Semantics and Accuracy of Gene Expression Threshold Computations

A Case Study

Jaime Seguel
Electrical and Computer Engineering Department
University of Puerto Rico at Mayaguez
Mayaguez, Puerto Rico
e-mail: jaime.seguel@upr.edu

Marie Lluberes
Doctoral Program in CISE
University of Puerto Rico at Mayaguez
Mayaguez, Puerto Rico
e-mail: marie.lluberes@upr.edu

Abstract— **The precise inner workings of cellular mechanisms remain largely unknown and, therefore, their modeling is usually based on conjectures. The availability of large amounts of genetic data, and the lack of abstract mathematical models, makes computer algorithms the only tool available for searching for these hypothetical realities. We call the conjectured algorithmic-independent reality that underlies the method design and intention, the semantics of the algorithm. This article is a brief semantics analysis exercise performed with four binary quantization algorithms for time series of gene expression data.**

Keywords- binary quantization; gene expression; quantitative semantics

I. INTRODUCTION

The post genomic era has brought a myriad of computer methods for modeling cellular mechanisms [1]. As the precise inner workings of these mechanisms is still unknown, several of these methods are based on an *implicit model* of the phenomenon under scrutiny, and its hypothetical manifestations in the data. This is a departure from standard scientific computing practices where algorithms are designed on the basis of well-defined mathematical models, which capture the essence of the phenomenon in abstract terms. Explicit mathematical models also guide the design of numerical simulations, and are often used to test their accuracy. The lack of an algorithmic-independent explicit model obscures the essence of the phenomenon simulated by the computer algorithm, as well as the interpretation of its results. Furthermore, taking the implicit model for granted may determine how the phenomenon is perceived and interpreted in further analyses or modeling endeavors [2].

In the absence of a standard term, we call *semantics of a computer method* the algorithmic-independent meaning of the problem that the method is designed to solve. In particular, we study the semantics of four binary quantization algorithms designed to separate expressed from non-expressed gene states, in a time series of gene expression measurements. Underlying the selected methods is the implicit model of a numeric value, or *threshold,* which separates the state expressions of the gene, in the time interval of the series. Such separation is called *binary quantization*, and the algorithms for splitting the time series data points in expressed and non-expressed states, *binary quantization algorithms*.

Two different ways of computing the threshold are manifested in the four methods. For two of the methods, the threshold is in fact, an explicit numerical value computed before the classification of the data. The other two methods do not compute a threshold, explicitly. Instead, they use statistics to separate the data into expressed and non-expressed states. The threshold is thus, a consequence of the classification of the data points. The semantic question that arises here is what is the algorithmic-independent nature of the threshold. We assume as a working hypothesis that the threshold is a numerical value and attempt to unveil its independent nature through computational experiments performed with the four binary quantization algorithms. The experiments assess the degree of consistency between the computed results and the effects of threshold variations in the simulation of gene regulatory networks (GRN) [3]. The ultimate purpose of a binary quantization is the construction of a probabilistic Boolean network representation (PBN) of a GRN [4]. A PBN is normally derived from prior knowledge of gene interconnections and statistical analyses performed on an array representation of the quantized gene expressions, usually called *binary expression matrix*. Most binary quantization methods are validated on the basis of the quality of the PBN representations derived from them. In order to eliminate the influence of the prior knowledge embedded in the PBN representation, we measure the variations in the binary expression matrices themselves.

The rest of this article is organized as follows: Section II discusses the concept of threshold. Section III is a brief summary of the binary quantization algorithms under consideration. Section IV describes the experiments conducted; and Section V analyses their results. Finally, Section VI summarizes some conclusions of the study.

II. NATURAL THRESHOLD, CONVERGENCE THRESHOLD AND COMPUTATION

Gene expressions are the result of a cascade of processes that are stochastic in nature. However, by the Law of Large Numbers, a smooth non-negative real valued function can approximate the average expression behavior of a significantly large number of cells, in an interval of time [5]. Provided that the variations in this function are large enough,

 ISBN: 978-1-61208-290-5

the lowest and highest values can be associated with expressed and non-expressed gene states, respectively. Hypothetically, someplace within the function range, there should be the point in which Nature separates the two states. We call this hypothetical point *Natural Threshold* (NT). We plan to answer the semantics question by investigating to what extent the result of the methods reveal a NT.

