical comparison of the models in the literature. Cement and Concrete Research **24** (1994) 885–895
5. Ridge, M.J.: A discussion of the paper: The kinetics of hydration of calcium sulphate hemihydrate: A critical comparison of the models in the literature by r. j. hand. Cement and Concrete Research **25** (1995) 224
6. Hand, R.J.: A reply to a discussion by m. j. ridge of the paper: The kinetics of hydration of calcium sulphate hemihydrate: A critical comparison of the models in the literature. Cement and Concrete Research **25** (1995) 225–226
7. D. P. Bentz, S. Mizell, S. G. Satterfield, J. E. Devaney, W. L. George, P. M. Ketcham, J. Graham, J. Porterfield, D. Quenard, F. Vallee, H. Sallee, E. Boller, J. Baruchel: The Visible Cement Dataset. J. Res. Natl. Inst. Stand. Technol. **107** (2002) 137–148
8. : The visible cement dataset (2002) [online] <http://visiblecement.nist.gov> .
9. Cheeseman, R., Kelley, J., Self, M., Taylor, W., Freeman, D.: Autoclass: A bayesian classification system. In: Proceedings of the Fifth International Conference on Machine Learning, San Francisco, CA, Morgan Kaufman (1988) 65–74

10. Cheeseman, P.: On finding the most probable model. In Shrager, J., Langley, P., eds.: Computational Models of Discovery and Theory Formation. Morgan Kaufman, San Francisco, CA (1991) 73–96
11. Goebel, J., Volk, K., Walker, H., Gerbault, P., Cheeseman, P., Self, M., Stutc, J., Taylor, W.: A bayesian classifiection of the iras lrs atlas. Astronomy and Astrophysics **222** (1989) L5–L8
12. Reichenbach, H.: Atom and Cosmos. Dover Publications, Inc., Mineola (1932)
13. Bentz, D.P.: Personal communication (2002)
14. Sattler, H., Bruckner, H.P.: Changes in volume and density during the hydration of gypsum binders as a function of the quantity of water. ZKG **54** (2001) 522
15. : C5 (2002) [online] <http://www.rulequest.com> .
16. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Klauffann, San Mateo (1993)
17. Crow, E.L., Siddiqui, M.N.: Robust estimation of location. Journal of the American Statistical Associatiuon **63** (1967) 363–389
18. Koza, J.R.: Genetic Programming. MIT Press (1992)
19. Koza, J.R.: Genetic Programming II. MIT Press (1994)
20. Koza, J.R., Andre, D., Bennett III, F.H., Keane, M.: Genetic Programming III. Morgan Kaufman (1999)
21. Hagedorn, J., Devaney, J.: A genetic programming system with a procedural program representation. In: 2001 Genetic and Evolutionary Computation Conference Late Breaking Papers. (2001) 152–159 http://math.nist.gov/mcsd/savg/papers .
22. Devaney, J., Hagedorn, J., Nicolas, O., Garg, G., Samson, A., Michel, M.: A genetic programming ecosystem. In: Proceedings 15th International Parallel and Distributed Processing Symposium, IEEE Computer Society (2001) 131
23. Matthews, B.W.: Comparison of the predicted and observed secondary structure of t4 phage lysozyme. Biochimica et Biophysica Acta **405** (1975) 442–451
24. Monagan, M.B., Geddes, K.O., Heal, K.M., Labahn, G., Vorkoetter, S.M., McCarron, J.: Maple 6 Programming Guide. Waterloo Maple Inc., Waterloo (2000)
25. Lathrop, R., Erbdster, T., Smith, R., Winston, P., Smith, T.: Integrating ai with sequence analysis. In Hunter, L., ed.: AI and Molecular Biology, Cambridge (1993)