1998

Techniques for Resolving Incomplete Systems in K-Systems Analysis.

Gary J. Asmus
Louisiana State University and Agricultural & Mechanical College

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

TECHNIQUES FOR RESOLVING INCOMPLETE SYSTEMS IN K-SYSTEMS ANALYSIS

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
the Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by
Gary J. Asmus
B.S., Louisiana State University, 1992
December 1998

Acknowledgments

I would like to thank all of the committee members for their advice and support. In particular, I would like to thank Dr. S. Sitharama Iyengar for his guidance of the department and his personal support and guidance of my work. Also, Dr. Robert Mathews was integral in helping me to look beyond the bounds of what is known and teaching me how to effectively communicate my thoughts. Most importantly, Dr. J. Bush Jones provided me with leadership and insight about all aspects of academic endeavors and investigations. It is not an exaggeration to say that without his unwavering support of my work, I would not have been able to complete this journey.

Finally, I must thank my parents for their patience and encouragement throughout my life. They taught me that any task, no matter how difficult, can be completed if I were to simply begin and persevere. Also, I must acknowledge the help and support of my friend and peer, Chris Branton, who has traveled this academic road with me from its beginning. The single most important person in my life, who has been a constant joy and perfect traveling companion throughout this journey, is my beautiful wife, Tammy. Without her understanding and love, I would not have been able to complete this work. She gives meaning to all that I do.

Table Of Contents

List Of Figures

Abstract

K-systems analysis is a generalization of reconstructability analysis (RA), where any general, complete multivariate system (g-system) can be transformed into an isomorphic, dimensionless system (a K-system) that has sufficient properties to be analyzed using probabilistic RA algorithms. In particular, a g-system consists of a set of states formed from a complete combination of the variables assigned specific values from a finite set of possible values and an associated system function value. The g-system must be complete in that all possible states must have an associated system function value. K-systems analysis has been applied to a variety of systems, but many real-world systems consist of data that is incomplete.

Impediments in real-world systems have been previously identified as state contradictions, data scattering and missing data [JONE 85d]. The problem of state contradictions has been adequately addressed, but while techniques for the resolution of data scattering and missing data have been proposed, additional issues remain. The author has condensed the understanding of data scattering and missing data into the single problem of an incomplete system. Within this context, techniques for resolving incomplete systems and, thereby, inducing a complete system have been developed.

If a g-system is incomplete, it may be viewed solely from the perspective of missing data. A new algorithm has been developed based on the state distance and uses this distance to determine unbiased estimates of the values for the system function. The state distance is a generalized Hamming distance and is shown to satisfy

the properties of a metric on the state space and to be superior to current methods for imputing system function values.

An incomplete system may be viewed from the perspective of data scattering. In general, scattered data may be resolved through clustering and, previously, this clustering has been done in one dimension. A method is developed that allows the meaningful use of two dimensions in the clustering. Further, a new pairwise similarity measure is developed based on the maximum entropy principle and mathematics that form the foundation of K-systems analysis. Use of this similarity measure is demonstrated within the context of an existing clustering algorithm.

Chapter 1. Overview of K-systems Analysis

Reconstructability analysis is the study of the relationship of subsystems to systems, the relationship of parts to wholes, the analysis of a system based solely on the information contained within it. The development of reconstructability analysis began in the 1960's with the work of Ross Ashby [HIGA 83]. Further developments occurred throughout the 1970's, but reconstructability analysis did not become fully developed until 1981 when a whole issue of the International Journal of General Systems was devoted to the evaluation of the reconstruction hypothesis. In particular, the work of Cavallo and Klir provided a detailed overview and definition of reconstructability analysis [CAVA 81a-b].

During the mid 1980's, Bush Jones published a series of papers that provided efficient algorithms which addressed the practical needs for performing reconstructability analysis. The topics covered in these papers include the determination of reconstruction families [JONE 82], the determination of unbiased reconstructions [JONE 85a], and a greedy algorithm for the generalization of the reconstruction problem [JONE 85b]. One paper of particular interest here concerned the reconstruction of general functions with arbitrary data [JONE 85c]. Previously, reconstructability analysis applied only to probabilistic or possibilistic systems. Jones was able to develop a method whereby a general system (g-system) could be transformed into an isomorphic Klir system (K-system) that could be analyzed using

the probabilistic reconstructability analysis algorithms. The results of the analysis could then be mapped directly back to the original g-system.

Following this work Jones published a paper on reconstructability considerations with arbitrary data, in which he proposed methods whereby impediments in real world data could be overcome so that reconstructability analysis could be applied [JONES 85d]. The main focus of the research reported here is on these impediments and their resolution. In particular, the focus will be on the problems referred to as data scattering and missing data.

First, we provide a cohesive and condensed understanding of the problems of data scattering and missing data. In effect, these two problems are actually a single problem that may be viewed from two perspectives. The one issue that ties these perspectives together is that both are interpretations of an incomplete system. Both interpretations are valid in particular contexts and discriminating between the two is often difficult. In general, the view of missing data is most general in that it does not assume that the source of the data has any particular attributes; it only assumes the existing data is all of the information that is available. Data scattering assumes that the observation or control of the variable values was imprecise and can be refined based on the existing system information. Additionally, the view of data scattering may be useful if the amount of missing data is large; clustering to resolve scattered data may reduce the amount of missing data in the system.

A new algorithm for imputing missing data is developed based on the distance between states. The state distance that is developed for the Cartesian state space is

shown to be a true distance because it satisfies the symmetry axiom, the identity axiom and the triangle inequality. The state distance is used to identify the set of closest states and this set is used to impute an unbiased estimation of the missing state. It is shown that the closest state set is superior to the existing technique by proving that it shares, on average, more information with the missing state than does the set of all states. The algorithm is defined and analyzed and shown to have a time complexity of $O(n^2)$, where n is the number of states.

If the incompleteness of the system is determined to be due to data scattering, the data will require clustering. Currently, clustering for K-systems analysis is done in one dimension for each variable that is considered to be scattered. The author describes the shortcomings of using clusters that are inherently one dimensional and develops a general methodology for using higher dimensional clusterings. In particular, focus is on the use of two dimensional clusters so that the system may still be submitted for K-systems analysis. While data that is clustered may be easily presented for K-systems analysis, use of the resulting analysis for prediction requires additional considerations. The author presents a method whereby previously unobserved variable values may be projected to the clusters used for the analysis and, subsequently, be used to predict the system response. In addition to predicting the system response, the methodology also enables the numerical qualification of the results independent of the algorithm that was used to produce the clustering.

Next, a technique that is similar to the one used to perform the K-system transformation will be applied in the context of a similarity measure based on the

information entropy. This entropy similarity measure is specifically defined for the pair wise comparison of *n*-dimensional points, but is also generalized to the overall similarity of an arbitrary number of *n*-dimensional points. Using only a K-systems type of transformation and the definition of information entropy, this similarity measure is analyzed in the context of its corresponding dissimilarity measure. It is shown that this dissimilarity measure does not meet all the properties of a metric as did the state distance, specifically, it does not satisfy the triangle inequality. The behavior of the entropy similarity is further explored to demonstrate why it does not satisfy this property.

Finally, the use of entropy similarity is demonstrated using a density based clustering algorithm known as the taxmap algorithm [CARM 69]. This algorithm is intended to simulate the way in which a human observer would detect clusters in two and three dimensions. It makes use of a similarity matrix that contains pair wise measures of similarity for all data points. It proceeds by finding the two most similar points to initiate a cluster and continues to add the next most similar point to that cluster until a measure of discontinuity is exceeded. It then repeats this process until all the points have been assigned to a cluster. The author uses the entropy similarity and derives a corresponding measure of discontinuity for use in the taxmap algorithm. Examples of the results of this algorithm are presented for a variety of data sets.

The remainder of this work will be organized as follows. First, the work of Cavallo and Klir [CAVA 81a-b] and Jones [JONES 85a-d] will be reviewed to form a foundation from which to work. Then the impediments in real world data will be

described along with the existing techniques that are applied for their resolution. Limitations of these techniques will be described along with the motivation for finding new methods. Next, follows a discussion of what exactly is an incomplete system and why we wish to work mainly with complete systems. An algorithm for filling in (or imputing) system function values for missing states will be derived, defined and the computational complexity will be analyzed. Finally, a new similarity measure will be derived, defined and used in a clustering algorithm that may also be used to resolve incomplete systems. Finally, the missing state algorithm and the clustering algorithm will be discussed in the context of their use for resolving incomplete systems.

1.1 Introduction to K-Systems Analysis

1.1.1 General Definitions

We begin with the work of Cavallo and Klir which established the basic terminology and concepts that are known as reconstructability analysis. When doing any type of systems modeling, it can be very useful to represent the overall system as a set of coupled subsystems. As Cavallo and Klir point out, a complex system which is represented by a set of coupled subsystem has a number of advantages.

> "It is usually easier to understand a large system when it can be decomposed into smaller systems. It is also easier to further develop such systems, as each subsystem may be investigated independently of other subsystems. Once sufficiently developed, it is usually easier to utilize it for various purposes. Moreover, the decomposed system may be easier to document or simulate on computer." [CAVA 81a]

Difficulties arise when trying to decompose a system into subsystems; it is possible that the overall system cannot be reconstructed from the set of subsystems.

This lead's to two distinct yet related problems for systems modeling. The first is the reconstructability problem which has been defined as the problem of determining which subsystems of an overall system are adequate for describing the overall system within some desired level of approximation. The identification problem can be viewed as the complement to the reconstructability problem. Given that the overall system is unknown, the problem is one of identifying properties of the overall system based solely on appropriate properties of the known subsystems [CAVA 81a]. The process of solving these two problems is known as reconstructability analysis (RA). Specifically, the term reconstructability analysis will be used when referring to systems that are inherently probabilistic and complete such that they can be directly submitted for analysis using the reconstructability algorithms. Arbitrary multivariate systems may require prior analysis and transformation into a form that is suitable for analysis using the probabilistic RA algorithms.

Reconstructability analysis concerns itself with systems that are composed of states. A system may be viewed as a set of multivariate data that consists of a set of variables and a system function which is defined on these variables. Thus, a system consists of a tuple of the form $< v_1, v_2, \ldots, v_n, f >$ where $v_1, v_2, \ldots, v_n$ are variables and f is a function defined over these variables. The function may be a probabilistic behavior function, a possibilistic behavior function, a fuzzy set membership function, or any arbitrary (and possibly non-linear) function over the set of variables. A state is a complete combination of values from a state set assigned to each variable and the function f is associated with each state.

Formally, a system may be defined as the following six-tuple:

$$B = (V, W, s, A, Q, f) \tag{1.1}$$

where

- $V = \{v_i \mid i \in 1, 2, \ldots, n\}$ is the set of variables
- $W = \{\mathcal{V}_j \mid j \in \{1, 2, \ldots, m\}, m \leq n\}$ is family of state sets
- $s: V \rightarrow W$ is an onto mapping that assigns each variable in V to one of the state sets
- $A = s(\mathcal{V}_1) \times s(\mathcal{V}_2) \times \ldots \times s(\mathcal{V}_n)$ is the set of all states
- Q is the set of real numbers
- $A \rightarrow Q$ is a system function that represents the meaning of the information regarding the total set of states in the system.

Typically, the function f and the system B referred to probabilistic or possibilistic systems as in [CAVA 81], but the use of these symbols here is more general in that they may represent any arbitrary system that can be represented a set of states and a corresponding system function. This is due to the fact that the early RA methodology was expanded to include functions from arbitrary general systems [JONE 85a-e] [TRIV 93].

The algorithms that were developed in [CAVA 81] and [JONE 85a] are targeted for probabilistic functions for use on completely specified systems. An example of a complete, probabilistic system is shown in figure 1.

In this example, $V = \{v_1, v_2, v_3\}$; $W = \{\{0, 1\}, \{0, 1, 2\}\}$; $s: V \rightarrow W$ is the onto mapping that assigns $\{0, 1\}$ to v_1 and v_2, and $\{0, 1, 2\}$ to v_3; $A = \{000, 001, 002, 010,$

011, 012, 100, 101, 102, 110, 111, 112}; $Q = [0, 1]$ is a set of real numbers and $f: A \rightarrow Q$ is a probabilistic function. Note that current conception of a system

v_1	v_2	v_3	$f(\alpha)$
0	0	0	0.1
0	0	1	0.1
0	0	2	0.0
0	1	0	0.2
0	1	1	0.1
0	1	2	0.2
1	0	0	0.1
1	0	1	0.0
1	0	2	0.0
1	1	0	0.1
1	1	1	0.0
1	1	2	0.1

Figure 1: A probabilistic system

for reconstructability analysis is a probabilistic system, but the RA algorithms may be applied to arbitrary systems through application of a transformation of a general system into a Klir-system (K-system). This transformation and its properties will be defined and demonstrated below.

We define some further notation so that we may understand the notion of states and substates. Using notation similar to [CAVA 81b]:

For each state

$\alpha = (\alpha_i | i \in N_n) \in A$, where N_n is the total number of variables

of a system defined by (1.1) and for each state

$\beta = (\beta_j | j \in X, X \subset N_n)$

associated with variables in set

$$Z = \{v_j \mid j \in X, X \subset N_n\} \subset V,$$

let β be called a substate of α (or α be called a superstate of β) if and only if

$$\beta_j = \alpha_j, \text{ for all } j \in X.$$

Let $\beta \prec \alpha$ (and $\alpha \succ \beta$) denote that β is a substate of α. [CAVA 81b]

Let $[f \downarrow Z]$ denote the projection of f which disregards all variables in V except those in set $Z \subset V$. Then, $[f \downarrow Z]$ is a mapping from a set of states (substates of states in A) to Q:

$$[f \downarrow Z]: \underset{v_i \in Z}{\times} s(v_i) \rightarrow Q$$

such that

$$[f \downarrow Z](\beta) = g(\{f(\alpha) \mid \alpha \succ \beta\}),$$

where function g is determined by the nature of function f. For instance,

$$[f \downarrow Z](\beta) = \sum_{\alpha \succ \beta} f(\alpha)$$

when f is a probabilistic function [CAVA 81b].

One final definition is needed before moving onto the notion of an unbiased reconstruction. Any system may also be viewed as a subsystem of an overall system. Given a system, B, as defined above, a collection of q subsystems is defined as,

$$S = \{^kB\} = \{(^kV, {}^kV, {}^ks, {}^kA, {}^kQ, {}^kf) \mid k \in \{1,2,\ldots,q\}\}$$

Elements of the set S are referred to as subsystems of B if and only if, for each k, the elements satisfy the following conditions:

- $^kV \subset V$;

- ${}^{k}\mathcal{V} \sqsubseteq \mathcal{V}$ such that ${}^{k}s$ is onto;
- ${}^{k}s : {}^{k}V \rightarrow {}^{k}\mathcal{V}$ such that ${}^{k}s(v_i) = s(v_i)$ for each $v_i \in {}^{k}V$;
- ${}^{k}A = \underset{v_i \in {}^{k}V}{\times} {}^{k}s(v_i)$
- ${}^{k}Q = Q$;
- ${}^{k}f = [f \downarrow {}^{k}V]$ [CAVA 81b].

We can also use the concise definition of a subsystem using the terminology of Jones. Given an overall system B with a behavior function f and a subsystem ${}^{k}B$, the behavior function ${}^{k}f$ must satisfy the following condition:

$$ {}^{k}f(\beta) = \sum_{\beta \prec \alpha} f(\alpha), $$

where ${}^{k}\beta \in A, \beta \prec \alpha$ (β is a substate of α) [JONE 82].

1.1.3 Unbiased Reconstructions

Based on the definitions provided so far, we can now review the idea of unbiased reconstructions. Given an overall system, there is a family of reconstructions that are compatible with this system. Given a particular reconstruction, there also exists a family of overall systems that are compatible with it [CAVA 81a]. It is desirable to select one member of the family of reconstructions, say f_S, to represent the overall system, f, and this selection requires some assumption which will justify this choice. When the overall system, f, is representing a probability distribution function, the principle of maximum entropy can be invoked to select that function which is maximally non-committal to all matters except the requirements that

$$\left[f_s \downarrow^k V\right] \models^k f = \left[f \downarrow^k V\right] \forall k \in N_q$$ [CAVA 81b].

The use of the principle of maximum entropy for selecting the function from the family of possible reconstructions has been justified by the following arguments :

- The maximum entropy probability distribution is the only unbiased distribution, that is, the only distribution which takes into account all available information about the system, but no additional information [CAVA 81b].
- The maximum entropy probability distribution is the most likely distribution. Given a reconstruction hypothesis, each element of the reconstruction family of that hypothesis could have been generated by any number of actual data sets. The largest number of possible data sets which are mutually comparable and compatible with the given reconstruction hypothesis are those which are also compatible with the maximum entropy overall probability distribution [CAVA 81b].
- Maximizing any function but entropy leads to inconsistencies unless that function has the same maxima as entropy [CAVA 81b].
- Every real world system can be represented by the maximum entropy reconstruction because joining the subsystems that represent the real world system always results in the maximum entropy distribution [CAVA 81b].

In addition to these arguments, Pittarelli has evaluated the use of the maximum entropy principle in detail and finds that while there are qualifications for the arguments provided above, these arguments along with others provide compelling evidence that justifies the use of maximum entropy distributions [PITT 89].

1.2 Reconstructability Analysis Algorithms

Cavallo and Klir provided several algorithms for computing the unbiased reconstruction, but it was not until the mid 1980's that Bush Jones invented a more efficient algorithm that implementation became a reality. Jones also created a greedy algorithm for a generalization of the reconstruction problem.

The first Jones algorithm is for determination of unbiased reconstructions. It makes use of only independent states for determining the reconstruction. Also, the algorithm is general in that it can be employed on arbitrary collections of states and substates. This makes it especially useful for the greedy algorithm for the reconstructions problem. This algorithm was proven to converge and provides the maximum entropy solution to the reconstruction hypothesis[JONE 85a].