In all the methods considered, the threshold varies with the number of input data points. In order to bound these variations, we compute the *Convergence Threshold* (CT). This threshold is obtained by iterative refinements of the method's thresholds, up until the values fall within a predetermined error tolerance. The data for the iterative refinements is taken from a cubic spline interpolation of the input time series.

The CT is also used to assess the accuracy of the methods. We do this by comparing the method's CT with the threshold of the original time series. We also compare the binary expression matrices derived from these thresholds.

III. THRESHOLD COMPUTATION METHODS

We classify the four methods considered in this study [6], [7], [8], and [9], according to their approach to the search for a threshold. This renders what we call *jump-based* and *cluster-based* methods.

A. Jump-based Methods

It is a frequent practice to use data variations —or *jumps*—, between data points, as a reference for the determination of a threshold value. This approach is taken in [6] and [7]. We refer to these methods as *Algorithm* 1 and *Algorithm* 2, respectively. *Algorithm* 1 computes first the average jump of a sorted version of the input data set, and sets the smallest point in the sorted data that exceeds this value, as the threshold. Fig. 1 shows the pseudo code of *Algorithm* 1.

Algorithm 1. Binarize: INPUT G_i, OUTPUT B_i
$S_i \leftarrow$ sort($G_{i,1}, \ldots, G_{i,k}$)
for j=1 to k-1 **do**
    $D_{i,j} \leftarrow (S_{i,j+1} - S_{i,j})$
endfor
    $t \leftarrow (S_{i,k} - S_{i,1})/(k-1)$
    m = min{j: $D_{i,j} > t$}
for j=1 to k **do**
    if $G_{i,j} \geq S_{i,m+1}$ **then**
        $B_{i,j} \leftarrow 1$
    else
        $B_{i,j} \leftarrow 0$
    endif
endfor

Figure 1. Algorithm 1. Binarize.

In this description, each $G_i = (G_{i,1}, \ldots, G_{i,k})$ is the k-point time series expression of the *i-th* gene, in a gene set. Thus, variable i is fixed in the routine, but runs in the main program. The main program, in turn, calls *Algorithm* 1 and receives its output B_{ij}; which is the *i-th* row in the binary expression matrix. For our purposes, however, the output is $S_{i,m+1}$, as this is the value that separates expressed and non-expressed states.

Algorithm 2, in turn, uses a multi-scale approach for detecting different jumps at different resolution levels. The method scores the jumps before deciding which one is the threshold. An a-posteriori analysis assesses the reliability of this choice. The algorithm consists of several processes. In general, the method finds step functions with different number of steps, which are also the best approximations to the sorted version of the input data. At each approach, the point where the highest jump occurs is identified for further analysis. A relation between the highest jump and the approximation error incurred by the step function is then computed. A high value for this ratio indicates a strong discontinuity in the sorted data, and therefore, a potential threshold candidate. Step functions are computed with a dynamic programming algorithm that returns a sequence of step functions of minimal Euclidian distance to the original data. With each step function approximation, a cost and break point index is calculated and stored. The cost of a step of the function is the distance to the mean of the approached data segment. The cost of the step function, in turn, is the sum of the costs of its steps. Both, the costs and break point indices are computed using the algorithm whose pseudo code is presented below. As in *Algorithm* 1, the first step is sorting the points in the time series. Fig. 2 shows the pseudo code of *Algorithm* 2.

Algorithm 2. Calculation of optimal step functions
Initialization:
$C_i(0) = c_{iN}$, $i = 1, \ldots, N$
Iteration:
for j = 1 to N -2 **do**
    for i =1 to N - j **do**
        $C_i(j) \leftarrow \min_{d=i \ldots N-j}(c_{id} + C_{d+1}(j-1))$
        $Ind_i(j) \leftarrow \text{argmin}_{d=i \ldots N-j}(c_{id} + C_{d+1}(j-1))$
    endfor
endfor

Figure 2. Algorithm 2. Optimal step functions.

Algorithm 3, shown in Fig. 3, reconstructs the break points from the array *Ind,* computed with the *Algorithm* 2.

