Parsimonious Regularization using Genetic Algorithms Applied to the Analysis of Analytical Ultracentrifugation Experiments

Emre H Brookes
Department of Computer Science
University of Texas at San Antonio
One UTSA Circle
San Antonio, TX 78249-1664 USA
ebrookes@cs.utsa.edu

Borries Demeler
Department of Biochemistry
University of Texas Health Science Center at
San Antonio
7703 Floyd Curl Drive
San Antonio, TX 78229-3900 USA
demeler@biochem.uthscsa.edu

ABSTRACT

Frequently in the physical sciences experimental data are analyzed to determine model parameters using techniques known as parameter estimation. Eliminating the effects of noise from experimental data often involves Tikhonov or Maximum-Entropy regularization. These methods introduce a bias which smoothes the solution. In the problems considered here, the exact answer is sharp, containing a sparse set of parameters. Therefore, it is desirable to find the simplest set of model parameters for the data with an equivalent goodness-of-fit. This paper explains how to bias the solution towards a parsimonious model with a careful application of Genetic Algorithms. A method of representation, initialization and mutation is introduced to efficiently find this model. The results are compared with results from two other methods on simulated data with known content. Our method is shown to be the only one to achieve the desired results. Analysis of Analytical Ultracentrifugation sedimentation velocity experimental data is the primary example application.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*Global optimization, Least squares methods*; J.3 [**Life and Medical Sciences**]: Biology and genetics

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Genetic algorithm, Inverse problem, Regularization, Analytical ultracentrifugation

GECCO'07, July 7–11, 2007 London, England, United Kingdom.
Copyright 2007 ACM 978-1-59593-697-4/07/0007 ...$5.00.

1. INTRODUCTION

Analytical Ultracentrifugation (AUC) is a powerful technique for determining hydrodynamic properties of biological macromolecules and synthetic polymers [7, 8, 18]. AUC can be used to identify the heterogeneity of a sample in both molecular weight and macromolecular shape. Since AUC experiments are conducted in solution, it is possible to observe macromolecules and macromolecular assemblies in a physiological environment, unconstrained by a crystal structure or electron microscope grid. Systems can be studied under high concentrations or under very dilute conditions, under virtually unlimited buffer conditions, and the methods are applicable to a very large range of molecular weights, extending from just a few hundred Daltons to systems as large as whole virus particles. Results of these studies can allow the researcher to follow assembly processes of multienzyme complexes, characterize recombinant proteins and assess sample purity before proceeding to NMR or X-ray crystallography experiments. The techniques addressed in the paper are currently being used in AUC studies focusing on macromolecular properties of systems related to disease, cancer and aging.

In AUC sedimentation velocity experiments a sample in solution contained in a sector shaped cell is placed in the ultracentrifuge. The ultracentrifuge is started and run at speeds from 2,000 to 60,000 RPM. At regular time intervals, the instrument records a radial concentration profile of the cell determined from light absorbance at a particular wavelength of light (see Figure 1). At the beginning of the experiment, the sample is uniformly distributed throughout the cell and therefore the first observation shows a uniform radial concentration profile. As the experiment progresses, the centripetal force, which can be as high as 230,000 g, causes the sample to sediment towards the bottom of the cell. After several hours or more, depending on the sample and the speed of the ultracentrifuge, the sample will be fully sedimented and further observations will contain an unchanging radial concentration profile of an exponential form. Radial concentration profiles are typically displayed superimposed as shown in Figure 2.

The sample may contain several solutes, each a different type of molecule present at some concentration. The behavior of an ideal solute is well described by a second order PDE known as the Lamm equation [14] and can be solved by finite element modeling (FEM) [6]. Given predetermined constant experimental parameters such as speed, temperature, viscosity and density of the solution, each solute's behavior can be described by a FEM solution of the Lamm equation of two parameters, the sedimentation coefficient s and frictional ratio k. The frictional ratio k is a measure of shape with a minimum value of 1 for a spherical molecule. Increasing values of k correspond to increasingly elongated molecules. An example set of parameters for a 3 solute system is shown in Table 1. Since superposition holds for multiple non-interacting solutes in such a setup, it is straightforward to simulate the experimental results for known multiple solute systems. It is much more difficult to determine the solute parameters, s and k, for unknown samples. If one can determine all solute parameters from experimental results, their molecular weights can be computed. It is very difficult to determine even the number of different types of solutes present. Knowing the number of solutes, their molecular weights, concentrations, and shapes is of primary importance to the researcher. Our techniques address these problems.