Jones also provided a greedy algorithm for the generalization of the reconstruction problem. The generalization of the problem can be stated as follows:

"Given an overall system **B** with known probabilistic behavior function f and hence known behavior functions ${}^{k}f$ for the set of substates $\{\beta\}$, determine a subset of $\{\beta\}$ of given size or whose unbiased reconstruction is within acceptable tolerance of f to represent the system." [JONE 85b]

The algorithm will work on an independent set of substates or on the complete set. While the independent set of substates is guaranteed to avoid using redundant information, the complete set may provide faster or more compact reconstructions.

The algorithm works on the set E, which is the set of all substates that are being using for the reconstruction (either independent states or all states). Let D represent the set of substates currently used for the reconstruction during execution of

the algorithm and let U(D)→ $\hat{f}_{i,j}$ represent the computation of the unbiased reconstruction for the substates of D. Finally, let γ(.) represent the function which is used to select the next substate to be included in the reconstruction; the greater the value of γ(β), the more desirable that state is for inclusion into the reconstruction.

The algorithm starts by initializing $\hat{f}_{i,j}$ to a flat distribution and letting D be initially empty. It selects one β to add to D such that γ(β) is a maximum and computes the unbiased reconstruction U(D) for the new D. If either a size limit of D is exceeded or if the approximation $\hat{f}_{i,j}$ is sufficiently close to the true $f_{i,j}$, then the algorithm stops. Otherwise it continues to add substates to the set D until one of the two criteria are satisfied. Using these two algorithms to form the reconstruction that approximates the overall distribution is what is known as reconstructability analysis. It provides the list of substates, in order of influence, that have the most effect on system behavior.

1.3 Reconstruction of General Functions

With the two preceding algorithms in hand, reconstructability analysis was fully defined and could be effectively applied to any probabilistic behavior function. The results of such an analysis would be correct for the information given and would introduce no extraneous information to the analysis. In order to utilize reconstructability analysis for non-behavior systems, Jones invented a transformation which can be applied to practically any multivariate function on discrete variables. Any general system (or g-system) can be transformed to an isomorphic Klir system (or K-system) which can then be analyzed using reconstructability analysis.

We begin by defining the g-system and will make use of the definitions of states and substates from the previous sections. First, associated with a g-system is a general behavior function $f(\alpha)$. If A is the set of all possible states of a systems and R^+ is the set of positive real numbers, then $f: A \rightarrow R^+$ is the function that represents the information that is associated with the system states. Note that, without loss of generality, we restrict the function values to the positive real numbers, since these values may be easily scaled to R^+. Two additional definitions are necessary before we can define a g-system. There is a set of functions defined for each subsystem:

$$^{m}f(\beta) \equiv \sum_{\alpha > \beta} f(\alpha)$$, where m uniquely identifies a substate

And a parameter:

$$\tau \equiv \sum_{\alpha \in A} f(\alpha) \quad .$$

Now we can define a g-system as a six-tuple:

$$(\tau, \{v_i\}, \{\alpha\}, \{\beta\}, f(.), \{^{m}f(.)\})$$

where as defined above

(1) τ is a parameter;

(2) $\{v_i\}$ is a set of variables;

(3) $\{\alpha\}$ is a set of states;

(4) $\{\beta\}$ is a set of substates;

(5) $f(.)$ is a function on $\{\alpha\}$;

(6) $\{^{m}f(.)\}$ are functions on $\{\beta\}$. [JONE 85c]

Figure 2 is an example of a g-system. Note that it is a discrete valued multivariate system.

v_1	v_2	v_3	$f(\alpha)$
0	0	0	3.70
0	0	1	6.10
0	0	2	9.50
0	1	0	3.70
0	1	1	7.10
0	1	2	14.50
1	0	0	3.70
1	0	1	6.40
1	0	2	10.20
1	1	0	3.70
1	1	1	8.40
1	1	2	16.00

$$v_1 v_2 f(11) = f(110) + f(111) + f(112)$$

$$\tau = 93.0$$

Figure 2: A g-system

Note that this particular g-system is completely defined in that all possible states in the system also have an associated system function value. The definition of a g-system does not require this property, but it will be shown that it is highly desirable. With this definition of a general system, we can now define a K-system. The first part of the transformation to a K-system makes use of the parameter τ that we defined above. A normalization is performed that removes the units from the system and we define the function:

$$k(\alpha) \equiv \frac{f(\alpha)}{\tau}, \forall \alpha$$

This converts the function $f(\alpha)$ to a dimensionless system, $k(\alpha)$, and makes the following properties true for the K-system:

$$0 \le k(\alpha) \le 1, \forall\alpha \quad \text{and} \quad \sum_{\alpha} k(\alpha) = 1 .$$ [JONE 85c]

It is important to note that these properties are the same properties that a probabilistic system has, but that this transformation does not make this into a probabilistic system. It creates a system with sufficient properties that the probabilistic reconstructability analysis algorithms may be applied. Finally, another set of functions is defined:

$$^{m}k(\beta) \equiv \sum_{\alpha \succ \beta} k(\alpha).$$

Now we define a K-system:

$(\tau, \{v_i\}, \{\alpha\}, \{\beta\}, k(.), \{^{m}k(.)\})$

where as defined above

(1) τ is a transformation factor;

(2) $\{v_i\}$ is a set of variables;

(3) $\{\alpha\}$ is a set of states;

(4) $\{\beta\}$ is a set of substates;

(5) $k(.)$ is a function on $\{\alpha\}$;

(6) $\{^{m}k(.)\}$ are functions on $\{\beta\}$. [JONE 85c]

Figure 3 is an example of a K-system. It is the g-system from figure 2 shown above after the transformation has been performed.

The g-system and the K-system are isomorphic in the sense the they both contain the same system information. The system function was scaled so that it has the

v_1	v_2	v_3	$f(\alpha)$	$k(\alpha)$
0	0	0	3.70	0.04
0	0	1	6.10	0.07
0	0	2	9.50	0.10
0	1	0	3.70	0.04
0	1	1	7.10	0.08
0	1	2	14.50	0.16
1	0	0	3.70	0.04
1	0	1	6.40	0.07
1	0	2	10.20	0.11
1	1	0	3.70	0.04
1	1	1	8.40	0.09
1	1	2	16.00	0.17

$v_1 v_2 k(11) = k(110) + k(111) + k(112)$

τ=93.0

Figure 3: A K-system

properties stated above. The relationships between the states and substates remain the same, so the results of reconstructability analysis performed on the k-system clearly map directly back to the g-system. The information is never modified, nor is any information removed or added to the system. Again, it should be noted that this K-system is also completely defined as was noted above for the corresponding g-system.

1.4 Reconstructions with Arbitrary Data

With the advent of the K-system transformation, reconstructability analysis could now be performed with general functions. While this provided the ability to analyze a greater number of function types, there still existed possible impediments in real world data that would prevent the use of reconstructability analysis. The general functions defined in the previous section still required that the variable values be discrete; continuously valued variables were not allowed. While it is trivially true that a finite data set has a finite set of discrete values for each variable, reconstructability

analysis also requires that the variable values be repeated as shown in figures 2 and 3. If the values are not discrete, we have an impediment referred to as data scattering [JONE 85e]. The reconstructability analysis algorithms also require that the data is complete; that is, there must be a system function value for every possible combination of variable values (states). If the system under consideration has possible states that do not have system function values, this is referred to as the problem of missing data. Finally, in real world data there may be redundantly defined states in the sense that there are repeated states in the table and these redundant states may have different function values; this is the problem referred to as state contradictions.

Overcoming these impediments in real world data is at the core of the research reported here. Jones provided basic techniques for overcoming these impediments in [JONE 85d], but while the methods allow reconstructability analysis to proceed, there are significant questions about their use and effectiveness. The focus of this section will be on the problems of data scattering and missing data. The problem of state contradictions has been adequately and reasonably addressed [JONE 85d]. An example will be provided and the method for resolving the contradiction will be described. In general, these three problems should be addressed in the following order. First, data scattering should be resolved, then state contradictions, and finally the problem of missing data [JONE 85d]. Our focus here is on data scattering and missing data, so we will deal briefly with the state contradiction problem first and then move on to the core of the research.

1.4.2 State Contradictions

State contradictions occur when there are multiple system function values for the same state. For example, suppose we have a system that has three variables, each of which can take the values of 0 or 1. Further, suppose that for the state where v_1=0, v_2 =1, and v_3 = 1 (or more concisely, state 001), we have two different values for the system function. We may take an average of the two values and use that single value to represent the state system function value. This does not add any information to the system, it condenses redundant information for a single state. Rather than being an impediment to reconstructability analysis, it can add new dimensions. It is also possible to use other common statistical techniques besides the mean, such as the mode, median, maximum or minimum[JONE 85e]. The analyst is free to choose whatever method that is appropriate for the data being analyzed. It is important to note that while some information is possibly, in some sense, "lost" when handling state contradictions, there is no information added to the system.

1.4.3 Data Scattering

Data scattering refers to the lack of repeated distinct values for each variable in a system that is submitted for K-systems analysis. This problem can best be illustrated by an example from [JONE 85d] reproduced here as figure 4.

Inspection of the table provides a solution to data scattering for this particular set of data. One can see that v_1 is taking the approximate values of {3, 7}, v_2 takes the values {9, 3} and v_3 is taking the values of {9, 6, 7}. Relabeling the states by

v_1	v_2	v_3	$f(.)$
7.2	3.1	9.4	3.7
6.9	2.9	6.2	6.1
6.7	2.7	7.1	9.5
7.1	9.2	9.2	3.7
6.9	8.9	5.9	7.1
7.5	8.6	6.9	14.5
2.8	3.3	9.3	3.7
3.1	2.9	5.7	6.4
3.2	3.0	7.0	10.2
2.7	8.7	9.2	3.7
2.7	8.9	6.3	8.4
2.7	9.2	7.1	16.0

Figure 4: Data Scattering Example

using these values as keys and replacing the actual values with 0, 1, or 2, as appropriate gives us the table shown in figure 5 which is in the exact form required for reconstructability analysis.

There are many one dimensional clustering techniques which can be applied that will resolve the problem of data scattering in this example. When the data is not quite so clearly grouped, the technique applied here may not give as consistent and understandable results as those shown here.

1.4.4 Missing Data

The final problem that must be resolved with real world systems is missing data. Up to this point we have only considered systems which are complete; that is, there is a system function value for every possible state or combination of variable values. Previously, Jones has defined two methods for conducting K-systems analysis when there is missing data [JONE 85d, JONE 89]. One is to modify the algorithms to

ensure that only known states are considered in the analysis and the other is to use an "entropy fill" to provide values for the missing states.

v_1	v_2	v_3	$f(.)$
0	0	0	3.7
0	0	1	6.1
0	0	2	9.5
0	1	0	3.7
0	1	1	7.1
0	1	2	14.5
1	0	0	3.7
1	0	1	6.4
1	0	2	10.2
1	1	0	3.7
1	1	1	8.4
1	1	2	16.0

Figure 5: Resolution of data scattering

First, we will review the method of using only known states. This technique is based on the idea that reconstructability analysis can only be as good as the data which is submitted for analysis. Given the general algorithm for the reconstructability problem, we need not use the complete set of states; we only use those states that exist and provide the most information for determining system behavior. The only changes required to the algorithms are in the computation of the transformation factor, τ, and the calculation of the K-system function k,.

First, τ is calculated for only those states which are known,

$$\tau = \sum_{\alpha, known} f(\alpha)$$

The function values for k are then calculated as,

$$k(\alpha)=\frac{f(\alpha)}{\tau},\alpha \text{ known,}$$

and the corresponding K-system has the property that,

$$\sum_{\alpha\,known} k(\alpha)=1. \text{ [JONE 85d]}$$

An important point is that the values of $k(\alpha)$ for missing states are treated as being unknown, not simply being set to zero. The formation of equivalence classes may then proceed once the states are relabeled in order to ensure that we use the most information possible given the data. Relabeling is done for each variable, where the variable value that occurs most frequently in the data is relabeled as zero and the rest of the values may then be relabeled arbitrarily. Once this has been done, the equivalence classes can be formed as in [JONE 85d] and this was shown to minimize the number of missing equivalence classes. The greedy algorithm may then be executed on this data considering only the existing classes and the analysis will account for only the data which exists. Since the equivalence classes only used known states and the K-system function was constructed using only these states, no information is introduced into the system and the analysis is correct for the given data.

The second method for addressing missing state data is known as the entropy fill. This is a matter of using the overall mean of the known states as the system function value for those states which are unknown. The reason that we wish to fill in these missing states is two fold. First, we may wish to predict the behavior of the system state which is missing, The previous technique ignores the missing data and can provide no prediction about the system behavior for the missing state. Secondly,

we may want to determine the interaction effect of the variables for a missing state. That is, we would like to know what portion of the effect of a state is due to the interaction of the variables and what portion is due to the variables acting alone [JONE 89].

1.5 The Importance Of Reconstructions With Arbitrary Data

The ability to apply K-systems analysis to arbitrary data greatly expands the set of systems that can be analyzed using only the maximum entropy mathematics embedded within reconstructability analysis. Previously, only systems which were explicitly probabilistic could be addressed. With the advent of the K-system transformation, a greater number of systems could be analyzed, but without the resolution of the impediments in real world data the systems which could be analyzed were still limited to only discretely valued systems that were completely specified. Overcoming these impediments allows the formalities of reconstructability analysis to apply to practically any multivariate system. This enables the correct model of the system to be induced without making any of the assumptions of classical statistical analysis [JONE 86]. While the methods described above enable reconstructability analysis for a variety of systems, there are limitations that can be overcome so that the results of the analysis are more meaningful.

1.5.1 Missing Data, Known States, and the Entropy Fill

A reconstruction using only known states is correct for the data as given, but there are aspects of the system that are missed. In particular, predictions of the effects of missing states and the interaction of variables are not possible when using only

known states [GOUW 96]. Also, there is information that exists in the system substates that may be used for determining the values of the missing states. By using the entropy fill technique, these limitations are overcome insofar as predictions may be possible, but the assumption that all missing states assume the overall system mean is, both, too restrictive and too broad.

The entropy fill technique will reduce the amount of variability that the system displays. Obviously, due to the nature of the variance calculation, the variance of a system may be greatly effected by using the same mean value for every missing state. While the effect will not be very significant if only one state is missing, if there are a large number of missing states, the system variability and, by extension, the entropy of the total system will be greatly reduced. In effect the system is "flattened" by assuming the overall mean. The reconstruction will then be weighted toward the mean, and the dynamics of the system will be subdued.

The entropy fill technique is used in order to minimize the amount of information that is added to the system. We propose that the amount of information added to the system when replacing missing states is the same regardless of the system function value used. That is, if we add one state to the system, we have added a set quantity of information. The system function value associated with the state added provides the meaning of that state, but does not effect the quantity of information added. Each state that is added to the system increases the amount of information available about the system exactly the same amount.

1.5.2 Data Scattering and Clustering

Real world data about a system will, more often than not, have scattered variable values. Even in the case of a designed experiment, where a finite discrete set of variable values is selected, the ability to control the values that the variables take is limited. For example, one may wish to use preset temperatures of 25 and 50 to control a chemical process, but given the nature of temperature controls it is possible that temperatures of {24.8, 25.2, ...} and {51.1, 50.2, ...} will actually be recorded. The clustering technique outlined above will adequately handle this specific case, but situations may arise where the solution to the problem is not so clear cut.

An example of a more difficult situation is the following. Suppose we have a variable, v_1, that takes the following values: {0.9, 1.0, 1.1, 2.8, 2.9, 3.0, 3.1}. Upon inspection it appears that this variable is taking two different values, approximately {1.0, 3.0}. Any good clustering algorithm would give the same results for this one dimensional system. Let us also suppose that there are system function values associated with each variable value as shown in figure 6. If we only cluster the data based on the variable values, we miss the obvious fact that by using the system function as a second dimension we have three clusters as shown by figure 7.

v_1	$f(.)$
0.9	5.0
1.0	4.8
1.1	5.2
2.8	6.0
2.9	12.0
3.0	6.1
3.1	11.9

Figure 6: Data for two-dimensional clustering

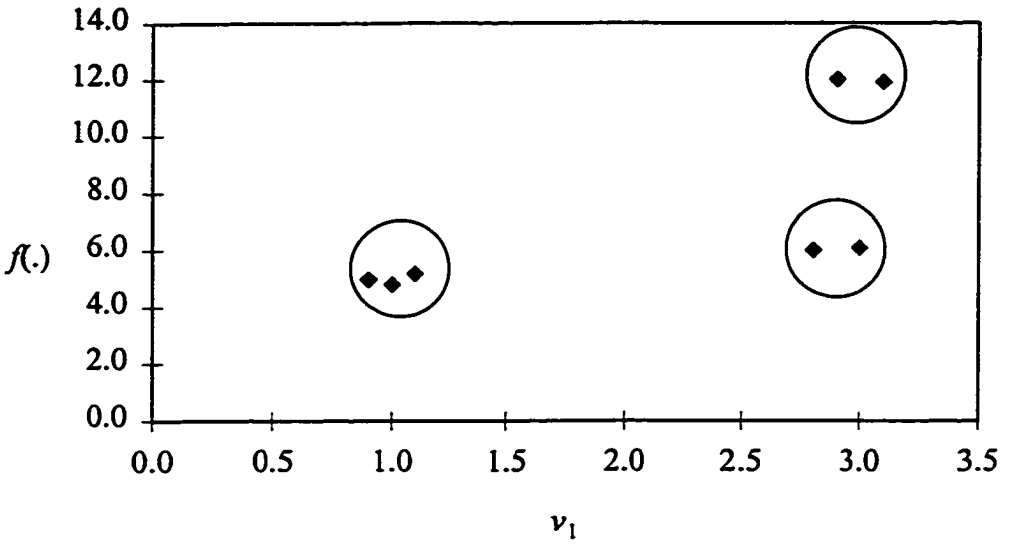


Figure 7: Two dimensional clusters

The reason that there are three clusters is not clear by looking at this data, but clearly there seem to be three. There could be another variable that interacts with this variable or there could be a missing variable. What can be done is for the data to be clustered into three clusters, label each cluster as a variable value, and proceed with the rest of the pre-processing and K-systems analysis. The results of the analysis will tell us which clusters are having the greatest effect in determining system behavior.