Algorithm 3. Compute the break points of all optimal step functions
for j = 1 to N - 2 **do**
    $z = j$
    $P_1(j) = Ind_1(z)$
    if j > 1 **then**
        $z \leftarrow z$ - 1
        for i = 2 to j **do**
            $P_i(j) \leftarrow Ind_{P_i - 1(j)+1}(z)$
            $z \leftarrow z - 1$
        endfor
    endif
endfor

Figure 3. Algorithm 3. Break points of optimal step functions.

Break points are used to compute the jump size h. The error of approximation e is the Euclidean distance of the step

function to the sorted input data set. The maximum of the radios $q = h/e$, determines the strongest discontinuity.

We refer the reader to [7] for further details on this method.

B. Cluster-based Methods

Cluster-based methods partition the input data set into a predetermined number k of disjoint data subsets, referred as *clusters*. The partition is made on the basis of the nearest center of each cluster. Here, we compare the well-known k-means method, for k =2, and *Algorithm* 4, which is a variant, proposed by the authors, that replaces the mean with the median. We provide no pseudo code for *2-means*, but recall that, in this method, data centers are initialized randomly. In contrast, in *Algorithm* 4, no random initialization is necessary, as shown in its pseudo code in Fig. 4.

Algorithm 4. Median separation. INPUT G_i, OUTPUT T
$S \leftarrow$ sort($G_{i,1}, \ldots, G_{i,m}$)
for j = 1 to m - 1 **do**
    $lm_j \leftarrow$ median ($S_1, \ldots, S_j$)
    $um_j \leftarrow$ median($S_{j+1}, \ldots, S_m$)
    $A_j \mid \leftarrow um_j - lm_j \mid$
endfor
$Ind \leftarrow \text{argmax}_{d=1\ldots m}(A)$
$lmMax \leftarrow lm_{Ind}$
$umMax \leftarrow um_{Ind}$
$T \leftarrow (umMax + lmMax) / 2$

Figure 4. Algorithm 4. Median Separation.

Algorithm 4 sorts first the m input data points, and for each value j, $1 \leq j \leq m$; computes the median of the first j points and that of the last m – j – 1 data points. Then, it finds the pair of medians that are farther apart and returns their average as the threshold. In both, *Algorithm* 4 and *2-means*, two points, an upper and a lower value, determine the binary quantization. Therefore, if there is a threshold, this is most probably given by their average.

IV. METHODS AND EXPERIMENTS

We designed two experiments. The first uses the threshold computed by each method. The second uses the CT of each method, instead. Both experiments were performed with two different data sets. The first data set [10], which corresponds to the mitotic cell cycle of yeast, is taken from [11]. We took the 17-point time series of four genes, namely cdc24, cdc19, cdc15, and cdc27. The second data set is a 6 x 8 randomly generated matrix of real values between 0 and 1, mimicking an eight point time series of six genes.

All experiments were run in Matlab 7.0.12.635 (R2011a) for Mac.

V. ANALYSIS OF RESULTS

Next is a brief analysis of the experimental results.

A. Numerical Variations of the Thresholds

Fig. 5 shows the thresholds obtained in the first experiment. Algorithms *2* and *2-means* exhibit the closest numerical values, while the thresholds returned by *Algorithm* 1 and *Algorithm* 4 are significantly farther apart. All threshold values are shown in Tables 2 and 3. In order to quantify these observations, we compute the ratio $d_{max}/range$, where d_{max} is the largest distance between thresholds for a given time series, and *range* is the difference between the largest and smallest values in the time series.

The results of these computations are depicted in Fig. 6.

For a more algorithmic-centered classification, we define the *distance between methods* as the Euclidian distance between the vectors formed by the thresholds of each time series of genes. Table 1 shows the distances between each pair of methods.

The shortest and largest distances are highlighted, as well. In both experiments and with the cdc data, the distance from *Algorithm* 1 to *2-means* is the largest. In turn, *2-means*