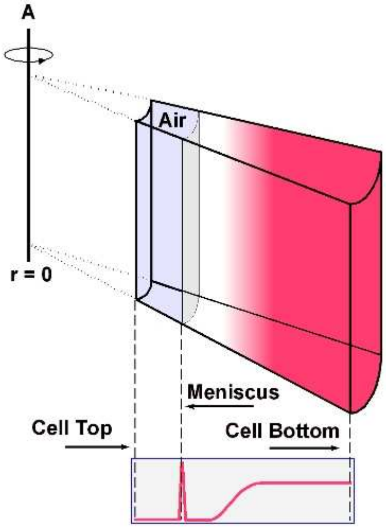

Figure 1: The basic AUC experimental setup. The sector-shaped cell is loaded with a sample and put into the ultracentrifuge. An observation is taken of the cell which creates data containing a radial concentration profile as shown at the bottom of the figure.

MW	s coefficient	frictional ratio k	concentration
1e4	1.3269e-13	1.3139	0.3
2e4	2.7675e-13	1	0.2
4e4	1.9214e-13	2.2865	0.4

Table 1: A 3 solute system listing molecular weight (MW) measured in Daltons, solute parameters s and k and the loading concentration for each solute. s and k are the target solute parameters which we wish to discover with our analyses, from which we compute MW and loading concentration.

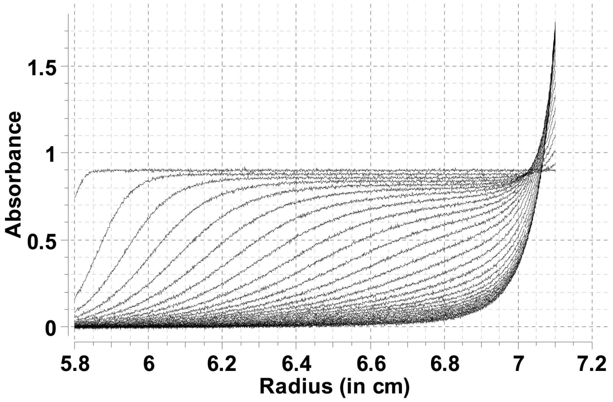

Figure 2: Typical experimental data of superimposed observations taken at regular time intervals. The horizontal axis corresponds to the radial axis along the cell, with the meniscus at approximately 5.8 cm and the bottom of the cell at 7.2cm. The vertical axis is the measured absorbance. The first observation has a mostly uniform absorbance of 0.9. These data resulted from the simulation of a 40k RPM run of 24 hours duration using values from Table 1. The program `UltraScan` [9] was used to simulate and display these data.

Mathematically, the experimental data are placed in a vector $\mathbf{b}$. The elements of $\mathbf{b}$ are the observed radial concentration profiles placed end-to-end for each time interval. For example, if each radial concentration profile contains r points, $\mathbf{b}[2r + 1]$ will contain the first radial concentration of the third observation. Similarly, solutions to the Lamm equation can be placed in vectors and collected into a matrix $\mathbf{A}$. Therefore, each column of $\mathbf{A}$ will be associated with solute parameters s and k used to solve the Lamm equation. Assuming the data contain normally distributed errors from a distribution of constant variance, the best fit solution vector $\mathbf{x}$, can be expressed as follows:

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_2 \qquad (1)$$

Due to noise, $\mathbf{b}$ is not generally in the range of $\mathbf{A}$, so that a best fit solution is required. After solving eq. (1) for $\mathbf{x}$, each element of $\mathbf{x}$ will contain the concentration of each solute associated with the corresponding column of $\mathbf{A}$. Since negative concentrations do not make physical sense, a nonnegatively constrained least squares minimization technique known as NNLS [15] is used to solve this equation.

Let U be the universal collection of sets of all possible solute parameters s and k. Let S be a finite subset of U. Then given experimental data $\mathbf{b}$, we can define a function:

$$f : S \mapsto (\mathbf{x}, \mathbb{R}) \qquad (2)$$

which takes the input set, builds the matrix $\mathbf{A}$, computes the NNLS solution of eq. (1), and returns $\mathbf{x}$ and the root mean square deviation (RMSD) of the vector difference between $\mathbf{Ax}$ and $\mathbf{b}$, a scalar measure of the goodness-of-fit. We introduce the following function:

$$nz : \mathbf{x} \mapsto \mathbb{N} \qquad (3)$$

which counts the number of nonzero elements of $\mathbf{x}$. Our goal is to find the parsimonious set of solute parameters to explain the data. This means that we wish to find the set S that minimizes $nz(\mathbf{x})$ and simultaneously maintains an RMSD that approximates the level of noise in the experiment.

Since we can not search all of U, we must select some subset S to search. Good solutions are not obtained if S does not contain representatives of all the solutes present in the sample. For example, if the sample contains two solutes, trying to solve f for just one of the solutes gives erroneous results. Constraining the search space is the first step towards application of all of the methods subsequently described. The range of s can be constrained by the van Holde-Weischet analysis [11]. Physical limits constrain the frictional ratio k to a minimum value of one (a spherical solute) and a maximum value from four for proteins to ten for elongated DNA chains.