The first aspect of this research related to data scattering is to develop a meaningful understanding of how to incorporate two dimensional clusters in reconstructability analysis. While we can directly use two dimensional clusterings in reconstructability analysis, it is not clear how to interpret the results. The second aspect of this part of the research is to develop a clustering algorithm which is congruent with the overall spirit of reconstructability analysis; that is, the idea that we use only information contained in the data, the formalities of information theory and

the principle of maximum entropy. Currently, most clustering algorithms assume a model and then fit the data to that model. We provide a method where the variable values are clustered using only the principle of maximum entropy and the formalities of information theory.

1.6 Existing Missing Data and Clustering Algorithms

There are many existing algorithms for imputing missing data and even more algorithms for clustering data. This section will provide a brief review of existing techniques and place the new methods derived here into specific contexts. The goal is not to review every existing technique, but to identify possible existing alternatives to those currently used in K-systems analysis and to establish a context for the new techniques reported here. Attention will be directed to limitations and assumptions of the algorithms reported in the existing literature that helped to motivate the development of new methods.

1.6.1 Missing Data Algorithms

Statistics have a long history of analysis when there is missing data. Additionally, there is a large number of research reported where missing data is handled in an ad hoc fashion. Most of the literature that specifically addresses missing data is fairly recent, mostly dating from 1970 onward. There are a few review papers that present a comprehensive overview of methods applied for statistical analysis when data is missing [HART 71], [ORCH 72], [LITT 87].

Procedures for dealing with missing data can be grouped into the few non-mutually exclusive categories [LITT 87]. First, there is the simple method of only

using units of data that are completely specified. Any incomplete or missing data is discarded and analysis proceeds using only those data units which are complete.

Second, there are imputation based procedures in which any missing data is filled in based on the existing data. Common methods for imputing missing data include hot deck imputation where values are randomly selected from existing data to fill in the missing data, mean imputation where the overall mean is used for all missing data units (this is the entropy fill), and regression imputation where missing values are imputed based on a regression of the existing data.

Third, there are weighting procedures that are used for non-response in sample surveys. This is related to missing responses for some portions of the data unit where the design weights assigned to possible choices prior to a survey are re-weighted afterwards to adjust the results for non-response.

The final class of procedures falls into the category of model-based missing data procedures. This is perhaps the most actively investigated type of missing data procedure. This class of procedures is based on a defining model for the partially missing data and the basing inferences on likelihood under that model. The focus of model based procedures is on estimating parameters such as the mean and the variance related to the total set of data. Missing data values are not imputed in this method, but the resulting analysis accounts for the missing data in the estimation of the overall parameters.

K-systems analysis requires some form of imputation since a complete set of states is required for the full power of K-system analysis to be used. K-system analysis

can proceed by not using the missing data as in the first class of missing data procedures; only known states are used and the rest are treated as missing. This provides an incomplete analysis of the system and, in particular, the interactions of the variables in states and substates will not be found. As we noted above, the entropy fill technique is a specific instantiation of the an unconditional mean imputation procedure from the second class of missing data procedures. Mean imputation suffers from a number of problems, the most serious of which is the distortion of the empirical distribution due to the under prediction of the variance. The final two classes of procedures, weighting and model based, are specifically the type of procedures that we wish to avoid. First, each assumes that the data being analyzed fits some specific model and then predictions of the effects of the missing data are made based on this model. K-systems analysis assumes no model; only the information contained within the data is used to induce the correct model. Secondly, these procedures do not provide values that may be used for K-systems analysis. Both procedures account for missing data so that summary parameters about the system may still be calculated and the results will be consistent with the model that is assumed to underlie the data being analyzed.

There is one existing imputation procedure that is similar to the technique that is proposed here. It is referred to as a nearest neighbor hot deck imputation [SAND 83]. It consists of defining a metric to measure the units based on the values of the covariates and then selects a value used in the set of closest units for the missing value. This is similar to the method proposed here, except that the distance is defined

on the states and a search is not conducted for the closest states; the closest states are defined by the structure of the system. Also, no single closest state is used to impute the missing values, instead, an unbiased estimator of all closest states is used for the imputed values.

1.6.2 Clustering Algorithms

There is extensive literature on clustering and there are many different methods and techniques that are available. Clustering is often considered referred to as a form of unsupervised learning or self-organization [BEZD 92]. Much of the current work related to clustering is in the field of fuzzy clustering. A clustering algorithm classically produces a hard clustering; each data unit is assigned to a single cluster. In fuzzy clustering, the general idea is to assign each data unit some level of membership in each cluster in accordance with the principles of fuzzy logic [ZADE 65]. The focus of the research here is to generate hard clusters so that the results of the clustering can be directly applied in K-systems analysis. Even if a fuzzy clustering procedure is applied to a data set, virtually all fuzzy partitions can be hardened by applying some method of determining the maximum cluster membership for each point.

Clustering algorithms may be divided into a number of groups based on the manner in which the clusters are formed. There are joining algorithms where each data unit is initially considered to be in a separate cluster and then clusters are iteratively joined based on a measure of similarity. Alternatively, there are splitting algorithms where all of the data units are initially assigned to a single cluster and then this initial cluster is then divided into smaller clusters. There are also switching style algorithms

where initial clusters are formed and then data units are switched between clusters in order to optimize some criteria. While this classification of clustering algorithms is by no means complete, it demonstrates the wide variety of clustering techniques that have been applied.

There is extensive literature about various clustering algorithms and their application to specific fields. Some general references for clustering include [ANDE 73], [HART 75], [EVER 93], and [BEZD 92] There are too many different clustering algorithms available to even make a passing attempt at providing an overview of all of them, but two recent algorithms that are relevant to the results reported here will be briefly reviewed.

Both algorithms are based on the principle of maximum entropy and the similarity measure and algorithm proposed here are also based on this principle. One is referred to as a least bias clustering algorithm in that it selects cluster centroids such that there is no initial bias towards any of the points [BENI 94]. It then proceeds to make each data point iterate towards one of the cluster centroids. The number of clusters is determined by varying a resolution parameter between zero and one and then counting the number of clusters that results over all values. The number of clusters that results most often is considered the most probable number of cluster. One possible problem with this algorithm is that it assumes that the entropy can be modeled by a particular type of distribution and it uses this assumption to generate the results. Also, as is typical of clustering algorithms, the results are reported for two relatively simple cases and are determined to be perceptually correct.

The other algorithm uses a simulated annealing approach for determining the clusters and implements the principle of maximum entropy based on the same distribution used in the preceding algorithm [HOFM 97]. This algorithm is based on the use of pair wise measures of similarity between all data points. This is similar to the method proposed here in that it also is based on the principle of maximum entropy and includes a pair wise measure of similarity. The annealing algorithm is able to use a similarity measure that is not a true metric and then applies an annealing algorithm to maximize the entropy as approximated by a distribution. There is no requirement placed on the type of similarity measure that may be used.

Both of the preceding algorithms are based on the principle of maximum entropy, but approximate the maximum entropy distribution by assuming a particular type of distribution, the Gibbs distribution. The work reported here does not use an approximation to the entropy, but instead calculate the entropy directly based solely on the data as presented to the algorithm. Both algorithms use the principle, but neither uses the underlying mathematics of information theory to determine the answer. There are many other clustering algorithms based on many other measures and assumptions. The two briefly outlined here are closest to the approach presented here and the work in [HOFM 97] may be used within the context of the entropy similarity measure that will be described in chapter 3.

Chapter 2. Incomplete Systems: Missing Data

The impediments of missing data and data scattering are actually the same problem viewed from two different perspectives. Missing data is the belief that all the variables in the system are non-continuously valued variables. Even if the values that a variable takes are listed as

{1.2, 1.3, 1.4, 2.2, 2.3, 2.4}

these values are already discrete and do not require any type of clustering or clumping together of values. If these variable values and the other variables and their values are all viewed as deriving from discrete variable systems, then there may be some combinations of variable values (states) that are missing.

Alternatively, if these values are believed to have been derived from a system that has variables that are continuously valued, the values are viewed as being scattered. If this belief is true, then it is perfectly valid to cluster these variable values into groups of points centered around 1.3 and 2.3. These values may then be used in place of the original values and analysis, of any type, may proceed.

It is useful to consider the two problems of data scattering and missing states as a single impediment to the use of K-systems analysis. This single problem may be readily viewed as having an incomplete system. Suppose that the data shown in Figure 8 is submitted for analysis.

The two views described above are possible based on inspection of this data. Either the values for variable v_1 are scattered or there are a significant number of states

v_1	v_2	v_3	$f(\alpha)$
1.3	0	0	3.7
1.2	0	1	6.1
1.4	0	2	9.5
1.4	1	0	3.7
1.2	1	1	7.1
1.3	1	2	14.5
2.4	0	0	3.7
2.2	0	1	6.4
2.4	0	2	10.2
2.3	1	0	3.7
2.3	1	1	8.4
2.2	1	2	16.0

Figure 8: Example of an incomplete system

involving the values of variable v_1 missing from this system. Without some prior knowledge about the source of this data, it is possible to make very poor decisions about how to resolve this incomplete system. The question that must be asked is how is a complete system can be induced from the existing system that is consistent with the known data, yet only adds minimal assumptions about the source from which the data is derived. While a completely general answer that will apply to all possible situations is unlikely, it is possible to develop algorithms that use only known information about the system and add minimal information based solely on the existing system information.

One part of the research presented here is to apply the notion of the distance between states in order to find a better value to use for the missing state. Instead of using the overall system mean, we would use a local mean based on those states which are close to the missing state. Since we are adding the same amount of information (the same number of states) to the system as the entropy fill, we do not add more

information to the system. By trying to use only local data to determine a value for the missing state, we can calculate "better" values; values that more accurately capture the dynamics of the system. This will enable better predictions of the effects of missing states on the reconstructed system and more accurate calculations of interactions. The second part of the research, reported in chapter 3, is related to using the overall entropy of the data and the calculated clusters for determining the appropriate clustering. Starting with the basic definition of information entropy, a similarity measure will be developed for use in a density based clustering algorithm.

2.1 Systems, Structure and Missing Data

First, the problem of incomplete systems will be addressed from the point of view of missing data. This section will develop a method whereby only the existing information will be used along with the existing structure provided by the information contained in the system. The structure of a system will be defined based solely on the information present in the data and this structure will then be used to fill in or impute the missing system function values.

A general system can be defined as consisting of a structure system that consists of the set of variables, V, and the set of values that each variable may be assigned, v_i; this also defines the set of states, A. So the structure of the system is completely defined by the states that exist in a system. Associated with each state is a system function value which may be interpreted as giving the state some meaning. That is, for an arbitrary structure system there may be many different system functions. These system functions describe the meaning of the structure and without

the system function the structure system has no meaning. We place no constraints on the structure system other than requiring each state be complete in itself; we do not explicitly consider substates. This means that if there are three variables in the structure system, each existing combination of variables consists of a value for each of the three variables. Each variable may take values from any arbitrary, finite set whether it is the set of real numbers, integer, natural numbers or some arbitrary set of values (e.g., {red, green, blue}). Also, the variables are not required to be homogenous, each may take different types of values. Figure 9 is an example of an arbitrary structure system and the associated system function values.

v_1	v_2	v_3	$f(\alpha)$
red	0.01	2	3.7
blue	0.12	15	6.1
red	0.12	2	9.5
green	0.12	15	3.7
green	0.12	2	7.1
red	0.01	15	14.5
blue	0.01	2	16.0

Figure 9: Example of a structure system

This system may be viewed as the subset of the complete structure system where the values of the system function are known. From the known structure system, we wish to impute values for the complete structure system from the parts of the structure that are currently known. The first step for inducing a complete system is to identify the overall structure system based solely on the information given.

The variable values for each variable will be relabeled so that all types of variables may be handled in the same manner. Each variable value is relabeled by counting the number of distinct values starting from zero and then relabeling each

value with the index, assigning the same index to values which are the same. The example in figure is thereby transformed as shown in Figure 10. Note that this transformation is an isomorphism in that one systems maps directly to the other and neither adds information to nor removes information from the original system.

v_1	v_2	v_3	$f(\alpha)$
0	0	0	3.7
1	1	1	6.1
0	1	0	9.5
2	1	1	3.7
2	1	0	7.1
0	0	1	14.5
1	0	0	16.0

Figure 10: Example of a relabeled structure system

This relabeling results in a structure system that is similar in form to the original behavior systems defined in the previous chapter. This transformation also allows the computations of the total number of state in the structure system, namely the product of the number of values that each variable takes, or equivalently, the product of the cardinality of each set of variable values. For the example currently under discussion, there are a total of (2) (2) (3) = 12 possible states. Given the seven existing states in the example we are, therefore, missing five states.

Previously, the entropy fill was used to find values for the missing states by calculating the average of all known states and using this value for all of the missing states. While this technique provided a method for imputing the missing function values, it created a system that did not capture the full dynamics of the original system in that by using the value of the mean repeatedly, it reduced the overall variability of the system. This was due, at least in part, to the assumption that all missing states were

similar to the system mean regardless of the structure of the system. This is the same as assuming that all states that are in the system are equally similar to every other state; there is no way to distinguish one particular state as being more similar to another state than it was to any other state in the system. The algorithm that will be described here will use the existing structure and associated information of the given system to impute the values for missing states.

2.2 Distance Between States

In order to further illuminate the structure of a system, a distance function will be defined on the set of states. This distance function will be applicable to any set of states and will be shown to be a true distance function or metric because it satisfies the identity and symmetry axioms and the triangle inequality. By defining a distance function for states, it will be possible to use the resultant structure of the system to impute values for missing states.

Without loss of generality, states will be represented by fixed length strings, where the length of the string is the number of variables and each position in the string may take one of the values from each variable's set of values. The ordering of variables in the string is fixed yet arbitrary; changing the order of the variables does not effect the results presented here, so long as the ordering of the variables is constant throughout the application of the algorithm. Using this representation, the set of known states from the previous example is {000, 111, 010, 211, 210, 001, 100}. The distance between two states is defined in a fashion that is similar to the Hamming distance, that is, the distance between two states is the number of positions in the state

strings where they differ. The only difference from the standard Hamming distance is that the values that are available for each position are not restricted to the set {0, 1}. Let the distance between two states, α_1 and α_2, be denoted as $d(\alpha_1, \alpha_2)$. For example, if we have the states $\alpha_1 = 211$, $\alpha_2 = 210$ and $\alpha_3 = 101$, the we have $d(\alpha_1,\alpha_2) = 1$, $d(\alpha_1,\alpha_3) = 2$ and $d(\alpha_2,\alpha_3) = 3$.

This definition of distance is a true metric in the sense that it satisfies the following typical properties:

1. $d(\alpha_1, \alpha_2) = 0$ if and only if $\alpha_1 = \alpha_2$ (the identity axiom)

2. $d(\alpha_1, \alpha_2) = d(\alpha_2, \alpha_1)$ (the symmetry axiom)

3. $d(\alpha_1, \alpha_3) \leq d(\alpha_1,\alpha_2) + d(\alpha_2, \alpha_3)$ (the triangle inequality)

The identity and symmetry axioms are obviously true based on the definition of state distance. The triangle inequality can be assumed to be true since the state distance defined here is similar to the Hamming distance, but the states are not restricted to variables that only take the values of 0 or 1. Therefore, we provide the following proof that the state distance satisfies the triangle inequality.

Proof. Suppose that we have three states, α_1, α_2, and α_3, such that $\alpha_1 \neq \alpha_2 \neq \alpha_3$ and that $d(\alpha_1,\alpha_3) = k$. Then the string representation of the states for α_1 and α_3 have differences in k positions. Assume that there exists an α_2 such that $d(\alpha_1,\alpha_2) + d(\alpha_2,\alpha_3) < k$. Suppose that α_2 is picked so that the $d(\alpha_1,\alpha_2) = 1$, that is α_1 differs from α_2 in exactly one position. Then $d(\alpha_2,\alpha_3) < k - 1$ for the strict inequality to be true. We can transform α_2 into α_1 by changing the one position where they differ. By the definition of the state distance, $d(\alpha_1,\alpha_3)$ would be at best k-1, but $d(\alpha_1,\alpha_3) = k$

and changing one position in α_1 can only leave the distance the same (if we change a position where they already differ to a value where they still differ) or, at best, decrease the distance by one (if we make the one differing position the same). Therefore, $d(\alpha_2,\alpha_3) \geq k\text{-}1$. For any α_2 that we pick we have $d(\alpha_1,\alpha_2) = q$ and every α_1 can be transformed to α_2 by changing q positions and the distance from each α_2 must be at least $(k - q)$. Therefore, assuming that

$d(\alpha_1,\alpha_2) + d(\alpha_2,\alpha_3) > k$, implies that

$q + (k - q) > k$

$k > k$

which is false. Therefore, $d(\alpha_1,\alpha_3) \leq d(\alpha_1,\alpha_2) + d(\alpha_2,\alpha_3)$ and it is proven that the triangle inequality holds and the state distance is a true metric. □

Now we have the structure of the system defined as the states and a distance metric on the Cartesian space of the states. This allows a type of similarity calculation to be performed so that instead of using the all of the existing states to determine the value of a missing state, only the most similar or closest states can be used. Applying this idea to the set of missing states leads directly to an algorithm that uses the closest states to determine appropriate values for those states.