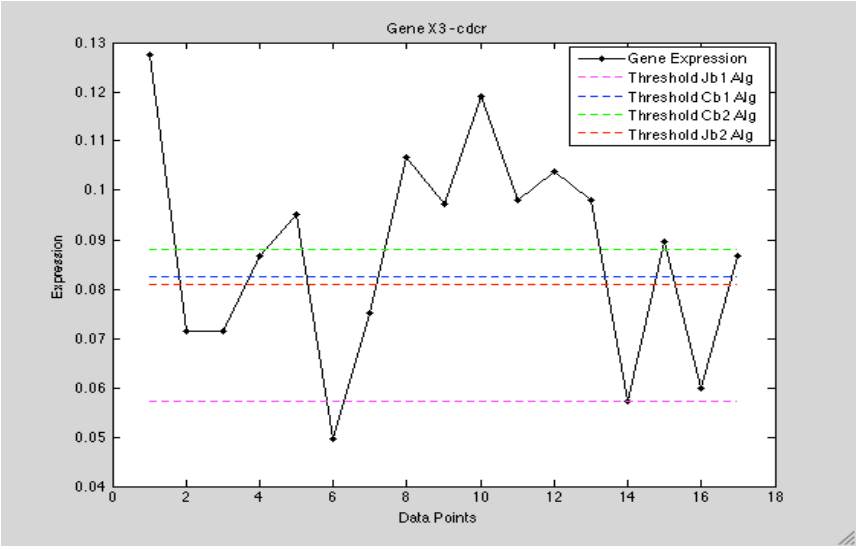

Figure 5. Thresholds plots of the four methods on Experiment 1. Here, Jb1 is Algorithm 1, Jb2 is Algorithm 2, Cb1 is 2-means and Cb2, Algorithm 4.

 ISBN: 978-1-61208-290-5

TABLE I. DISTANCES BETWEEN ALL METHODS EXPRESSED AS HAMMING DISTANCE. LOWEST AND HIGHEST SCORES ARE HIGHLIGHTED.

Hamming Distance							
	Original				*Convergence*		
	Jb2	***Cb1***	***Cb2***		***Jb2***	***Cb1***	***Cb2***
			cdc data				
Jb1	0.23529	0.25	0.32353	***Jb1***	0.45588	0.47059	0.13235
Jb2	—	0.014706	0.088235	***Jb2***	—	0.014706	0.38235
Cb1	—	—	0.073529	***Cb1***	—	—	0.39706
			Random data				
Jb1	0.3125	0.22917	0.3125	***Jb1***	0.4375	0.41667	0.10417
Jb2	—	0.083333	0	***Jb2***	—	0.020833	0.33333
Cb1	—	—	0.083333	***Cb1***	—	—	0.3125

and *Algorithm* 4 are separated by the shortest distance, followed closely by algorithms 1 and 4. We also compared the variations of each algorithm with respect to the input data sets. The results are reported in Table 4. Table 5 shows the Euclidean distance between the two experiments, for each of the four methods. Among the methods, *Algorithm* 1 turned out to be the less stable as it has both the shortest and largest distances between data sets.

B. *Variations in the Binary Quantization*

The Hamming distances divided by the number of data points measure the differences between the binary quantizations derived with each threshold on the same time series. Fig. 7 shows the results for the cdc data set. The highlighted rows are pairs of different methods whose binary quantizations coincide. The methods with the largest number of coincidences are *Algorithm* 2 and *2-means*.

Table 1 shows the Hamming distances between the binary quantization computed with the convergence threshold of each method. The closest methods are again, *Algorithm* 2 and *2-means*. In turn, the distances between algorithms 1 and 4, and algorithms 1 and 2 are the largest.

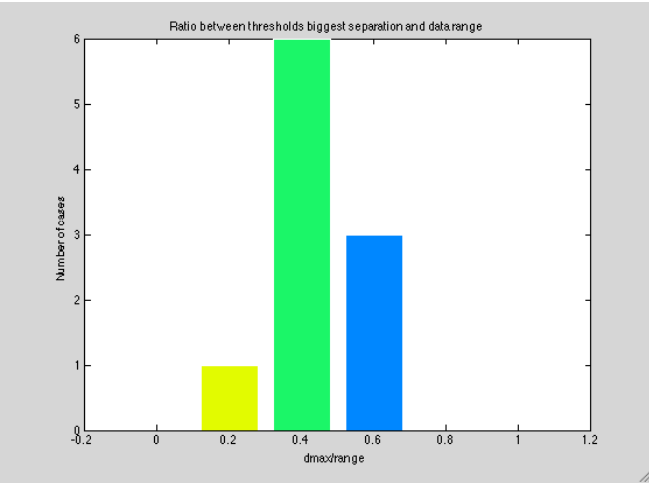

Figure 6. Relations between maximal distance between threshold and data range.