One method known as *C(s)* [16] takes a discrete subset of values for s and uses a fixed value for k, then a one-dimensional line search is performed on the RMSD of eq. (2) over k. This method can be used when the value of k is identical for all solutes in the sample. It is often the case that the sample exhibits heterogeneity in k due to the presence of molecules with different shapes. Correct k values are needed to compute accurate molecular weights, which *C(s)* can not determine for the general case. Another method developed by the authors known as the two-dimensional spectrum analysis (2DSA) [3] places a two-dimensional grid over the solute parameters s and k and evaluates f. 2DSA can use moving grids and iterative searches to further refine the solution. For *C(s)* and 2DSA, the results are a set of parameters that describe these data. (see Figures 3, 4). Neither method can generally give an accurate assessment of the number of different solutes present in the sample. We will describe a method using a GA to find the parsimonious set of solute parameters that best describe the experimental data in section 2 of this paper.

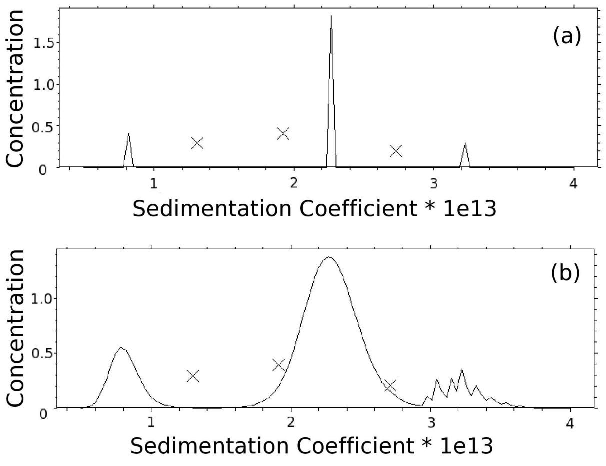

Figure 3: *C(s)* analysis of the 3 solute simulated system of Table 1 comparing results without regularization (a) and with regularization (b). X marks the target solute parameter s and concentration in the sample. The data were produced from the program `sedfit` [16].

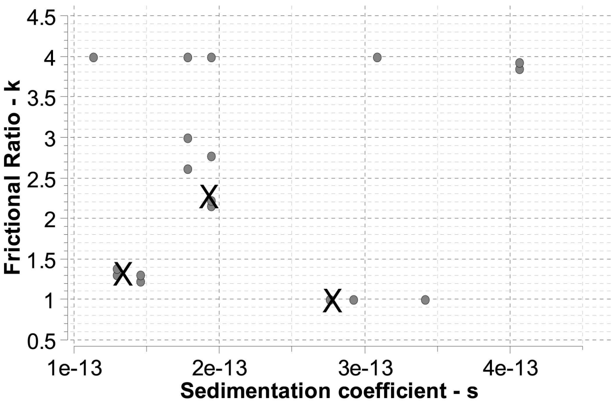

Figure 4: 2DSA analysis of the 3 solute simulated system of Table 1. X marks the target solute parameters s and k in the sample. The circles are possible solute parameters identified by the analysis.

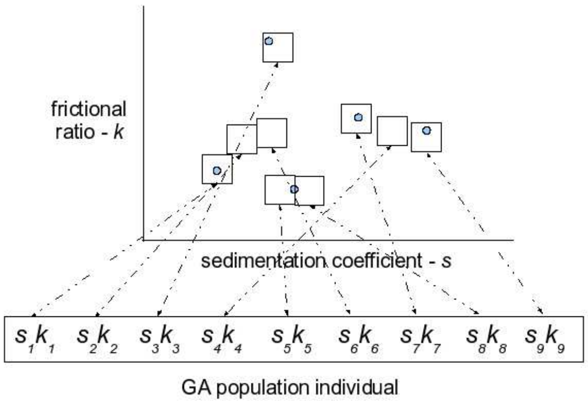

Figure 5: This is a pictorial representation of the association between the target solute parameters, a set of buckets and the string of floats of the population individual. Each target solute parameter is marked with a dot. Each bucket is a box centered around a solute parameter returned from a preliminary analysis method such as 2DSA. The population individual will be initialized and mutated with random values that are constrained by the values of the associated bucket throughout the evolution of the GA.