The state distance is a true metric on the Cartesian state space. This identifies additional structure in the state space along with the structure that may already be discovered though the application of K-systems analysis to a complete system. We can now use the state distance to impute the missing values from an incomplete system. It is useful to note that the state distance is a direct generalization of the Hamming

distance defined on binary strings. Instead of binary strings, states can be viewed as a generalized string where the maximum length of a string is the number of variables in the system and each variable may be assigned values from a different set of symbols, not just a symbol from the binary set {0,1}.

2.2 Closest States

Given the state distance it is now possible to rank all of the states in a system relative to a single particular state. In particular, we can find a set of states which are closest to any state that is missing a system function value and use this set to impute a value. Since the state distance may take values from the set of natural numbers, the set of closest states consists of all states that are a distance of one from the state that is missing a value.

$$A_c^{\alpha} = \{\alpha_i \mid d(\alpha,\alpha_i) = 1, \forall \alpha_i \in A\}.$$

The following example illustrates the idea of the set of closest states. Suppose we are given a system that has three variables, v_1, v_2, and v_3 that take values from the sets {0,1}, {0,1}, {0,1,2}, respectively. Therefore, we can construct the set of states shown in table 11. The table also shows the known values of the system function $f(\alpha)$.

Let us consider two states as examples, 010 and 102. First, we will construct the set of closest states for the state 010. We can apply the definition of state distance and find all states that are a distance of one from this state. This can be readily done by alternating each variable value, in turn, to another value from the set of possible values

v_1	v_2	v_3	$f(\alpha)$
0	0	0	3.7
0	0	1	6.1
0	0	2	9.5
0	1	0	3.7
0	1	1	7.1
0	1	2	14.5
1	0	0	3.7
1	0	1	6.4
1	0	2	10.2
1	1	0	3.7
1	1	1	8.4
1	1	2	16.0

Figure 11: Example System

for that variable and this produces the set $A_c^{010} = \{110, 000, 011, 012\}$. Each of these states differs in one variable position from the missing state 010. We can perform the same construction on the state 102 and we generate the set $A_c^{102} = \{002, 112, 100, 101\}$. In general, it is also possible to construct sets of states that are any distance from a given state up to the maximum distance, which is the number of variables in a system.

2.3 Use of the Closest States

Given some arbitrary set of states and the associated system function values, we can now define an algorithm for imputing system function values for all missing states. The idea that underlies this algorithm is the following: use only the states that are most similar (or closest) to impute a value for a missing state; this is a form of maximum likelihood estimation. The distance metric defined in the previous section can be used to determine which states from the set of all states are closest to the

missing state. Intuitively, this is a matter of using states where the values of all variables are the same, except for one variable.

The set of closest states may be readily formed based on the state distance, but it is not immediately clear how this set of states should be used to impute a value for the missing state. We observe that the structure of the system based on the state distance is similar to the structure of an error correcting code. This similarity can be used to give an indication on how the set of closest states can be used. Error correcting codes are constructed so that the Hamming distance between code words is maximized. This allows errors to be detected by recognizing that the current code word is not in the set of valid code words and allows error correction by selection of the valid code word that is closest to the code word containing an error. This is a type of maximum likelihood decoding in that it selects the valid code word based on the closest code word and the probability that there are a specific number of errors. In the case of a system consisting of variables and the values that each can take, the states correspond to the code words in a communication system and code words with errors correspond to missing states. Since the system was not explicitly constructed to maximize the distance between valid code words, all of the states exist and there is no single state that is closest to the missing state. Instead there is a set of states that are closest to the missing state and all the closest states are equally probable corrections to the missing state. The most likely value of the missing state is the expected value of the set of closest states, namely, the mean. The mean of the set is an unbiased estimator of the missing state. This is in keeping with the maximum entropy principle

that is the basis of the reconstructability algorithms. Using the mean of the closest state set is the analog of the maximum likelihood solution to the error correcting codes.

Alternatively, other estimators for the missing value may be used in a similar fashion to the resolution of state contradictions. The analyst may wish to use the maximum, the minimum, the median or the mode as the estimator for the missing state. Also, other methods may be applied using the distance function and additional sets of states that are further away from the missing state. It may be possible to use the relationship of the members of the closest state set to their own corresponding closest state sets to determine the values that should be used for the closest states. In particular, it is possible to use the deviation from the mean of the closest state set members from the mean of their corresponding closest state sets. In effect, this incorporates information from states that are one step further away from the missing state and weights this effect by adjusting the mean value of the closest states by the average deviation of the actual value from the closest state's closest states.

The mean of the closest state set is used here for a number of reasons. First, the mean is an unbiased estimator of an unknown distribution. An unbiased estimator is in keeping with the principle of maximum entropy in that it makes the fewest assumptions regarding the underlying structure. Also, if the original variable values are not from a set of variable values that ordered, it may not be possible to assign a magnitude to these values and thereby use the variable values to perform some other type of estimation. Using the mean of the missing states places the least constraint on

the type of systems that may be considered for the closest state algorithm and, by extension, submitted for K-systems analysis.

In addition, to the previous arguments the set of closest states may be characterized by the amount of information that it shares with the missing states. Using the state distance metric, it is possible to define the amount of information that states share based on their distance. In addition, the average amount of information that a state shares with a particular set of states may also be determined. The state distance is defined on the number of variable values that are the same between a pair of states and each variable may be considered a single information entity. We can use the notion of similarity and distance to show why using the closest states is superior to the entropy fill.

We begin by defining the amount of information shared by two states as the proportion of the number of positions in the state string that are the same. This is the following function of the state distance:

$$I(\alpha_1,\alpha_2)=\frac{N_v-d(a_1,\alpha_2)}{N_v}, \tag{2.1}$$

where N_V is the number of variables and d is the state distance as defined above. For a given system and a particular state, α, all members of the closest state set have a distance of one from the state. This implies that the information shared between a particular state and each member of the closest state set is equal to

$$I(A_c,\alpha)=\frac{N_v-1}{N_v} \tag{2.2}$$

as defined by equation 2.1. We claim that the closest state set shares more information with the missing state than the set of all states used in the entropy fill and, in general, shares the most information possible with the missing state.

Proof: As defined by 2.1, the average amount of information shared between a particular state and its closest state set is $I(A_C,\alpha)$. The entropy fill technique uses the set of all existing states. The average amount of information shared by this set with the missing state can be defined as the average of each of the following sets formed for each possible state distance from α:

$$A_n = \{\alpha_i \in A_n, d(\alpha,\alpha_i) = n, n = \{1,2,\ldots,N_v\}\} \tag{2.3}$$

So we can calculate the average amount of information shared by each member of the above sets as:

$$\bar{I}(A_n,\alpha) = \frac{\sum_{\alpha \in A_n} \frac{N_v - n}{N_v}}{|A_n|} \text{ and} \tag{2.4}$$

$$|A_c| \le |A_n|, 2 \le n < N_v.$$

This implies that the average amount of information that the total set shares with the missing state is:

$$\bar{I}(A',\alpha) = \frac{\sum_{n=1}^{N_v} \sum_{\alpha \in A_n} \frac{N_v - n}{N_v}}{|A'|}, \text{ where } A' = A - \alpha \tag{2.5}$$

$$\le \frac{\frac{N_v - 1}{N_v} + \frac{N_v - 2}{N_v} \ldots + \frac{1}{N_v} + \frac{0}{N_v}}{N_v}. \tag{2.6}$$

Note that the term for those states that are the maximum distance from the state of interest are included and that this term is zero, but it is included as member for the calculation of the average shared information. This yields the N_v in the denominator instead of N_v - 1 if the term had not been included. This expression simplifies to:

$$\bar{I}(A',\alpha)=\frac{\sum_{i=0}^{N_v-1} i}{N_v^2} \tag{2.7}$$

$$=\frac{\frac{1}{2}N_v^2-\frac{1}{2}N_v}{N_v^2} \tag{2.8}$$

$$=\frac{1}{2}-\frac{1}{2N_v} \tag{2.9}$$

We know that as

$$N_v\to\infty,\bar{I}(A',\alpha)\to\frac{1}{2}, \text{ and as} \tag{2.10}$$

$$N_v\to\infty,\bar{I}(A_c,\alpha)=\frac{N_v-1}{N_v}\to 1. \tag{2.11}$$

Therefore, the set of closest states shares more information, on average, with a missing state than does the set of all states.

It is obvious from the preceding that the set of closest states is also the largest set of states that shares the maximum amount of information on average with any particular state. A subset of the closest state set will have the same average shared information, but will be a smaller set. Any larger set will include states which must share less information with the initial state and, therefore, will have a lower average shared information. This implies that the closest state set consists of the most similar

states that share the most information possible with the initial state. The closest states can then be used to impute values for the system function and the only assumption about the function on the states is that each variable that is used in the state is also used in the function.

2.4 Closest States Algorithm

Now that we have the set of closest states that was created based on the state distance function, this enables the determination of the mean for this set to be used as the unbiased estimator for the value of the missing state. Based on these definitions and principles we can begin to define an algorithm that can be used to impute values for the missing states.

The following notations will be used for the algorithms that follow. The number of variables will be denoted as N_v, the number of values that each variable, v_i, takes will be denoted as n_{vi} and the total number of states is denoted as N_n. The following is the basic algorithm for calculating the system function values for all missing states.

First, we define a data structure that will be used to store and track the states and associated systems function. It should be noted that the data structure that is used here is purposely very primitive. Various programming languages may provide capabilities, such as multidimensional arrays, that may simplify the implementation of the algorithm. The following data structure consists of an array of the variable values and the system function value.

(1) data structure State

(2) Variable(N_v) : Integer

(3) SystemFunc : Double

(4) end data structure

The following routine is the high level portion of the algorithm that will maintain two copies of the structure system. Since the algorithm imputes values for every missing state based on the existing system, this allows the original system values to be used and then filled into the other copy without inadvertently filling in values that are considered missing that will be subsequently used for imputing other missing states. This routine assumes that the State structure has already been populated by all existing states and that the number of variables and the number of values that each variable is allowed to take is already known. Also, it assumes that the missing states have been assigned a special value referred to as the MISSING_FLAG. This allows the algorithm to account for any missing states that have a closest state set that is missing all of the system function values. It will iterate through the complete set of states, filling in missing states where possible and then check to see if all the states are imputed. If not, it will perform the closest state loop again until each state has been imputed. There is an assignment that is made to copy all of the states in one copy to the states in another copy. The implementation of this routine is trivial and at the most straightforward iterates through every state assigning the values from one state to the other. Again, specific languages may have capabilities that make this routine even more elementary, allowing direct assignment of one array to the other.

(1) Routine ImputeMissingStates

(2) CurrentStates(0 to N_n - 1): State, NextStates(0 to N_n - 1): State

(3) Complete : boolean

(4) Complete = false

(5) while ¬ Complete

(6) Complete = true

(7) NextStates() = CurrentStates()

(8) for i = 1 To N_n

(9) if (NextStates(i).SystemFunc = MISSING_FLAG) Then

(10) NextStates(i).SystemFunc = ClosestStateMean (CurrStates(i) , CurrentStates(),N_v, n_{vi}())

(11) if (NextStates(i).SystemFunc = MISSING_FLAG) Then

(12) Complete = false

(13) fi

(14) fi

(15) Next i

(16)Wend

The following routine iterates through each state, checks to see whether each state is missing and, if it is, calls that routine that calculates the mean of the corresponding closest state set.

(1) Function ClosestStateMean (CurrentState:State, theStates(): State, N_v, n_{vi}())

(2) i, j: integer, AState:State, stateSum: real, divisor: real, adder: real

(3) stateSum = 0

```
(4) divisor = 0
(5) For i= 1 To N_v
(6)     For j = 1 To n_vi(i)
(7)         if ¬ (j  = currentState.values(i)) then
(8)             Astate = currentState
(9)             AState.values(i) = j - 1
(10)            adder = theStates(GetIndex(N_v, n_vi(), AState)).SystemFunc
(11)            if (adder = MISSING_FLAG) then
(12)                stateSum = stateSum + adder
(13)                divisor = divisor + 1
(14)            fi
(15)        fi
(16)    next j
(17)next i
(18)if ¬(divisor = 0) Then
(19)    ClosestStateMean = stateSum / divisor
(20)Else
(21)    ClosestStateMean = MISSING_FLAG
(22)fi
```

The following routine is responsible for returning the index of a particular state based on the variable values of that it is assigned. Again, note that the variable values are assumed to have been relabeled so that each variable, v_i, takes values from the set

{0, 1, . . . n_{vi}}. This routine will then return the index into the state array that corresponds to the particular state and, thereby, allows access to the corresponding system function value.

(1) routine GetIndex (N_v, n_{vi}(), AState:State)

(2) i, multiplier, rv : integer

(3) rv = 0

(4) multiplier = 1

(5) For i = N_v - 1 DownTo 0

(6) rv = rv + AState.Variable(i) * multiplier

(7) multiplier = multiplier * n_{vi}(i)

(8) Next i

(9) return rv - 1

(10)end routine

2.4.1 Space and Time Complexity of the Closest States Algorithm

Given these routines, we can determine the space and time complexity of the overall algorithm for calculating imputed values for missing states. First, the ImputeMissingStates routine iterates through each of the states at least once in the loop starting at statement 8. This implies that there will be a total of N_n iterations through this loop. In addition, there is a while loop starting at statement 5 that ensures that every missing state receives an imputed value. This loop is necessary because the closest state set of any specific missing state may be empty; that is, none of the closest states has a system function value. The maximum number of times that this loop may

be executed is readily seen to be the maximum distance between any two states in the system. The maximum distance between any two states is the number of variables in the system, N_v. This implies that the complexity of this particular routine is $O(N_v N_n)$

Within this loop is a call to the ClosestStateMean routine which has two loops in it. One loop, beginning at statement 5, iterates though all of the variables and the other loop iterates through all of the values for each variable. This means that there are n_{vi} iterations in the inner loop and N_v iterations in the outer loop for a total of $(n_{vi} N_v)$. This expression is equal to the total number of states, so the complexity of this routine is $O(N_n)$.

Finally, the routine that returns the index of an arbitrary state in the system based on the assignment of values to the variables has a complexity of $O(N_v)$, the number of variables in the system. The overall complexity of the algorithm is then a product of all of these complexities, namely, $O(N_v N_n \; N_n \; N_v)$. This expression simplifies to $O(N_v^2 \; N_n^2)$. In general, the number of states will be far greater that the number of variables in the system, $N_n >> N_v$. This is particularly true if there are many different values that each variable can take, so we may characterize the complexity of the overall algorithm solely in terms of the number of states, $O(N_n^2)$. Finally, we observe that there are two copies of the states maintained throughout the algorithm, so we know the space complexity is $2N_n$ or $O(N_n)$.

2.4.2 Closest State Algorithm Examples

To demonstrate the effectiveness of the closest state algorithm, the following examples demonstrate the results that can be expected. In particular, the results are

compared to the entropy fill technique. Consider the following system which is based on the following function using the values indicated in the table in figure 12:

$$f(\alpha) = v_1 \sin\left(\frac{v_1 \pi}{(v_3 + 1)}\right) + (v_2 + 2)^{v_3} + v_3\sqrt{2 + v_1} + 2.7$$

v_1	v_2	v_3	$f(\alpha)$
0	0	0	3.70
0	0	1	6.1
0	0	2	9.5
0	1	0	3.7
0	1	1	7.1
0	1	2	14.5
1	0	0	3.7
1	0	1	6.4
1	0	2	10.2
1	1	0	3.7
1	1	1	8.4
1	1	2	16.0

Figure 12: Original System - No Missing Data

For this example, assume that the following states are missing from the system: 001, 012, 101, and 111. The first step to the algorithm is to form the set of closest states for each missing state, so we have the following sets:

$A_c^{001} = \{101, 011, 000, 002\}$

$A_c^{012} = \{112, 002, 010, 011\}$

$A_c^{101} = \{001, 111, 100, 102\}$

$A_c^{111} = \{011, 101, 110, 112\}$

For each set we calculate the mean of the closest state set counting only those closest states that are not missing themselves. So, we have the following results. Note

that we use the symbol ϕ to indicate a missing function value which will effect the number of values used for calculating the mean of the closest state set.

$$f(001) = (f(101) + f(011) + f(000) + f(002)) / n$$

$$= (\phi + 7.1 + 3.7 + 9.5) / 3 = 6.8$$

$$f(012) = (f(112) + f(002) + f(010) + f(011)) / n$$

$$= (\phi + 9.5 + 3.7 + 7.1) / 3 = 9.1$$

$$f(101) = (f(001) + f(111) + f(100) + f(102)) / n$$

$$= (\phi + \phi + 3.7 + 10.2) / 2 = 7.0$$

$$f(111) = (f(011) + f(101) + f(110) + f(112)) / n$$

$$= (7.1 + \phi + 3.7 + 16.0) / 3 = 8.9$$

The overall system mean of the existing states is 7.2. The following table provides a direct comparison with the known values, $f(\alpha)$, the values calculated using the entropy fill, $f_e(\alpha)$, and the values calculated using the closest states algorithm $f_c(\alpha)$.

v_1	v_2	v_3	$f_e(\alpha)$	$f_c(\alpha)$	$f(\alpha)$
0	0	1	7.2	6.8	6.1
0	1	2	7.2	9.1	14.5
1	0	1	7.2	7.0	6.4
1	1	1	7.2	8.9	8.4

Figure 13: Comparison of entropy fill and closest states

The following example demonstrates the behavior of the algorithm when one of the missing states has an empty closest state set. The system will be the same as the system from the precious example, but will have the following missing states: 001, 000, 011, 101 and 002. The mean of the remaining states is 8.6. We proceed with the

algorithm as before by first forming the sets of closest states for each missing state and then calculating the mean of the closest state set.

$$f(000) = (f(100) + f(010) + f(001) + f(002)) / n$$

$$= (3.7 + 3.7 + \phi + \phi) / 2 \quad = 3.7$$

$$f(001) = (f(101) + f(011) + f(000) + f(002)) / n$$

$$= \phi + \phi + \phi + \phi \ / 0 \qquad = \text{undefined}$$

$$f(011) = (f(111) + f(001) + f(010) + f(012)) / n$$

$$= (8.4 + \phi + 3.7 + 14.5)) / 3 \quad = 8.87$$

$$f(101) = (f(001) + f(111) + f(100) + f(102)) / n$$

$$= (\phi + 8.4 + 3.7 + 10.2) / 3 \quad = 7.43$$

$$f(002) = (f(102) + f(012) + f(000) + f(001)) / n$$

$$= (10.2 + 14.5 + \phi + \phi) / 2 \quad = 12.35$$

Note that the state 001 had no states in its closest state set that had an associated function value. This is easily remedied in the algorithm by checking to see if all of the states have values and, if not, another pass is made through the closest states algorithm Only those values which are still undefined are considered missing and the algorithm will proceed as before except that the imputed values for the missing states will then be used to attempt to fill in the remaining states. So, the example proceeds by using the imputed values for the states in the closest state set of 001. See figure 14 for summary..

$$f(001) = (f(101) + f(011) + f(000) + f(002)) / n$$

$$= 3.7 + 8.87 + 7.43 + 12.35 \ / 4 \quad = 8.09$$

2.5 Alternative Closest State Algorithm

The state distance and the general idea of using the closest states can be extended to a number of different realizations of a closest states algorithm. In this

v_1	v_2	v_3	$f_e(\alpha)$	$f_c(\alpha)$	$f(\alpha)$
0	0	0	8.6	3.7	3.7
0	0	1	8.6	8.1	6.1
0	0	2	8.6	12.4	9.5
0	1	1	8.6	8.9	7.1
1	0	1	8.6	7.4	6.4

Figure 14: Comparison of entropy fill and closest states

section, two alternatives will be briefly considered along with some examples using them. Using the mean of the closest states is convenient and makes the fewest assumptions regarding the structure of the systems and the breadth of effect of the variables, but adding additional information from more distance states may frequently be justified and may also provide superior results. The two alternatives that we briefly consider here are an iterated version of the preceding closest states algorithm and a method of using the closest states to the members of the initial closest state set.