VI. CONCLUSIONS

The results show that the thresholds computed by the four methods are significantly different. This is a clear rejection of the hypothesis that the methods compute the algorithmic-independent value, referred in this article as Natural Threshold. As expected, convergence threshold differs from thresholds. This sensitivity to the sample size is also a negative answer to the question of the accuracy of the methods. Also, the convergence thresholds produced binary expression matrices that are also significantly different to the ones obtained by the thresholds of each method. An important implication that can be drawn from these observations is that the models of gene regulatory networks, whose construction uses a binary quantization as a first step, are biased by the choice of the binary quantization method. The success of some PBN representations of GRNs suggests that this bias is being corrected, in part, with the incorporation of prior gene interconnection knowledge, and expected results.

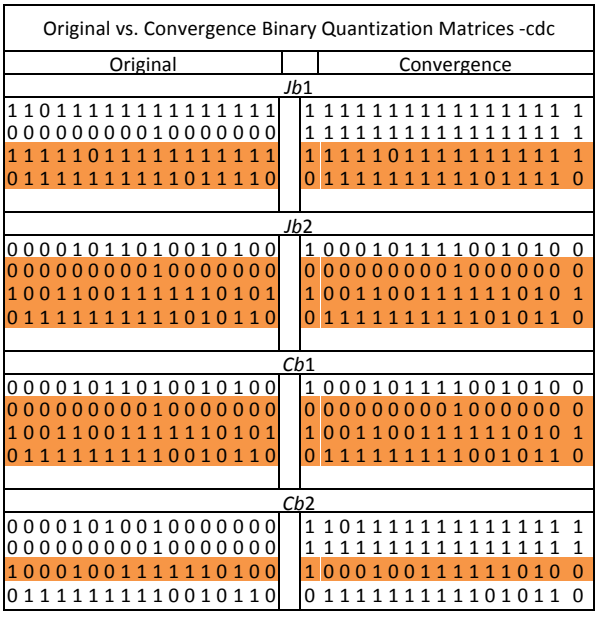

Figure 7. Comparison between binary quantization matrices between same methods using thresholds obtained from both experiments. Matching binarizations are highlighted. Cdc data set.

TABLE II. THRESHOLD VALUES FOR RANDOM GENERATED DATA. HIGHLIGHTED VALUES ARE MATCHING VALUES ON BOTH EXPERIMENTS.

Experiment 1				Experiment 2			
Jb1	*Jb2*	*Cb1*	*Cb2*	*Jb1*	*Jb2*	*Cb1*	*Cb2*
0.28072	0.54384	0.50574	0.45214	0.28072	0.48069	0.4671	0.18751
0.2805	0.4284	0.45131	0.41986	0.2805	0.37147	0.40742	0.23781
0.34758	0.45847	0.49741	0.44159	0.34758	0.77806	0.70254	0.37476
0.40338	0.50213	0.51927	0.47716	0.32267	0.50211	0.51097	0.47145
0.51041	0.71765	0.66833	0.64514	0.34708	0.67214	0.64782	0.32805
0.13834	0.60205	0.30395	0.61079	0.13834	0.50511	0.52622	0.15539

TABLE III. THRESHOLD VALUES FOR CDC DATA. HIGHLIGHTED VALUES ARE MATCHING VALUES ON BOTH EXPERIMENTS.

Experiment 1				Experiment 2			
Jb1	*Jb2*	*Cb1*	*Cb2*	*Jb1*	*Jb2*	*Cb1*	*Cb2*
0.1981	0.27476	0.27286	0.30952	0.18095	0.25524	0.25531	0.19169
5.6067	4.1914	3.7892	3.8212	0.85382	3.2098	3.3162	0.89969
0.057143	0.080952	0.082453	0.087976	0.057143	0.080952	0.082453	0.087976
0.11143	0.1181	0.12625	0.125	0.11143	0.11976	0.12571	0.1196

TABLE IV. EUCLIDEAN DISTANCE BETWEEN DIFFERENT METHODS ON SAME EXPERIMENT.