Experimental data contain noise which makes solving these systems more difficult. For example, a fingerprint on the cell often causes time invariant noise which can be removed in a preprocessing stage [17]. Random noise is present and can be as low as one percent on a well maintained ultracentrifuge. Nevertheless, noise will have an effect on the results obtained. This can result in false positive and inaccurate solute parameters. Methods known as Tikhonov and Maximum-Entropy regularization have been used to compensate for the effects of noise [2, 5, 16]. These methods penalize sharp peaks in the solution $\mathbf{x}$, introducing a bias which spreads the peaks. This increases the number of solute parameters returned by f (see Figure 3), making it more dif-

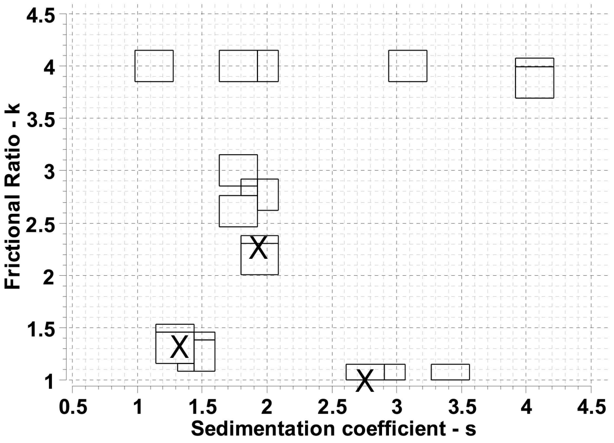

Figure 6: This graph shows the automatically assigned buckets for initialization and mutation constraints of GA individuals based upon the 2DSA analysis results of Figure 4. X marks the target solute parameters in the sample.

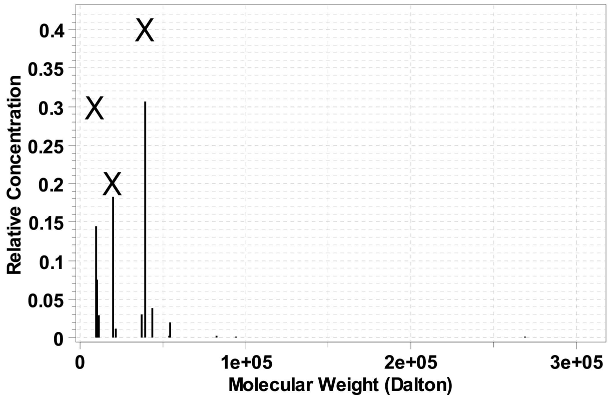

Figure 7: This is a another view of the 2DSA analysis of Figure 4. Although hard to see on this figure, there are some false positives of small concentration near 2.75e5 molecular weight. The correct molecular weights and concentrations are marked by X.

ficult to accurately determine the number of solutes present. The correct solutions to the described experiments are extremely sharp, in that they contain a few point parameters from an infinite space of possibilities.

Our contributions described in this paper are using a Genetic Algorithm (GA) [12] for parsimonious regularization and a method of data representation, initialization and mutation for efficient convergence of the GA on this problem. Here, we compare the results obtained from the GA with and without regularization and a method known as 2-Dimensional Spectrum Analysis (2DSA) [3]. A preliminary GA solution for this problem without parsimonious regularization and our advanced initialization resulting in much slower convergence was previously reported [4].

2. METHOD

In AUC sedimentation velocity experiments, we wish to find the parsimonious set of solute parameters that best fit the data. Other methods generally produce sets of solute parameters with much higher cardinality than the actual number of different solutes present. We will demonstrate that this method can determine the actual number of different solutes present in section 4.

Recall the basic procedure for determining solute parameters used in *C(s)* and 2DSA is to establish a set of solute parameters S and apply f. Naturally, we choose sets of solute parameters S as our population individuals and the goodness-of-fit RMSD from f as a building block for our fitness function. The GA population individuals are represented as a string of an even number of floats, containing one s followed by one k value. Each sequential pair of s and k values corresponds to a solute parameter of the set S. Evaluating the fitness begins with an application of f of eq. (2). The fitness is computed as follows:

$$fitness = rmsd * (1 + (rf * nz(\mathbf{x}))^2) \tag{4}$$

Where *rmsd* is the RMSD from eq. (2) and *rf* is the regularization factor. Positive values of *rf* penalize solutions with greater numbers of non-zero parameters in the solution. *rf* can be considered a penalty factor. Typically, we either use *rf*=0 for no regularization or *rf*=.05.

Population initialization is critical to good performance. We used a more naive initialization strategy in a previous study [4], and although we achieved good final results, it was not general enough (it could not solve systems with solutes containing identical s values and differing k values), required some *a priori* knowledge, and took a lot of computation time. On one problem, the previous method took 1,000 CPU hours for a single GA solution. With our new methods, much better solutions are obtained for similar problems in 5 CPU hours, which allows us to compute 100 repetitions of the stochastic GA analysis in 500 CPU hours. We believe better performance could be obtained with further GA tuning analysis for this problem.