2.5.1 Iterated Closest States

A direct extension of the closest state algorithm is an iterated version. Imputing values for the missing states based on the set of closest states provides reasonable and quick estimates of the missing values. It is also possible to take the idea of state distance and closest states and apply it iteratively until the imputed values converge to some fixed point. The algorithm would proceed as before, but instead of just checking to see whether all of the states have values and then terminating, it begins to iteratively

calculate the means for all of the states that were originally missing. It proceeds in this manner until all of the states have reached a stable value.

Examples have been executed using this algorithm and the imputed values that it provides seem to be reasonable. Compared to the non-iterated version, the values tend to be closer to the entropy fill values. This is appeals to intuition in that each iteration propagates values from states which are further away from the missing state and with enough missing states, the imputed values would be expected to be quite close to the overall system mean. The following series of plots will compare the imputed and original values for all of the states as more known states are added to the system. The equation that is the basis of the system being compared is the following:

$$f(\alpha) = v_1 \sin\left(\frac{v_2}{\pi}\right) + v_1 v_3 + v_2 v_3^2,$$

and all variables take values from the set {0, 1, 2}. The plots begin with only three known states (000, 111, 222) and proceeds to add known states to the system which reduces the number of missing states. The plots make it obvious that as the amount of missing data in the system is reduced, both the original and the iterated version yield increasingly better results.

It can be seen from the preceding figures that the iterated version and the original version of the closest states algorithm provide reasonable results for the missing states. In general, it can be seen that the non-iterated version seems to over-predict the amount of variance in the system and the iterated version seems to under predict. Taken together, they seem to provide bounds on the variability of the missing

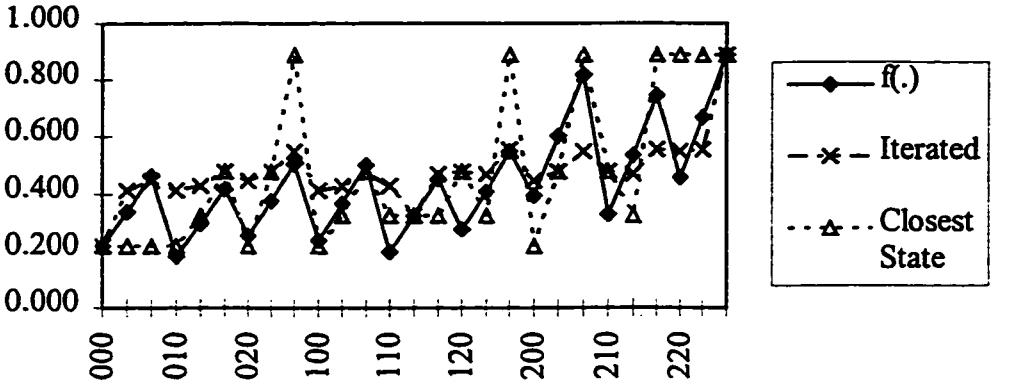


Figure 15: States 000, 111, and 222

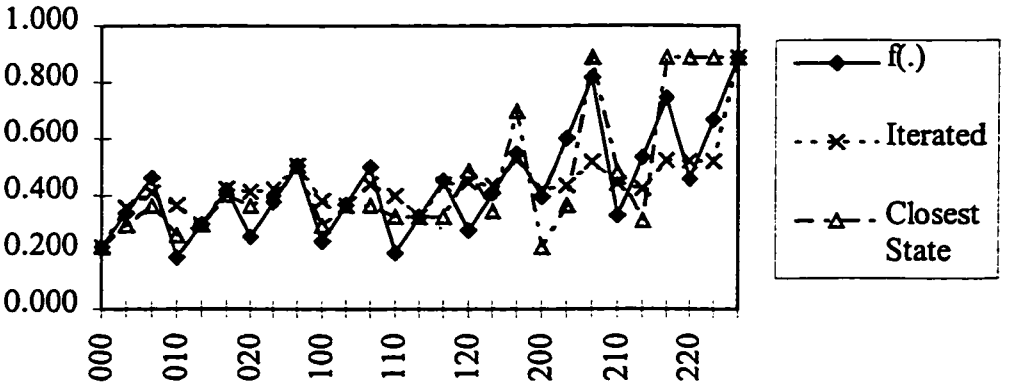


Figure 16: Added States 022, 101, and 011

states. One final note about the iterated version is that while it seems reasonable that the iterated version will converge, this has not been proven. The imputed values of the missing states are obviously bounded by the maximum and minimum values for the known states and it seems likely that the iterated version should converge for all states to values near the mean.

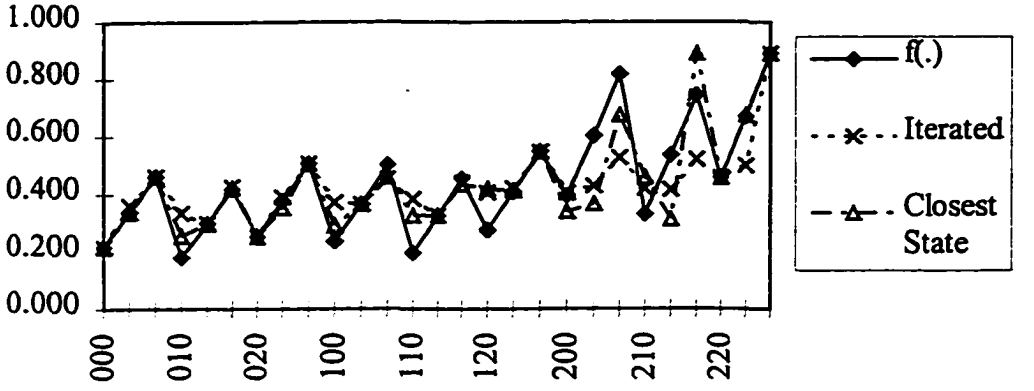


Figure 17: Added States 122, 002, 220, and 020

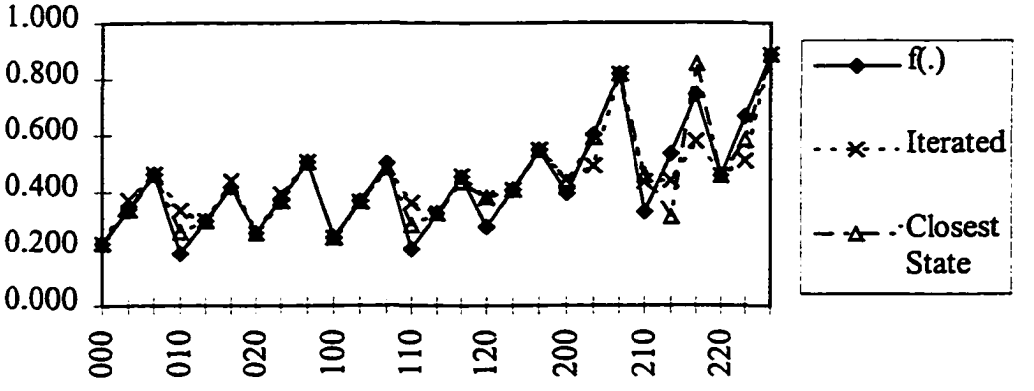


Figure 18: Added States 121, 100, and 202

2.5.2 Mean Deviation from the Mean of the Closest States

The preceding examples for both the iterated and non-iterated versions of the closest states algorithm indicate that quite often the mean of the closest states is offset from the known value. In practice, it is impossible to know what the value of the missing state is, but this observation suggests another alternative form of the closest states algorithm.

The members of the set of closest states also have a set of closest states. For the members of the closest state set that have a known value, the closest state sets for these members can be determined and the mean of these sets can be calculated. This calculated mean can then be compared to the known value and the deviation of the prediction from the known can be directly calculated. The deviations of all the members in the set of closest states can be calculated and the mean deviation of the closest state set can be determined. This mean deviation for the closest state set can then be used to adjust the calculated mean for the missing state.

The following results are calculated for a preceding example system that is shown in figure 12 and starts with the following missing states: 001, 012, 101, and 111. Starting the same as previously, each of the closest state sets is formed for the missing states.

$$A_c^{001} = \{101, 011, 000, 002\}$$

$$A_c^{012} = \{112, 002, 010, 011\}$$

$$A_c^{101} = \{001, 111, 100, 102\}$$

$$A_c^{111} = \{011, 101, 110, 112\}$$

The difference in this algorithm begins with finding all of the closest state sets for all of the states in each of the sets listed above. There is some redundancy in the members of the closest state sets, so the table in figure 19 shows each unique state set only once. Note that each closest state set has four members, since there are two distinct values for v_1 and v_2 and three values for v_3.

States	Closest States			
000	100	010	001	002
001	101	011	000	002
002	102	012	000	001
010	110	000	011	012
011	111	001	010	012
100	000	110	101	102
101	001	111	100	102
102	002	112	100	101
110	010	100	111	112
111	011	101	110	112
112	012	102	110	111

Figure 19: Closest States to the Members of the Closest State Sets

The mean for each of these closest state sets and the deviation of this mean from the known value can be calculated. These deviations for each of the states in the closest state set can then be averaged and this value can be added to the calculated mean of the closest state set. This technique may capture the remote behavior of the system relative to each of the states and then weight that behavior when imputing values for the missing states. The following table displays the results of the calculations of the means and the deviations.

These results can then be used to adjust the imputed values from the standard closest state algorithms based on the average deviation for each missing state.

$f(001)$ = closest state mean + the average deviation

$= 6.8 + 0.461 = 7.261$

$f(012) = 9.1 + 0.767 = 9.867$

$f(101) = 7.0 + (-0.909) = 6.091$

$f(111) = 8.9 + 2.783 \quad = 11.683$

Missing State 001	Known	Mean	Deviation	Average Deviation
101		6.95		
011	7.1	3.7	3.4	
000	3.7	5.75	-2.05	
002	9.5	9.467	0.033	0.461
Missing State 012	Known	Mean	Deviation	
112				
002	9.5	9.467	0.033	
010	3.7	4.833	-1.133	
011	7.1	3.7	3.4	0.767
Missing State 101	Known	Mean	Deviation	
001				
111				
100	3.7	5.867	-2.167	
102	10.2	9.85	0.35	-0.909
Missing State 111	Known	Mean	Deviation	
011	7.1	3.7	3.4	
101				
110	3.7	7.8	-4.1	
112	16	6.95	9.05	2.783

Figure 20: Deviations from the Mean

The following table summarizes the results and compares them to the standard closest state results. For this example, the results are comparable and possibly better from using the mean deviation as an adjustment to the closest state algorithm. It has not been determined, nor is it clear whether this version of a closest states algorithm is superior to the previous versions. Additional research is required to make any determination, but the next section provides a method for selecting an imputation method.

v_1	v_2	v_3	$f_{dev}(\alpha)$	$f_C(\alpha)$	$f(\alpha)$
0	0	1	7.3	6.8	6.1
0	1	2	9.9	9.1	14.5
1	0	1	6.1	7.0	6.4
1	1	1	11.7	8.9	8.4

Figure 21: Comparison of entropy fill and closest states

2.5.3 Test for Selecting an Imputation Algorithm

While the closest states algorithm is the most general in that it makes the fewest assumptions and shares the most information with the missing states, it is possible that for specific instances an alternative algorithm may provide superior results. This section provides a test to determine which of the algorithms may generate the best results for a specific set of data. The test proceeds by imputing values for states in the system that are already known, calculating the sum of squares error based on the known value. This procedure may be repeated for each of the algorithms defined previously, a variant of the closest state algorithm, or any other procedure for imputing data that the analyst may choose. The algorithm that has the minimum error will be selected to impute values for the missing states. While this test is not foolproof, it may provide some guidance for deciding what technique to use for imputing the data.

The test may be stated as follows:

For each imputation algorithm Alg_j and a given g-system as previously defined, determine the set of existing states A_e. For each state, α_i, in A_e, impute a value, $\hat{\alpha}_i$, using algorithm Alg_j. Calculate the error for each state as:

$$e_i = \alpha_i - \hat{\alpha}_i, \tag{2.12}$$

and the total sum of squares error for the algorithm as

$$E(A\lg_j) = \sum_{\alpha_i \in A_e} e_i^2 . \qquad (2.13)$$

The minimum value of $E(Alg_j)$ indicates that algorithm j had the minimum error for predicting function values for existing states. This may indicate that algorithm j is the optimal algorithm to use to impute values for the missing states.

Chapter 3. Incomplete Systems: Data Scattering and Clustering

The closest states algorithm allows imputation of values for missing states regardless of the number of missing states. Even if the extreme case of having a single state is considered, the closest states algorithms will provide values for the missing states. In this case, the value is the same value as that of the single existing state. The solution that the algorithm provides in this case is a reasonable answer, but there are other special cases and reasons why applying the closest state algorithm may not be the best approach. As the discussion in chapter 1 indicated, it is easy to conceive of the problem of scattered data as being the same as missing data. The discussion also included the possibility that the missing states are not missing, but that the missing data is due to some form of scattering.

Previously, the problem of an incomplete system was first viewed as a data scattering problem and then, once this problem was resolved, as a problem of missing data [JONE 85e]. As was discussed above, these are not necessarily two separate problems, but two alternate interpretations of the same problem. It is perhaps more general to initially view an incomplete system as first missing data and then, if necessary, regard the data as scattered if there is too much data missing. In chapter 4, two criteria for deciding if there is too much missing data will be presented. Assuming that the given variable values are scattered led to the solution of applying one dimensional clustering techniques to each set of variable values that were determined to be scattered. One limitation of this approach is that the clustering is

performed on the variable values alone, in effect, assuming that the variable values and the system function values are independent.

Given the closest state algorithm which is capable of taking any incomplete system and imputing a complete system, there are still reasons for using some clustering technique on the variable values. First, the true reason that the system is incomplete may indeed be that the variable values are scattered. Frequently, in the execution of a designed experiment, the control of the variable values may not be fine enough so that every trial uses exactly the same variable setting. Obviously, any analysis of the results of the designed experiment would want to take into account this fact. Second, clustering the variable values may help in reducing the number of missing states in the data set. Suppose that the system under consideration is the example that was previously considered and is reproduced here as figure 22. If the system is treated as having missing states, there are a very large number of possible states missing. In the example, there are nine unique values for v_1 and v_2, and ten unique values for v_3. This yields a total of 810 possible states, so with twelve states defined in the table, there are 798 missing states.

Obviously, even with a good algorithm for imputing missing function values for all of the missing states, it is not reasonable to think that this small percentage of existing states will provide enough information to generate good values for the missing states. In addition, it seems obvious from an inspection of the table that the values may very well be reasonably clustered; in this case visual inspection provides a good solution as was described previously. The need for clustering for K-systems

analysis is obvious, but the question remains as to how the clustering should be done and that is the question that we propose to answer here.

v_1	v_2	v_3	$f(.)$
7.2	3.1	9.4	3.7
6.9	2.9	6.2	6.1
6.7	2.7	7.1	9.5
7.1	9.2	9.2	3.7
6.9	8.9	5.9	7.1
7.5	8.6	6.9	14.5
2.8	3.3	9.3	3.7
3.1	2.9	5.7	6.4
3.2	3.0	7.0	10.2
2.7	8.7	9.2	3.7
2.7	8.9	6.3	8.4
2.7	9.2	7.1	16.0

Figure 22: Data Scattering Example

First, we propose a method for performing clustering of the variable values that takes in account the system function value. The clustering method discussed here will be focused on the needs of K-systems analysis, specifically, the need to reduce the amount of data scattering and the number of variable values so that K-systems analysis may proceed. A general method will be described that will be allow K-systems analysis to proceed utilizing any type of clustering algorithm.