		Jb1—Jb2	*Jb2—Cb1*	*Cb1—Cb2*	*Jb1—Cb1*	*Jb2—Cb2*	*Jb1—Cb2*
Exp. 1	***cdc***	1.41759	0.40229	0.04899	1.81927	0.37196	1.78929
	random	0.60920	0.30835	0.32162	0.40995	0.12134	0.55177
Exp. 2	***cdc***	2.35729	0.10658	2.41736	2.46367	2.31099	0.05689
	random	0.71131	0.09108	0.67506	0.67320	0.71282	0.18449

TABLE V. EUCLIDEAN DISTANCE BETWEEN SAME METHODS ON DIFFERENT EXPERIMENT

		Jb1—Jb1	*Jb2—Jb2*	*Cb1—Cb1*	*Cb2—Cb2*
Exp1 vs Exp2	***cdc***	4.75291	0.98180	0.47333	2.92389
	random	0.18218	0.34761	0.30885	0.64467

The difficulties in determining a numerical threshold may arise from the intrinsic nature of gene expressions. Both assumptions, jumps in the data or statistical separation in two groups may be too strict, in some sense, as data may have some natural perturbation or noise. It may be the case that on average, expressed and not expressed gene states are separated in nature by an interval, not a point. In the interval model, expressed states will correspond to values above the interval's upper limit while non-expressed states, to values below its lower limit. And these expression values that fall within the interval shall be declared *noisy data-points*. Threshold intervals in gene expression time series may be investigated by adding filters that eliminate expression values that are too close to the threshold points returned by the previous methods.

Threshold computation is not a large-scale problem, at least not with the amount of data compilation currently available. However, this may change as models evolve and parameters, such as time, are incorporated. In such cases, parallel and distributed computing versions of the algorithms will be a necessary algorithmic development. Most probably, because of the strong interdependence of data expression, these methods will be mostly implemented in shared memory systems.

This paper suggests a line of research that may be worth pursuing. Its ultimate aim should be a mathematical framework for validating implicit models from their different algorithmic approaches. This validation might eventually lead to, or replace an explicit abstract mathematical representation of the reality behind by the implicit model. The development of such a framework will support current tendencies of using multi-algorithmic approaches to data based computational modeling.

ACKNOWLEDGMENT

This research was supported in part by grants NIH-MARC 5T36GM095335-02 and NIH-R25GM088023 from the National Institute of General Medical Sciences.

REFERENCES

[1] H. Kitano, "Systems Biology: a brief overview," vol. 295, no. 5560, Science, March 2002, pp. 1662-1664.

[2] Y. Kim, S. Han, S. Choi, and D. Hwang, "Inference of dynamic networks using time-course data," Briefings in Bioinformatics, May 2013, doi:10.1093/bib/bbt028.

[3] C. Needham, I. Manfield, A. Bulpitt, P. Gilmartin, and D. Westhead, "From gene expression to gene regulatory networks in Arabioposis thaliana," BMC Systems Biology, 3:85, Sept. 2009, doi:10.1186/1752-0509-3-85.

[4] I. Shmulevich and E. Dougherty, "Probabilisitic Boolean networks: the modeling and control of gene regulatory networks," SIAM, 2010.

[5] L. B. Klebanov and A.Y. Yakovlev, "A nitty-gritty aspect of correlation and network inference from gene expression data," Biology Direct, vol.3, Aug. 2008.

[6] I. Shmulevich and W. Zhang, "Binary analysis and optimization-based normalization of gene expression data," Bioinformatics, vol. 18, no. 4, April 2002, pp. 555–565.

[7] M. Hopfensitz, et al., "Multiscale Binarization of Gene Expression Data for Reconstructing Boolean Networks," IEEE/ACM transactions on computational biology and bioinformatics, vol. 9, no. 2, March 2011, pp. 487–498.

[8] X. Zhou, X. Wang, and E. R. Dougherty, "Binarization of microarray data on the basis of a mixture model," Molecular cancer therapeutics, vol. 2, no. 7, July 2003, pp. 679–84.

[9] K. Hakamada, T. Hanai, H. Honda, and T. Kobayashi, "A preprocessing method for inferring genetic interaction from gene expression data using Boolean algorithm," Journal of bioscience and bioengineering, vol. 98, no. 6, Jan. 2004, pp. 457–63.

[10] R. J. Cho, et al., "A genome-wide transcriptional analysis of the mitotic cell cycle," Molecular Cell, vol. 2, 1998, pp. 65–73.

[11] http//:arep.med.hsrvsrd.edu/ExpressDB/EDS16/EDS16data.txt.

 ISBN: 978-1-61208-290-5