The desire for a better initialization strategy for the GA inspired the development of 2DSA which has become a method in its own right. The key to initialization is to determine appropriate buckets for each possible solute parameter. These buckets limit the range of s and k for initialization and mutation. This requires each solute parameter of the population individual to be positionally associated with a bucket (see Figure 5). Since the number of buckets is predetermined (as will be shortly explained), our choice is to fix the length of each population individual to the number of buckets and positionally determine the association between the individual's solute parameters and the buckets. To determine the values for the buckets (see Figures 4, 6), we use the results from 2DSA to automatically compute buckets as follows: Each solute parameter in the 2DSA solution becomes the center of a bucket with a fixed range in s and k. To insure no overlap occurs between buckets, overlapping buckets are shrunk and new buckets (containing no solute parameter of the 2DSA solution) are added to fill the space of the originally determined bucket. The k range of a bucket may be clipped at 1 since this is a physical limit.

The initialization strategy of our previous study partitioned only the s range, while the k value of each solute was allowed to float over the entire k range. The absence of a constraint on k in our previous study caused its slow convergence.

We use two methods to help maintain population diversity: removal of duplicate population members and limiting floating point resolution. Early in the generational loop after the population is sorted by fitness, we perform a pass to remove any duplicate population members. If population members have identical fitness, we must further check to see if they contain the same set of solute parameters. To facilitate the comparison of solute parameters between individuals, we disallow bucket overlap. Eliminating bucket overlap simplifies the comparison by allowing a simple positional equality test of s and k values. We also fix the floating point resolution of s and k to a predetermined number of significant digits, typically 3 or 4. We believe it may be beneficial to make the floating point resolution an increasing function of the generation number. Combining the removal of duplicates and limiting resolution significantly helps maintain diversity.

One point crossover is used exclusively, primarily because our buckets are positionally determined. The break point for crossover is at solute parameter boundaries to assure pairing of s and k values. Intuitively, if we have one parameter pair contributing well in the first half of one individual and one parameter pair contributing well in the second half of another then we could get a better fit solution via one point crossover.

During the generational loop, when an individual is probabilistically selected for mutation, exactly one of the solute parameters is selected. Then, either the s, k or both values are selected to mutate. For each value selected to mutate, the procedure begins by computing a number m to add to the selected value. m is decreased depending on the generation using the following equation where g is the generation:

$$gfactor = 6 * \log_2(2 + 2 * g) \tag{5}$$

A random number m is selected from a normal distribution with a mean of zero and a standard deviation equal to the length of the corresponding bucket's s or k range divided by *gfactor*. *gfactor* slowly decreases the range of mutation as the system evolves. The formula for *gfactor* was determined by a visual inspection of its magnitude for g varying from 0 to 100 and is not known to be optimal for this problem. Next, we add m to the selected value. Finally, if the selected value falls outside of the corresponding bucket's range for s or k, we replace it with the bucket's nearest in-range value.

We use standard values for most of the GA parameters (see Table 2). The only parameters we generally adjust are the number of demes based upon the number of processors we wish to run on, the population sizes (determined coarsely as 10 to 30 times the number of buckets), and the regularization factor (zero or .05).

Summarizing our method, we start with experimental data, determine s value constraints with the van Holde-Weischet analysis, perform the 2DSA analysis, use the 2DSA results to build buckets for GA initialization, then perform 100 iterations of the GA. In section 4 we will compare the effect of using GA regularization on the results.

3. IMPLEMENTATION

The GA implementation of our method was written into a genetic programming (GP) package of our own creation written in C and based upon the GP model of John Koza [13]. The software was initially written for a study of GP on various problems. Our use of a GA to contain a GP is not recommended but is nevertheless described. The data structure of a GP population individual is a tree of nodes. For our problem of parsimonious regularization we require individuals consisting of a string of solute parameters. To implement this in GP, we restrict our trees to use a branching factor of 1. We created a node type which contains the solute parameters. Evaluating this node pushes the solute parameters onto a stack. When the population individual's chain of nodes is fully evaluated, then the fitness computation of eq. (2) can proceed using values from the stack for the set S. In this way, we contain GA functionality within the GP.

Features of our software include demes with a bidirectional ring topology for migration. We use MPI for communication in a distributed environment. For the researcher, there is a public web interface available to submit these jobs to a queued environment [10].

We currently run these analyses on several different local clusters running variants of Linux, as well as on resources made available through TeraGrid.

Parameter	Value used
Population size	200
Generations	100
1 pt Crossover %	50
Mutation %	50
Elitism	2
Number of Demes	10
Deme Migration %	3
Selection method	Exponential by fitness

Table 2: The GA parameters used for analysis. These values are fixed for runs with no regularization and with regularization.