3.1 Using Clustered Data in K-systems Analysis

K-systems analysis may be performed and the results are valid for the given data system independent of the underlying system. In the same way that the K-system function must be induced from a general system function, the variables and the values that they take must be in the correct form for the analysis to proceed. It is desirable to

have each variable assigned a state set of the form of the natural numbers, $V_i = \{0, 1, \ldots, n\}$. This can be easily accomplished given a finite set of data by assigning a unique integer value to each distinguishably unique variable value.

Current methodology for using clustered data in K-systems analysis requires that the clustering be done in one dimension. Each set of variable values is clustered independently of all other variables and the system function. This leads to a problem in the use and interpretation of the cluster values that are used in place of the original variable values. This problem is not one of relabeling the values since the relabeling results in an isomorphism. The problem is related to the meaning of the clusters that are submitted for analysis. Generating clusters independently of the system function assumes that the system function is independent of the variable values; the variable values do not effect nor control the system function. The reason for doing any type of data analysis is to find what the effect that the variable values have in determining the behavior of the system. This is especially true in K-systems analysis because not only are the variable effects being sought, but even finer effects may be discovered through the states and substates.

We propose that the clustering be done across two dimensions, a variable and the systems function, in what is effectively a preprocessing step to K-systems analysis. The preceding discussion made it obvious that clustering based solely on the variable values is insufficient and the results from the analysis may be ambiguous. If we are to propose doing clustering using two dimensions, it may seem a natural extension that the clustering proceed across all of the variables and the system function. The true

purpose of doing the clustering for K-systems analysis is to induce a system that has properties sufficiently close to a probabilistic system that the probabilistic reconstructability analysis algorithms can be applied. In effect, the K-systems analysis is a form of clustering or unsupervised learning in that it finds the important subsystems that effect system behavior and is able to group variables and their values in different combinations that make the controlling subsystems obvious.

Limiting the clustering to the two dimensions of each variable and the associated system function values allows the results to be submitted to the finer analysis of K-systems analysis. As in the closest states algorithm, the system function provides the meaning of the variable clusters. Clustering generally proceeds across all dimensions of the data and groups like data vectors together. K-systems analysis also proceeds across all dimensions in the data, but it is able to produce more refined groupings of the data; it finds categories of effects based on all possible combinations of values and distributes their effects though maximum entropy mathematics. Clustering for each variable in combinations with the system function allows the high level behavior of the system to be found. This can then be submitted for K-systems analysis so that the finer effects of the variable clusters and combinations of variables clusters on the overall system can be found.

One final aspect of clustering for K-systems analysis is the use of these two dimensional clusters and, in particular, using the results of the analysis to make predictions of the effect of previously unknown variable values. Given a reconstruction of a system, the analyst may wish to predict the system function output

based on some combination of specific variable values. The variable values must be mapped into one of the existing clusters and this must be done in the absence of a system function value. If the existing clusters are not linearly separable perpendicular to the axis of the variable, the variable value being considered may not map into a single unique cluster. This means that the variable value can not be unambiguously assigned to a single cluster and there is not a single prediction about the effect that the variable values will have on the system. Instead of being a drawback, this can add new insight in the analysis of the system. The variable value can be determined to have some probability of inclusion into multiple clusters based on whether it falls in the range of the cluster and on the distance of that value from the centroids of the clusters. While this does not provide a single solution to the effect that the variable values will have, it will provide a set of possible solutions along with associated probabilities based on the possible inclusion of the variable values in the existing clusters. This extends the K-systems analysis from a method of providing deterministic answers for every possible state, into a stochastic type of modeling system that will provide a number of solutions along with associated qualifications based on the possibility that specific values fall into specific clusters.

For each two dimensional cluster generated by any clustering algorithm, the following information is generated by, in effect, projecting the two dimensional cluster to the variable axis. So, each cluster will have maximum and minimum variable values, v^c_{max}, v^c_{min} and a centroid or mean value, $\bar{v}^c$ of the cluster can also be calculated. Given a specific variable value, v, it must first be determined whether this

value falls into the range of any of the existing clusters. If it falls into one or more, then the probability that this values falls into one of these specific clusters can be calculated as:

$$\Pr(v \in c_i) = \frac{\left|v - \bar{v}^{c_i}\right|}{\sum_{\forall c} \left|v - \bar{v}^{c}\right|}. \tag{3.1}$$

If the value does not fall inside the range for any of the clusters, it may be assigned to a single closest cluster or the probability may be calculated for all clusters.

Once these probabilities have been calculated they may be used to qualify the predictions that the K-systems analysis would provide for specific cluster assignments. Specifically, these probabilities may be used in one of two ways. First, the analyst may simply select a single cluster where $\Pr(v \in c_i)$ is a maximum. Alternatively, all possible clusters and combinations of clusters may be used to generate predictions and the product of the probabilities for each cluster can be used as qualifiers. In effect, the analysis is stochastic and provides multiple possible solutions that are qualified by the product probabilities.

3.2 Entropy Similarity

Clearly some kind of clustering may be necessary for K-systems analysis to be effective on the widest possible range of systems. It is also quite clear that clustering independently for each variable may lead to results which are ambiguous. The previous section described in some detail the problems that may be encountered and outlined a general methodology whereby they may be overcome. There remains a

question about the details of how best to determine the clusters that should be submitted for K-systems analysis.

Any common clustering algorithm may be applied to the problem of determining the clusters. While all of these algorithms are useful is certain contexts, none were developed specifically for application in the field of reconstructability analysis. Here we present an algorithm that is directly based on information entropy and uses the same techniques for inducing a K-system to induce a system that may be analyzed using information entropy.

The algorithm that will be developed here is based on the taxmap algorithm that was developed by Carmichael et al. [CARM 68] and Carmichael and Sneath [CARM 69]. This algorithm attempts to imitate the procedure that is used by a human who is manually detecting clusters through observation of two and three dimensions. This algorithm tries to detect relative distances between pairs of points and searches for continuous and relatively dense regions of space that are surrounded by mostly empty space. This method is based on the use of a similarity matrix that contains the relative similarity of all pairs of points. In general, the matrix values are based on some general method of similarity, but the specific similarity function or relation is not explicitly defined. This section will develop a measure of similarity that is based on the formalities of information theory and the entropy mathematics used for K-systems analysis.

We begin by defining the pairwise similarity measure using the information entropy. First, consider the equation for the joint entropy of two independent variables [SHAN 48]:

$$H(x,y) = -\sum_{\forall x,y} p(x,y)\log_2 p(x,y) \tag{3.2}$$

where $p(x,y)$ is the joint probability distribution. The maximum value of the entropy will be when the distribution, $p(x,y)$, is uniform. This is the basic equation that will be used to calculate the similarity of the each pair of points. Note that the similarity measure will be defined for two dimensional space, but may be extended for multiple dimensions.

Given two points the question is then how the joint probability distribution can be calculated based solely on these two points. A joint distribution can be calculated given the marginals of a system and this distribution will be the maximum entropy distribution given that the two marginal distributions are independent. If the two marginal distributions are not independent, the resulting distribution calculated as below is then one solution in a family of distributions that satisfy the constraints of the joint distribution. For example, suppose that we have the two discrete marginal distributions $p(x) = \{0.25, 0.30, 0.45\}$ and $p(y) = \{0.30, 0.40, 0.30\}$. The problem may then be set up in the tabular format of a joint probability distribution as shown below.

X\Y	1	2	3	
1	$p(x_1,y_1)$	$p(x_1,y_2)$	$p(x_1,y_3)$	0.25
2	$p(x_2,y_1)$	$p(x_2,y_2)$	$p(x_2,y_3)$	0.30
3	$p(x_3,y_1)$	$p(x_3,y_2)$	$p(x_2,y_2)$	0.45
	0.30	0.40	0.30	

Figure 23: Joint Probability Distribution

Given this distribution, the joint distribution can be directly calculated as $p(x_i, y_j) = p(x_i)p(y_j) \forall i, j$. This results in the distribution shown in figure 24:

X\Y	1	2	3	
1	0.075	0.100	0.075	0.25
2	0.090	0.120	0.090	0.30
3	0.135	0.180	0.135	0.45
	0.30	0.40	0.30	

Figure 24: Maximum Entropy Distribution

Based on this distribution, the entropy of the system can be directly calculated from equation 3.2 and in this particular case, $H(x, y) = 3.11$. While this allows the calculation of the joint distribution and the overall joint entropy of this distribution, it still does not lead us directly to the similarity measure. It shows that given the marginals of a system, it is possible to calculate the joint maximum entropy distribution given the marginal distributions.

Next, we show how to induce the properties of marginal distributions directly from a pair of data points. The technique that is used is similar to the transformations that results in a K-system [JONE 85c]. Again, we note that the transformation used here is for two dimensions, but that this transformation can be readily extended to multiple dimensions. Suppose that we are given two data points, (x_1, y_1) and (x_2, y_2), such that $x_i, y_i \in R^+$. Without loss of generality, we restrict these points to the positive real numbers because any points outside this range can be easily mapped into the range of positive real numbers. We define the following transformations factors:

$$\tau_x = x_1 + x_2 \tag{3.3}$$

$$\tau_y = y_1 + y_2. \tag{3.4}$$

These factors can then be used to transform the points as follows:

$$x_1' = \frac{x_1}{\tau_x} \tag{3.5}$$

$$y_1' = \frac{y_1}{\tau_y} \tag{3.6}$$

$$x_2' = \frac{x_2}{\tau_x} \tag{3.7}$$

$$y_2' = \frac{y_2}{\tau_y}. \tag{3.8}$$

Clearly, the resulting points will have the following properties of marginal probability distributions:

$$\sum_{\forall i} x_i = 1.0,\ 0 \le x_i \le 1 \text{ and}$$

$$\sum_{\forall i} y_i = 1.0,\ 0 \le y_i \le 1.$$

Note that information has neither been added nor removed by this scaling of the variable values. The results of the transformations are not probability distributions, but has created a system that has sufficient properties that they can be used in the same manner as marginal distributions.

The entropy of two points given the transformation and the use of the points as marginals can then be calculated based on the joint distribution derived from the marginals as follows. Given two arbitrary points in two dimensional space , (x_1, y_1) and (x_2, y_2), can first be transformed as in equations 3.5 through 3.8 into the points, (x_1', y_1') and (x_2', y_2'). The joint maximum entropy distributions can then be formed as

if the these transformed points were marginals. The tabular form of the problem is shown below:

X\Y	1	2	
1	$p(x'_1, y'_1)$	$p(x'_1, y'_2)$	x'_1
2	$p(x'_2, y'_1)$	$p(x'_2, y'_2)$	x'_2
	y'_1	y'_2	

Figure 25: Entropy Similarity Joint Distribution

Note that $p(x_i, y_j)$ is equal to the product ($x'_i y'_j$).The entropy of this distribution can then be calculated as defined by the equation for information entropy, equation 3.2.

Note that the maximum value for the two dimensional joint entropy is when the distribution is uniform. This implies that the scaled values must all be equal and, since they must sum to one, all values of $p(x'_i y'_j)$ must be equal to 0.25. This, in turn, implies that the maximum value that the entropy can take is equal to 2.0. This value will be used to normalize the entropies calculated to the interval [0,1]. The normalized entropies will then be able to serve as a measure of similarity between two points and will conveniently fall within the [0,1] interval.

In general, this normalization factor is dependent on the number of dimensions and the number of points that are being assessed in the joint entropy. We will derive an expression that can be used to determine this normalization factor for *n*-dimensional points. As was stated previously, the joint entropy is at a maximum when the distribution is uniform. Let n_d stand for the number of dimensions (or variables) in the data. Since the similarity involves two points, this means that the distribution for each marginal can be expressed as

$$p(v_i) = \frac{1}{n_p} \tag{3.9}$$

where n_p is the number of points to be compared and in the pair wise comparison, n_p = two. Therefore, each portion of the joint distribution would be expressed as

$$p(v_1, v_2, \ldots, v_{n_d}) = \frac{1}{(n_p)^{n_d}}. \tag{3.10}$$

The n_p in the denominator is due to the fact that this comparison is applicable for n_p number of the points and the power of n_d is due to the number of marginals that make up the joint distribution, so we have two raised to the number of dimensions as the number of partitions into which the joint distribution is divided. Note that if this equation is used for a general comparison of 2 data points, $n_p = 2$. The distribution is then used in the calculation of the joint entropy so that,

$$H(v_1, v_2, \ldots, v_{n_d}) = -\sum p(v_1, v_2, \ldots, v_{n_d}) \log p(v_1, v_2, \ldots, v_{n_d}). \tag{3.11}$$

The summation occurs over all combinations of the n_d dimensions for each variable which yields the upper limit for the summation of $n_p^{n_d}$. Substituting equation 3.10 into equation 3.11 yields,

$$H(v_1, v_2, \ldots, v_{n_d}) = -\sum_{i=1}^{n_p^{n_d}} \frac{1}{(n_p)^{n_d}} \log \frac{1}{(n_p)^{n_d}}, \tag{3.12}$$

where there are $(n_p)^{n_d}$ summations so we have,

$$H(v_1, v_2, \ldots, v_{n_d}) = -\left(\frac{1}{(n_p)^{n_d}} \log \frac{1}{(n_p)^{n_d}} \right) n_p^{n_d} \tag{3.13}$$

Which finally yields the maximum value of the entropy for distributions that have n_d dimensions,

$$n_H = -\log_2\left(\frac{1}{(n_p)^{n_d}}\right), \tag{3.14}$$

For the pairwise comparison of data points, n_p = 2 and the normalization factor simplifies to:

$$n_H = n_d . \tag{3.15}$$

So we find that the pairwise normalization factor, n_H is equal to n_d, the number of dimensions.

Given the equations set forth above, we can explicitly derive equations for the two dimensional entropy similarity measure as follows. We begin with two points (x_1, y_1) and (x_2, y_2), which are then scaled to the points (x_1', y_1') and (x_2', y_2') as in equations 2.2 through 2.7. The joint maximum entropy distribution can then be calculated resulting in the values for p_{ij}. The value for two dimensional entropy can then be explicitly calculated as:

$$H(x,y) = -\begin{pmatrix} p(x_1,y_1)\log_2 p(x_1,y_1) + p(x_1,y_2)\log_2 p(x_1,y_2) \\ + p(x_2,y_1)\log_2 p(x_2,y_1) + p(x_2,y_2)\log_2 p(x_2,y_2) \end{pmatrix} \tag{3.16}$$

We define the two dimensional entropy similarity, S_H^2, to be:

$$S_H^2(p_1,p_2) = \frac{H(x,y)}{2}, \tag{3.17}$$

where p_1 and p_2 are two points in 2 dimensional space the 2 in the denominator is to scale the results to the [0,1] interval as described above. In general, the entropy can be defined as in equation 3.11 and we can now provide a corresponding general equation for the entropy similarity for any dimension, n_d, as follows:

$$S_H^{n_d}(p_1, p_2, \ldots, p_{n_d}) = \frac{H(v_1, v_2, \ldots, v_{n_d})}{n_H} \tag{3.18}$$

where n_H is defined as in equation 3.14. We may now use this as a measure of similarity, form a similarity matrix and apply a taxmap style clustering algorithm. First, we will present some examples to help characterize the behavior of this function.

The following plot will show the points for the examples that follow.

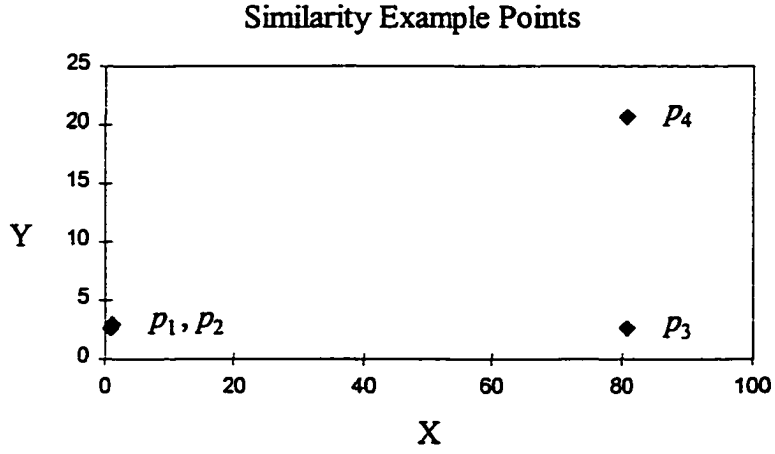


Figure 26: Plot of Example Points

Suppose the we are given the two points p_1 = (0.8, 2.7) and p_2 = (1.1, 3.0). Upon inspection of figure 26, it is obvious that these two points are very similar. Transforming these points as specified in equations 3.3 through 3.8 we calculate:

$\tau_x = 1.9, \tau_y = 5.7$ and

$$x_1' = \frac{x_1}{\tau_x} = \frac{0.8}{1.9} = 0.421,\ y_1' = \frac{y_1}{\tau_y} = \frac{2.7}{5.7} = 0.474$$

$$x_2' = \frac{x_2}{\tau_x} = \frac{1.1}{1.9} = 0.579,\ y_2' = \frac{y_2}{\tau_y} = \frac{3.0}{5.7} = 0.526.$$

Using these values as marginals yields the following distribution,

X\Y	1	2	
1	0.199	0.222	0.421
2	0.274	0.305	0.579
	0.474	0.526	

Figure 27: Entropy Distribution

Calculating the entropy as in equation 3.16 yields *H(x,y)* = 1.9799 and normalizing from equation 3.17 yields $S_H^2(p_1, p_2) = 0.990$. This agrees with our intuition that the two points are very similar to each other.