Method	No. of Parameters	RMSD
C(s) - no regularization	3*	1.5807e-2
C(s) - regularization	N/A*	1.5949e-2
2DSA	18	4.513e-3
GA *rf*=0	10.62	4.586e-3
GA *rf*=0.5	3.62	4.542e-3

Table 3: Comparing the number of solute parameters and goodness-of-fit returned for *C(s)* with and without regularization, 2DSA, GA without regularization *rf*=0, and GA with regularization *rf*=.05 for simulated experimental data of 3 solutes (Table 1). For the GA runs, the number is the average of 100 GA iterations. * Results produced by `sedfit` for *C(s)* do not provide the number of solute parameters, 3 is by visual inspection of the graphical output.

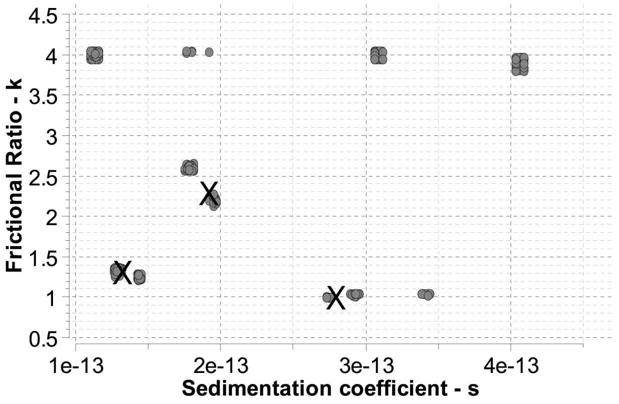

Figure 8: The GA analysis results. The GA was run using the parameters of Table 2, the buckets of Figure 6 and without regularization. These data contain the superposition of 100 iterations of the GA. X marks the target solute parameters from Table 1. Note that the target solute parameters are identified but there are multiple false positives, as in the original 2DSA analysis.

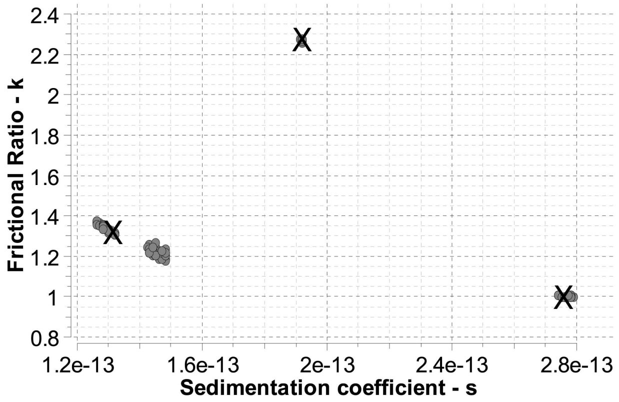

Figure 10: Results of GA with regularization. This run was identical to the run of Figure 8 except regularization was used (rf=.05 in Equation 4). X marks the target solute parameters in the sample. Note since the dots representing each solution are very tight for the middle solute parameter, it might not be obvious that there are 100 superimposed solution solute parameters under that X.

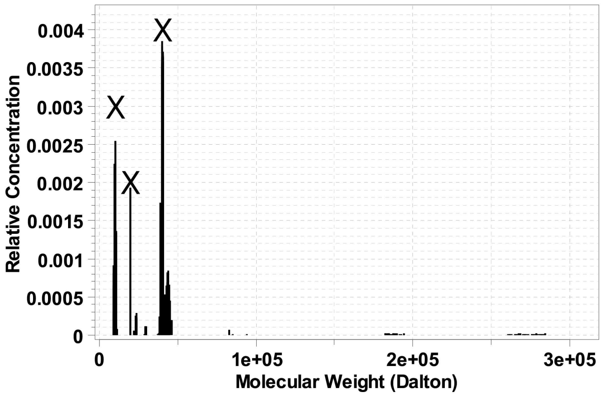

Figure 9: This is another view of the GA analysis without regularization of Figure 8. The vertical axis of relative concentration needs to be multiplied by 100, the number of GA iterations. X marks the target solute molecular weights and concentrations.

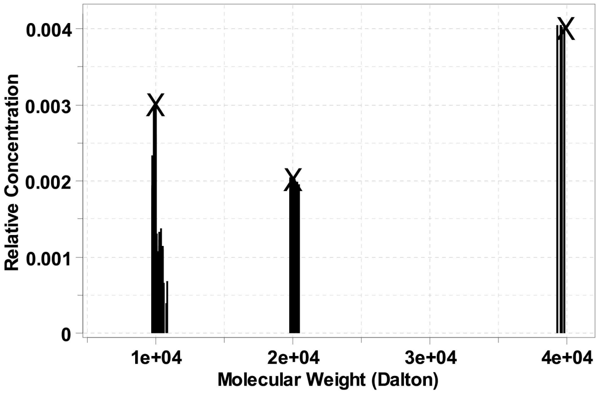

Figure 11: This is another view of the results from the regularized GA analysis of Figure 10. The vertical axis of relative concentration needs to be multiplied by 100, the number of GA iterations. X marks the target solute molecular weight and concentration.