Using the previous example as a starting point, suppose that we are given the two points, p_2 = (1.1, 3.0) and p_3 = (80.8, 2.7). Inspection yields the observation that these points are not very similar, in general, but they are very similar along one dimension, x_1 = 2.7 and x_2 = 3.0. Calculating the similarity as above yields, $S_H^2(p_2, p_3) = 0.550$. This value captures the fact that while the points are not very similar in terms of the distance from each other, they are very similar along on of their dimensions.

Suppose that the points are p_2 = (1.1, 3.0) and p_4 = (80.8, 20.7). The observation this time is that the points are not very similar at all. Calculating the entropy similarity yields, $S_H^2(p_2, p_4) = 0.325$. This result is fairly intuitive because

one might expect that this value should be even lower than the previous two points, where $S_H^2(p_2, p_3) = 0.550$.

Suppose that we wish to use the Euclidean distance to calculate the similarity between the respective pairs of points where the distance is defined as:

$$d_E(p_i, p_j) = \sum_{k=1}^{n_d} \sqrt{(v_{ik} - v_{jk})^2}. \tag{3.19}$$

Then the distances are $d_E(p1,p2) = 0.424$, $d_E(p2,p3) = 79.70$, are $d_E(p2,p4) = 81.64$ for each example. Using the inverse of the distance as a measure of similarity, defined as

$$s_E(p_i, p_j) = \frac{1}{d_E(p_i, p_j)}, \tag{3.20}$$

yields values of $s_E(p1,p2) = 2.36$, $s_E(p1,p2) = 0.0125$, and $s_E(p1,p2) = 0.0122$. This Euclidean measure does not greatly discriminate between the last two pairs of points, while the maximum entropy measure captures the similarity of the two points in the second example based on their similarity along one of the dimensions.

Suppose that we also consider the city block distance that is recommended for use in the original taxmap algorithm [CARM 69]. The city block distance was used because the wanted points to have the same distance if the points were, say, two units apart on each variable or if they were one unit apart on one variable and three on the other.

$$d_{cb}(p_i, p_j) = \sum_{k=1}^{n_d} |x_{ik} - x_{jk}|, \tag{3.22}$$

For the preceding points, the city block distances are $d_{cb(p1,p2)} = 0.3$, $d_{cb(p2,p3)} = 80$, and $d_{cb(p2,p4)} = 98$. Again, using the inverse as the measure of similarity

$$s_{cb}(p_i, p_j) = \frac{1}{d_{cb}(p_i, p_j)}, \tag{3.23}$$

yields, $s_{cb(p1,p2)} = 3.33$, $s_{cb(p2,p3)} = 0.015$, and $s_{cb(p2,p4)} = 0.0102$. While the city block distance does a better job of recognizing the similarity along a dimension, it still does not greatly discriminate the similarity between the pairs of points p_1, p_2 and p_1, p_3.

Suppose that we consider the final combination of pairs of points, p_3 = (80.8, 2.7) and p_4 = (80.8, 20.7). The entropy similarity between these two points is $S_H^2(p_3, p_4) = 0.758$, the Euclidean distance is $d_{E(p2,\ p4)} = 18.0$, the Euclidean similarity is $s_{E(p2,\ p4)} = 0.0556$, the city block distance is $d_{cb(p2,\ p4)} = 18.0$, and the city block similarity is $s_{cb(p2,\ p4)} = 0.055$. Again, the entropy similarity captures the fact that the points are very similar, in fact exactly the same, along one dimension. Also, it captures the same information as the Euclidean similarity, namely, that the distance between these two points is less than the preceding pairs and it yields a correspondingly greater value for similarity.

The entropy similarity satisfies some of the typical properties of similarity relations, but does not satisfy any common type of transitivity or alternatively the triangle inequality. This is not unusual since the relation is not a fuzzy membership function which is typically the subject of similarity relations [ZADE 71]. Further,

measures of similarity that produce results in the interval [0,1] have a corresponding measure of dissimilarity defined as[EVER 93]:

$$d(p_i, p_j) = 1 - s(p_i, p_j), \tag{3.24}$$

which, in this case, is symmetric and non-negative. Some dissimilarity measures also satisfy the triangle inequality, in which case these measures qualify as distance measures as was the case with the state distance in the previous section.

The entropy similarity has a corresponding dissimilarity measure defined exactly as in equation 3.24. This similarity is obviously symmetric so that $S_H^2(p_i, p_j) = S_H^2(p_j, p_i)$ and is non-negative, so the dissimilarity as defined in equation 3.24 has these properties as well. We denote the entropy dissimilarity as $d_H(p_j, p_i)$ and will show that it does not satisfy the triangle inequality. We prove this by a counter-example and then explain this behavior.

Suppose that we assume the triangle inequality is true for the entropy dissimilarity which means that

$$d_H(p_i, p_j) \leq d_H(p_i, p_k) + d_H(p_j, p_k) \forall k. \tag{3.25}$$

Suppose that we have three data points as shown in the following table and plot.

p_i	x	y
1	7	0.5
2	6	10
3	8	2

Figure 28: Triangle Inequality Counterexample

For the triangle inequality to be true, we must have

$$d_H(p_1, p_2) \leq d_H(p_1, p_3) + d_H(p_2, p_3) \tag{3.26}$$

Calculation of the similarities yields

$$S_H^2(p_1, p_2) = 0.659,\ S_H^2(p_1, p_3) = 0.931, \text{ and } S_H^2(p_2, p_3) = 0.780.$$

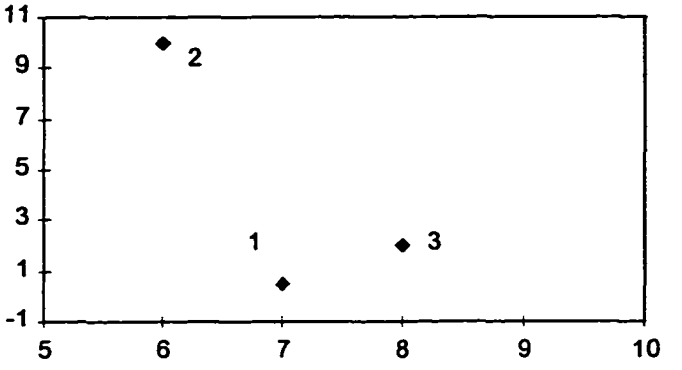


Figure 29: Plot of Counterexample

The corresponding dissimilarities are

$$d_H(p_1, p_2) = 0.341, d_H(p_1, p_3) = 0.069, \text{ and } d_H(p_2, p_3) = 0.220.$$

Substituting these values into 2.20 yields

$$0.341 \leq 0.069 + 0.220 \text{ and}$$

$$0.341 \leq 0.289,$$

which is a contradiction so we know that the entropy dissimilarity does not satisfy the triangle inequality.

Obviously, the Euclidean distance satisfies the triangle inequality and we would therefore expect to get a different ordering for these points if we compare the Euclidean and the maximum entropy dissimilarities. Calculating the Euclidean distance between these points yields the following:

$d_E(p_1, p_2) = 9.55$, $d_E(p_1, p_3) = 1.80$, and $d_E(p_2, p_3) = 8.25$.

Based on these distances we can observe that p_1 and p_3 are closest, p_2 and p_3 are next closest, and p_1 and p_2 are furthest apart. The previous results for the entropy similarity provide the following ordering: p_1 and p_3 are closest, p_2 and p_3 are next closest, and p_1 and p_2 are furthest apart; this is the same as the Euclidean ordering. We find that while the triangle inequality is not satisfied by this measure for these points, that the entropy dissimilarity still yields the same relative ordering for these pairs of points. Note that, in general, the same relative ordering of pairs of points does not hold between the Euclidean distance and the entropy dissimilarity.

The reason for this behavior is due the normalization of the points in the first step of the calculation of the entropy similarity. The similarity between the points is determined solely on their relative relation to each other and does not account for other points that may be included in the set of data. The entropy similarity and dissimilarity asymptotically approach zero and one, respectively, as the Euclidean distance increases. For this reason, once the points of interest begin to get far apart, the triangle inequality becomes false. Even though the points in the previous example maintained the same relative relationship in terms of pairwise distances, the entropy dissimilarity will fail to satisfy this ordering when compared to the Euclidean orderings.

An interesting property of the entropy similarity is that it captures the similarity of points relative to their relationship along each dimension. This means that if, say, the x-coordinate is exactly equal ($x_1 = x_2$) between two 2-D points, that the similarity between the two points will be, at least, 0.5, regardless of the values of the

y-coordinate. As the value of y_2 increases from y_1 to positive infinity, the similarity starts at 1.0 and asymptotically approaches 0.5.

The plot in figure 30 shows the relationship between the Euclidean distance and the entropy dissimilarity. The plot shows how the dissimilarity changes as the distance remains constant. The data underlying the plot is for different series of points that are exactly the same Euclidean distance from a single point. The difference between each series is their relationship to one the axes. The series labeled as along axis, is a series of points where one coordinate is the same as the source point and the other coordinate varies directly in relation to the distance. The series labeled as 0.01 degrees, has the coordinates from the first series projected so that they are oriented 0.01 degrees off the axis. The rest of the series are represented in the same way; each point is projected to new coordinates at a particular angle from the axis by the following equations:

$$x_{new} = x_0 + d_E \cos(\theta), \text{ and} \tag{3.27}$$

$$y_{new} = y_0 + d_E \sin(\theta), \tag{3.28}$$

where (x_0, y_0) is the source point from which the distance is measured and (x_{new}, y_{new}) is the new point that is a distance of d_E from the original point.

This type of relationship and bounding is found in higher dimensional entropy similarity calculations as well. For example, when n_d = 3, we find two limits for similarity at 0.667 and 0.333, when two and one coordinates, respectively, are exactly the same. The type of relationship to the Euclidean distance as shown in figure 30 will

also be maintained, but will be more complicated as it will depend upon the angle offset from two axes.

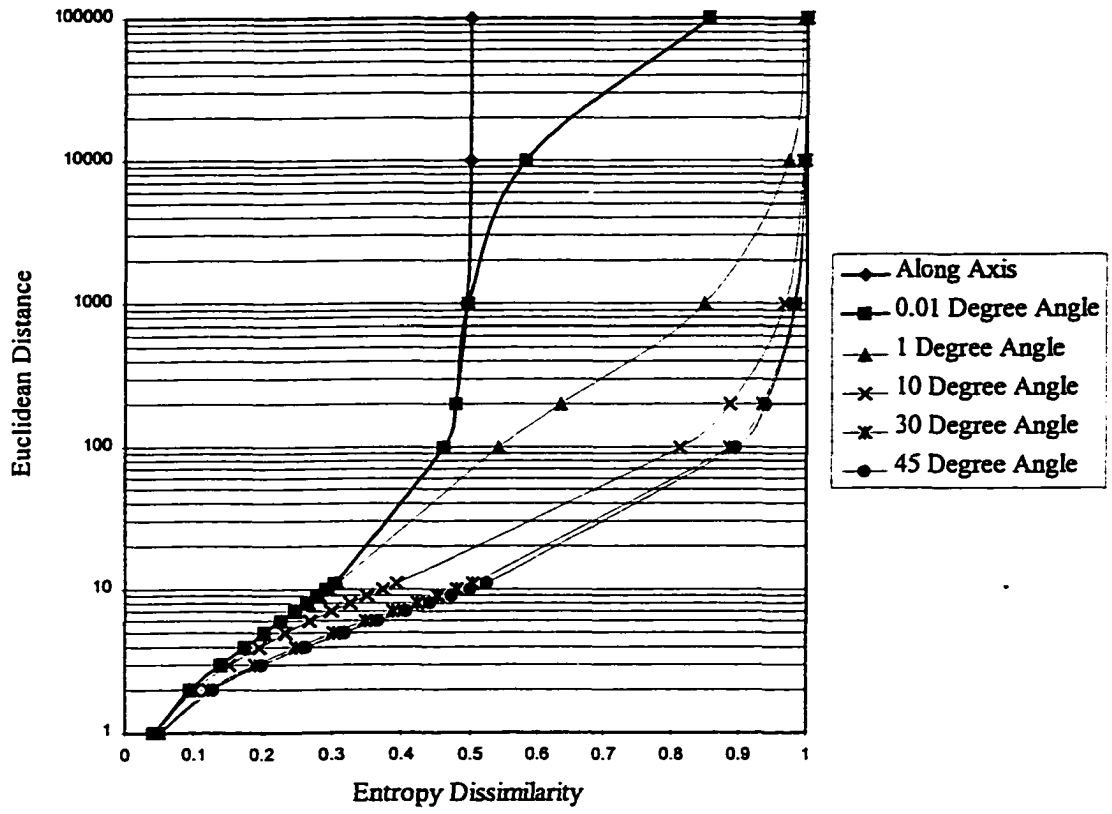


Figure 30: Relationship of Euclidean Distance to Entropy Dissimilarity

3.2.1 Entropy Similarity in the taxmap Algorithm

The taxmap algorithm was developed with the intention of mimicking the way that a human would visually detect clusters in two and three dimensions. An individual would compare relative distances between points and search for continuous and relatively dense regions of space that are surrounded by continuous relatively empty spaces [CARM 69]. The algorithm proceeds by first identifying the two most

similar points in the data and creating an initial cluster consisting of these points. Then it searches for the point that is the most similar to the points already in the cluster and considers it for admission to the existing cluster. There are some criteria for judging whether this new point should be included in the cluster. If it meets the criteria, then it is added to the cluster and if it does not meet the criteria, the current cluster is considered complete and a new cluster is begun. The new cluster is started in the same way as the first; the two closest points that are not already in a cluster initiate the new cluster.

While this algorithm may be simply stated, it requires some measure of similarity or distance and a corresponding criterion for determining whether a new point is added to an existing cluster. The original statement of the algorithm suggested the use of the city block distance function for populating a similarity matrix. The criterion for determining inclusion of a new point was somewhat arbitrary. This criterion was based on a measure of discontinuity that was derived from the change in the average similarity if the new point was added. Specifically, the drop in similarity was defined as the average similarity before the point was added minus the average similarity after the point was added. The measure of discontinuity was defined to be the average similarity after the point was added minus the drop in similarity. If this value was considered low, the point would not be added and a new cluster would be started.

The entropy similarity measure should also have a corresponding drop in similarity and a discontinuity measure for use in the taxmap algorithm. Given the

nature of this measure, the preceding definitions would not be appropriate due to the logarithm in the calculations; the similarity asymptotically approaches one. Based on the equations in the previous sections, the drop in similarity will be defined specifically for use with the entropy similarity as follows.

First, the entropy of the current cluster will be calculated by using the same technique and equations as was defined for the similarity comparison of two data points. The equations need only be modified to the extent that the 2 should be replaced by n_p, the number of points in the comparison or, in this case, the number of points in the current cluster. This enables the calculation of the overall cohesiveness of the current cluster though the use of the equations for entropy. The value for entropy can then be normalized based on the number of points based on equation this equation which is equation 3.14:

$$n_H = -\log_2\left(\frac{1}{n_p^{n_d}}\right). \tag{3.29}$$

The normalized entropy of the current cluster can be calculated, the next candidate point can be added to the cluster and the normalized entropy for the proposed new cluster can also be calculated. The difference between these two entropies can then be used as the drop in similarity due to the addition of the new cluster point. Let Δ_S be the drop in similarity, then we have,

$$\Delta_s = H_{old}\left(v_1, v_2, \ldots, v_{n_p}\right) - H_{new}\left(v_1, v_2, \ldots, v_n\right) \tag{3.28}$$

The measure of discontinuity may then be calculated similarly to the original taxmap algorithm by defining it as the drop in similarity minus the new similarity. Let M_d denote the measure of discontinuity:

$$M_d = \Delta_s - H_{new}(v_1, v_2, \ldots, v_n) \tag{3.29}$$

The measure of discontinuity is then compared to some threshold value, T, that determines whether the point is added to the current cluster ($M_d \geq$ T) or a new cluster is begun ($M_d <$ T). The value that is used for the threshold will depend on the data that is being clustered. One interpretation of the meaning of the threshold value is that is signifies the density or cohesiveness that the analyst desires in the data. The threshold may take values in the [0,1] interval and values that are close to one indicate that a high level of cohesiveness is required in the clusters. If a value if one is used, then the number of clusters will be the same as the number of unique points. If a value of zero is used, then there will only be a single cluster that contains all of the points. In general, using values which are close to one will be most effective. This is due to the manner in which the similarity rapidly approaches one as the distance between points decreases.

The taxmap algorithm is based on the idea of looking for areas that have a high density of points and forming a cluster. This leads to the behavior that the first cluster that is formed tends to be the most dense cluster in the data and the following clusters get gradually less dense. This is not true in all cases, but it indicates that the threshold value that is used for determining when to start a new cluster may need to be adjusted to a lower value as each cluster is finalized and a new cluster is started. We add an

additional parameter, ρ, to the algorithm that allows the threshold value, T, to be reduced after each cluster is formed as follows:

$$T_{new} = T_{old} - \rho.$$

While this technique will not be optimal in all cases, it was effectively used in most of the examples presented below.

One final issue related to clustering algorithms, in general, will be discussed before we present examples of the results that can be expected from the algorithm. This issue relates to determining the correct number of clusters in the data. This is often referred to a validity measure for the clustering that has been generated [XIE 91]. Many clustering algorithms such as the k-means algorithm, require the analyst to select a number of clusters that are desired and then some particular criterion or criteria are optimized to yield that number of clusters [BEZD 80]. Determining the appropriate number of clusters is done either through the analyst inspection or through the use of some validity measure. This validity measure is generally specific to the type of algorithm that is being used to perform the clustering. Usually, it consists of a calculation that tries to assess the compactness of the clusters generated and the separation between these clusters. In the original version of the taxmap algorithm, as well as the new version proposed here, the number of clusters is determined by the threshold value. Depending on the initial value and whether it is adjusted over time, different clusterings may be produced.