4. RESULTS

The 'No Free Lunch' theorems [19] imply that for any search method a problem can be developed where the method performs worse than random guessing. We wished to determine if the GA was better than random guessing for solving this specific problem. To achieve this, we ran three different tests: random-guessing, mutation-only and mutation-with-crossover. All of the tests were run on the same data sets with identical GA parameters. For our random-guessing test we created a random population of 1,000 individuals with 1 generation (by this we mean simple initialization and no runs through the generational loop). For our mutation-only and mutation-with-crossover tests we created a random population of 100 individuals with 10 generations. For each test we ran 30 trials. It was observed that mutation-with-crossover outperformed each trial of mutation-only which outperformed each trial of random-guessing (data not shown). Therefore, we conclude that the GA is useful for solving this specific problem.

To test the effects of regularization, we simulated a system with three different solutes containing noise of the same quality as what would have been observed in an actual experiment and compared *C(s)*, 2DSA, and GA analyses with and without regularization. Using such data permits us to compare the results to a set of known input parameters, yet we faithfully reproduce a realistic experiment. The simulated experimental data were produced using the UltraScan software [9]. The target solute parameters are shown in Table 1. From the simulated experimental data, the van Holde-Weischet analysis was performed to determine the range of the s values.

Our first analysis was run with *C(s)*. The results are shown in Figure 3. The *C(s)* analysis only reports *s* values and concentrations. The *s* values were missed by *C(s)*, although it did seem to indicate three different solutes present. No correct molecular weights could be computed from these data. Adding regularization to the *C(s)* analysis did not seem to improve the quality of this result and made it less clear how many solutes were present. From the figure, one can see the effect of regularization where the sharp peaks are penalized.

Next a 2DSA analysis was run on these data and a solution containing 18 solute parameters was returned. The results are shown in Figures 4 and 7. 2DSA did manage to accurately locate the three target solute parameters, but also reported multiple false positives. The molecular weights were correctly reported. From this information it does appear there are three solute molecular weights present, from which one could infer that three solutes are present in the data. The relative concentrations of the three solutes are incorrect.

The results from the 2DSA analysis were used to generate buckets for GA initialization as shown in Figure 6. The 18 solute parameters of 2DSA resulted in 21 buckets due to the processing involved in eliminating overlapping buckets. Using these buckets, we ran 100 iterations of GA analysis without regularization using the parameters of Table 2. The results of this analysis are shown in Figures 8 and 9. This method returned an average of 10.62 solute parameters per iteration. The resulting number of solute parameters is much greater than the target 3 present in the simulated data, but less than the 18 returned by the 2DSA analysis and also less than the 21 buckets used for initialization. The GA without regularization, similar to the 2DSA analysis, identified the target solute parameters, but still included multiple false positives. The GA without regularization did, however, do a much better job at estimating the relative concentrations of the solutes than the 2DSA alone.

An identical 100 iterations of GA was run with regularization and the results are shown in Figures 10 and 11. In this case, an average of 3.62 solutes were reported, closely matching the actual three solutes. There is a split in the data for smallest sedimentation coefficient, and it is known that slower sedimenting solutes (with smaller *s* values) are harder to resolve. There is an excellent match for the faster sedimenting solutes. This is in excellent correspondence with our original simulation data of Table 1. The quality of the results is clearly much better than any other method. The false positives are eliminated and the molecular weights and concentrations are correctly identified. This is summarized in Table 3. GA without regularization provided a more parsimonious solution than 2DSA. GA with regularization found by far the most parsimonious solution.

An important issue is the quality of the final results obtained in terms of goodness-of-fit as measured by RMSD. These values were computed for each analysis type and are summarized in Table 3. The RMSD was comparable in all methods, except the *C(s)* method which failed to adequately describe the system, and closely matched the known level of noise.

Visual inspection of the residual vectors indicated no systematic error in the solutions provided by any of the methods, except again the *C(s)* method, which failed to account for the heterogeneity in *k*. GA with regularization does not suffer from the number of false positives reported by the other methods, and was the only method to give precise concentrations for the molecular weights observed.

Our results clearly show the significant benefit of using the GA for parsimonious regularization. This work is being used by researchers worldwide in the analysis of AUC experiments and implemented by the authors in [9].

5. FUTURE

The success of our method brings up some interesting questions for further research. Determining the ideal size of buckets is of importance. Larger buckets increase the search space and subsequently slow convergence. Whereas, making the buckets too small can potentially miss optimal solute parameters. Other shapes of buckets (elliptical) could be examined. We have used a constant regularization factor of .05. The sensitivity of the algorithm to the regularization factor has yet to be thoroughly researched. It is possible that the regularization factor can be eliminated from the algorithm by replacing the fitness function, eq. (4), with the Akaike Information Criteria [1]. We plan to examine a much larger range of simulated and experimental systems over the next year to determine the resolving ability of the algorithm.

6. CONCLUSION

We have presented a new method using a GA with regularization for parsimonious solutions in AUC. The method correctly indicates the number of solute parameters present in the data with identical fitness to much less parsimonious solutions. False positives are removed and the information in the final result closely matches the original model used to produce the experimental data. This is of major importance to the researcher, as this is the first known method to do so and can subsequently give the most accurate molecular weight and shape determinations.

We believe this method may have application to other parameter estimation or inverse problems, such as astronomical image reconstruction, where non-negatively constrained least squares methods are used.

7. ACKNOWLEDGMENTS

We would like to thank the reviewers of this paper for their helpful suggestions and Jeremy Mann for assistance with the BCF Linux cluster.

This research has been supported by NSF Grant DBI-9974819, NIH-RRR022200 and the San Antonio Life Science Institute with Grant #10001642, all to B.D.

8. REFERENCES

[1] H. Akaike. A new look at the statistical model identification. In *IEEE Transactions on Automatic Control*, volume 19, pages 716–723. IEEE, 1974.

[2] R. C. Aster, B. Borchers, and C. H. Thurber. *Parameter Estimation and Inverse Problems*. Elsevier Academic Press, London, 2005.

[3] E. H. Brookes, R. V. Boppana, and B. Demeler. Computing large sparse multivariate optimization problems with an application in biophysics. In *SuperComputing 2006 Conference Proceedings*. ACM, IEEE, November 2006.

[4] E. H. Brookes and B. Demeler. Genetic algorithm optimization for obtaining accurate molecular weight distributions for sedimentation velocity experiments. In *Analytical Ultracentrifugation VIII, Progr. Colloid Polym Sci. 131*, pages 78–82. Springer-Verlag, 2006.

[5] P. H. Brown and P. Schuck. Macromolecular size-and-shape distributions by sedimentation velocity analytical ultracentrifugation. *Biophysical Journal*, 90:4651–4661, June 2006.

[6] W. Cao and B. Demeler. Modeling analytical ultracentrifugation experiments with an adaptive space-time fine element solution of the lamm equation. In *Biophys J.*, volume 83, pages 1589–1602, 2005.

[7] J. L. Cole and J. C. Hansen. Analytical ultracentrifugation as a contemporary biomolecular research tool. In *J. Biomolecular Techniques*, volume 10, pages 163–174, 1999.

[8] B. Demeler. Hydrodynamic Methods. In *Bioinformatics Basics: Applications in Biological Science and Medicine. 2nd Edition*, pages 226–255. CRC Press LLC, 2005.

[9] B. Demeler. UltraScan: A Comprehensive Data Analysis Software Package for Analytical Ultracentrifugation Experiments. In *Modern Analytical Ultracentrifugation: Techniques and Methods*, pages 210–229. Royal Society of Chemistry, UK, 2005.

[10] B. Demeler et al. Center for analytical ultracentrifugation of macromolecular assemblies. http://bcf.uthscsa.edu/cauma.

[11] B. Demeler and K. E. van Holde. Sedimentation velocity analysis of highly heterogeneous systems. In *Anal. Biochem.*, volume 335, pages 279–288, 2004.

[12] J. H. Holland. *Adaptation in Natural and Artificial Systems, 2nd Edition*. MIT Press, Cambridge, MA, 1992.

[13] J. R. Koza. *Genetic Programming*. MIT Press, Cambridge, MA, 1992.

[14] O. Lamm. Die differentialgleichung der ultrazentrifugierung. In *Ark. Mat. Astrol. Fys.*, volume 21B, pages 1–4, 1929.

[15] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Prentice Hall, New Jersey, 1974.

[16] P. Schuck. Size-distribution anal. of macromolecules by sedimentation velocity ultracentrifugation and lamm equation modeling. In *Biophys. J.*, volume 78, pages 1609–1619, 2000.

[17] P. Schuck and B. Demeler. Direct sedimentation analysis of interference optical data in analytical ultracentrifugation. In *Biophys. J.*, volume 76, pages 2288–2296, 1999.

[18] K. van Holde. *Physical Biochemistry, 2nd Edition*. Prentice Hall, New Jersey, 1985.

[19] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. In *IEEE Transactions on Evolutionary Computation*, volume 1, pages 67–82. IEEE, 1997.