A method that can be applied here which is similar to a method that is used for another clustering algorithm that is based on the principle of maximum entropy, the

least bias fuzzy clustering algorithm[BENI 94]. This algorithm is very different in that it is an optimization technique, but is also has a threshold or resolution parameter. The method starts by assuming that their is initially no particular bias as to the value that the threshold should take. It proceeds by testing a finite quantized set of values from the [0,1] interval based on a quantization parameter, $1/Q$. Where Q is the number of values for the threshold that are tested in the algorithm. All values of the threshold that yield the same number of clusters, γ, are counted as $p(\gamma)$. Their fraction $P(\gamma)$ of the total Q, $p(\gamma)/Q$, is regarded as the probability that the solution to the clustering problem will yield γ clusters. Therefore, the value of γ that has the maximum value of $P(\gamma)$ is the most likely number of clusters. This same technique can be applied here with the note that the threshold values, T, may either be held constant or varied for each cluster created as was described above. In practice, it has been found that the correct number of clusters can be found without varying T, but that the quality of the clusters is not as good. So, the most probable number of clusters can be found using a constant T, but the best clusters are often found by allowing T to decrease during the creation of clusters. One final note about this technique is that when applied here, one cluster will tend to be the most frequently occurring number. This is because of the non-linearity inherent in the similarity measure. In [BENI 94], the resolution parameter is allowed to vary between 0 and 1, but for use here, it is generally started at 0.9 and uses smaller increment. Generally, the analyst will be responsible for determining whether there is a single cluster and the algorithm will determine the

optimal number of clusters starting from evaluating the measure for two or more clusters.

The preceding discussion indicated that the correct number of clusters may be found, but that the specific clustering may not be optimal. This leads to one final problem of determining the best clustering once the correct number of clusters is known. This is often done through the visual inspection of the resulting clusters and the subjective judgment of a human being. Ideally, we would have a measure of the quality of the clusters independent of the need for human judgment. This can be done in a manner consistent with the previous measure applied in the clustering algorithm. Since the clustering that is done here is for determining cluster values for each variable and the system function, the centroids of one of the coordinates (the variable) of the two dimensional clusters that were generated for the correct number of clusters can be substituted for the original values and the overall joint entropy for each clustering, H_C, can be calculated as specified previously for each clustering that yields the most probable number of clusters. The optimal clustering will be the one that has the maximum joint entropy, $H_c^{\max}$ when the substituted values are used. Note that the value of $H_c^{\max}$ is itself an indicator of the number of clusters in that it tends to reach a maximum for the optimal number of cluster. The only drawback to using is as an indicator by itself is that its true maximum is at the point where the number of clusters is equal to the number of points; the In conjunction with the method for finding the correct number of clusters, selecting a clustering that has the maximum entropy will

lead to the best clustering using the most probable number of clusters. The following section contains examples that will illustrate the use of T and H_c^{max}.

3.2.2 Entropy Similarity taxmap Examples

This section will present a number of examples of the application of the taxmap type of algorithm using the entropy similarity measures presented in the previous section. There are many types examples identified in the literature that are difficult for existing algorithms to correctly cluster. The examples presented here will all be in two dimensions and will start with simple clusterings that will be used to illustrate the effects of using different values of T and the resulting values of H_c^{max}. Following these examples will be the more difficult examples and the results that the algorithm specified here generated.

First, we begin with a simple example that is readily clustered by most common clustering algorithms and can be easily verified through visual inspection. The following four figures display the results of the algorithm with different starting values of T and show the resulting value of H_C. Note that the value of H_C increases as the value of T increases and that the clusterings get visibly better. Also note that in figure 34, the value of H_C decreased and that this resulted in three clusters.

The plot in figure 35 shows the results of varying the starting threshold, T, and the resultant number of clusters based on that starting value. These results indicate that the most probable number of clusters is two. Again, it should be noted that the threshold value will tend to generate many clusterings that have only a single cluster

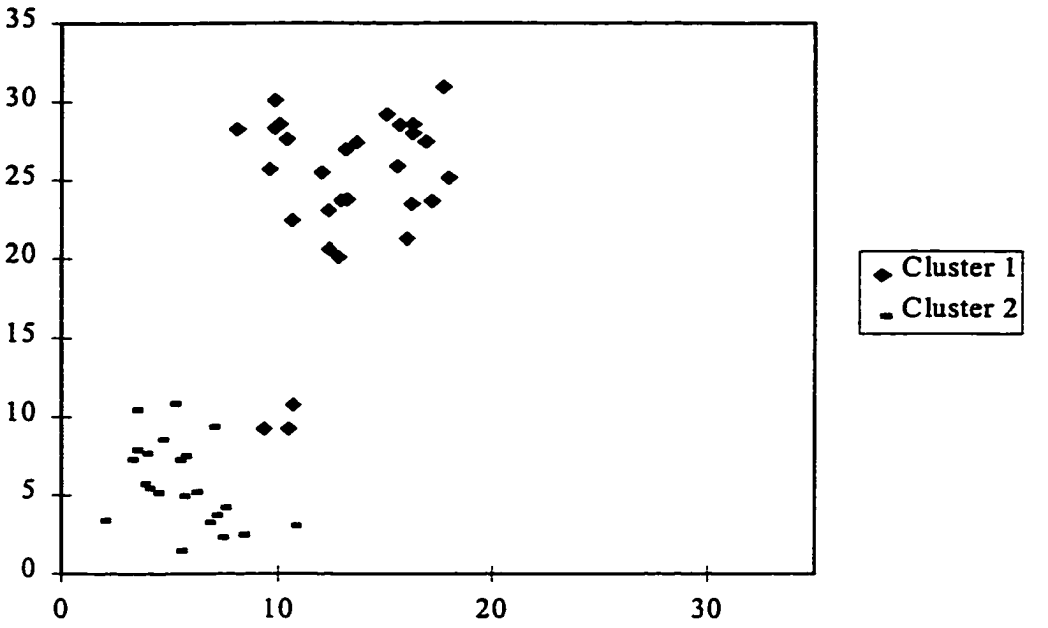


Figure 31: T = 0.990, $H_C = 0.96157$

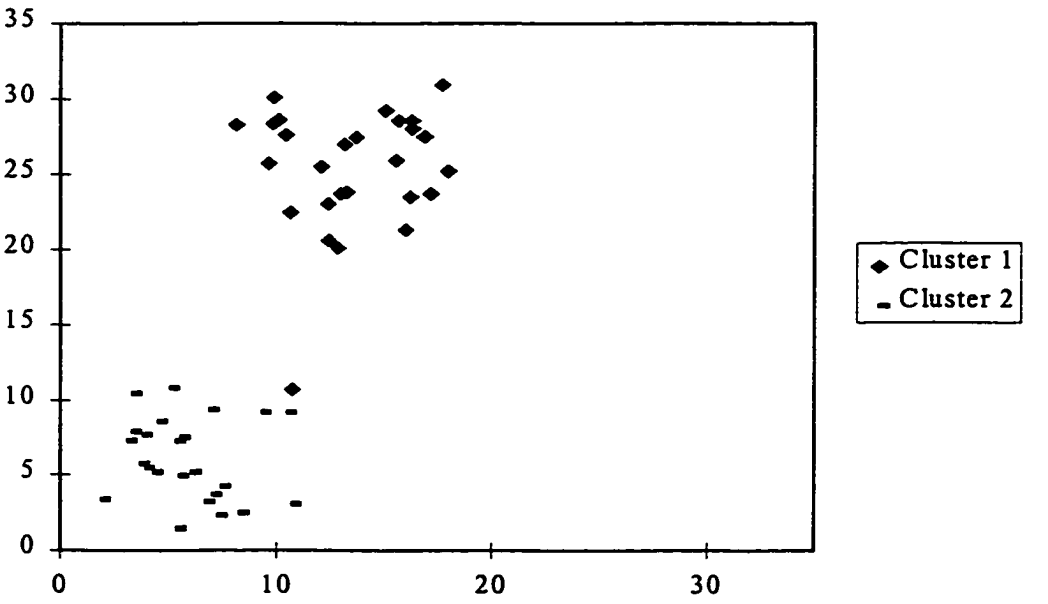


Figure 32: T = 0.993, $H_C = 0.96197$

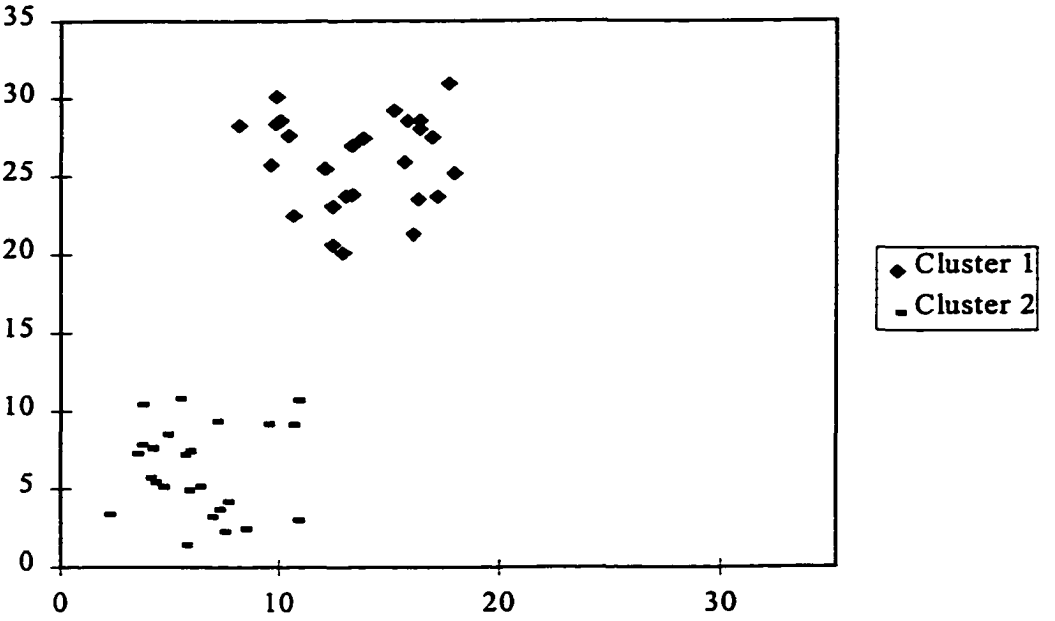


Figure 33: T = 0.994, H_C = 0.96226

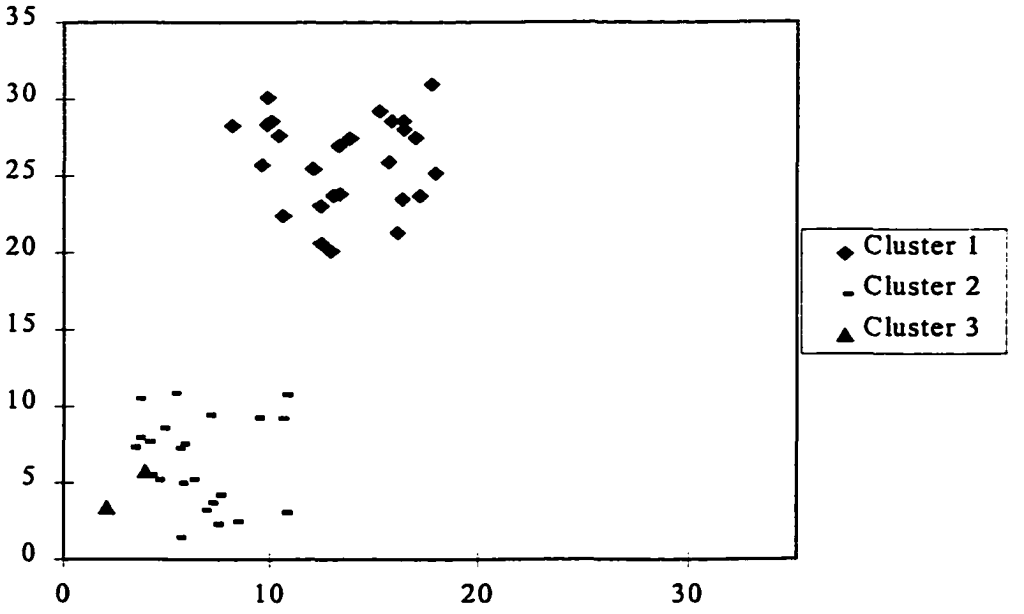


Figure 34: T = 0.995, H_C = 0.96171

and that the determination that there is more than a single cluster must be made manually.

The next example is one of the more difficult problems for clustering algorithms to solve. In consists of data that does not readily form circular clusters, which is a problem for many clustering techniques [RUSP 69].

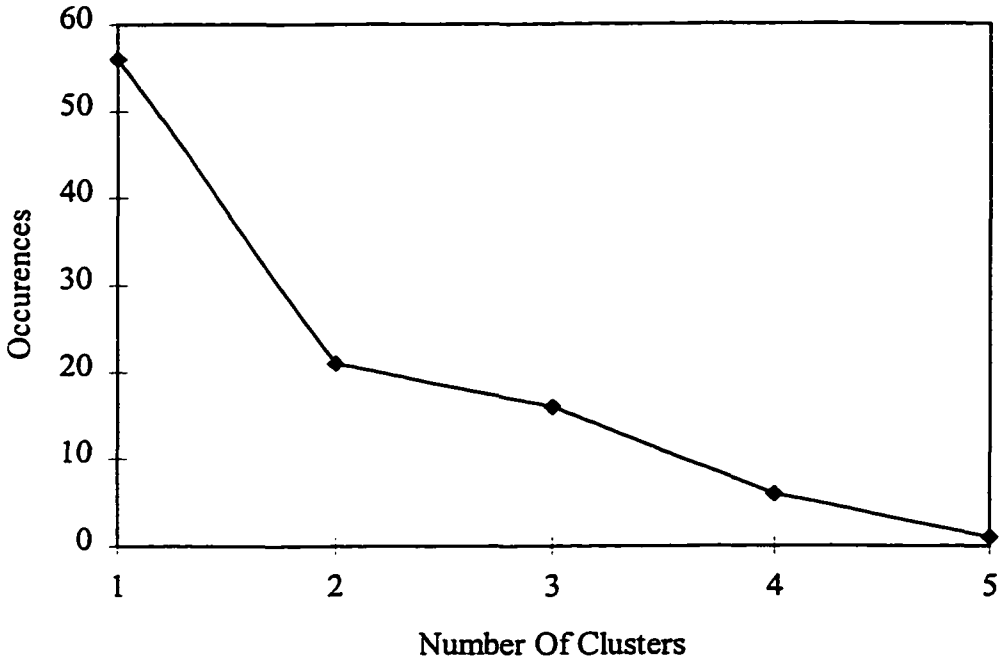


Figure 35: T = 0.90 to 1.0, incremented by 0.01

Note that the threshold value required to generate these clusters was very high. This is because the points are very similar along one dimension, in fact, the x coordinates are all exactly the same in each cluster. The only difference between these two clusters is along the y axis. Therefore, the similarity between points in different

clusters will be very high and pairs of points between clusters will be guaranteed to have a similarity of at least 0.5

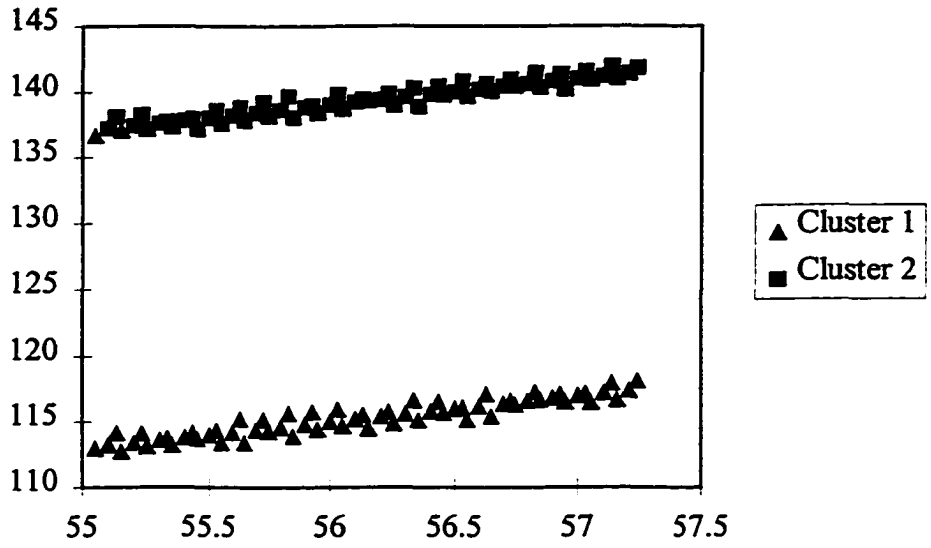


Figure 36: Non-spherical Clusters, T = 0.9999

The next example consists of two different shaped clusters. One is a very narrow and linear type of cluster and the other is a rather diffuse roughly spherical cluster.

Cluster which are linearly non-separable are also known to be problematic for clustering algorithms. The following two examples are both linearly non-separable and help to demonstrate why the taxmap method is known as a density based clustering algorithm. The first example in figure 38 is a rather sparse crescent shaped cluster that has a circular cluster contained within the arms of the crescent. The second example contains exactly the same points as the previous example, but includes additional

points in both clusters; this example has clusters that are denser than the one before. The algorithm is unable to determine good clusters for the first crescent, but is able to generate a better answer for the denser version of the same type of data.

Another problem type for clustering algorithms is when the clusters have a bridge that connects the two clusters. This makes it difficult to determine where one cluster starts and another cluster begins. The following figure shows that the

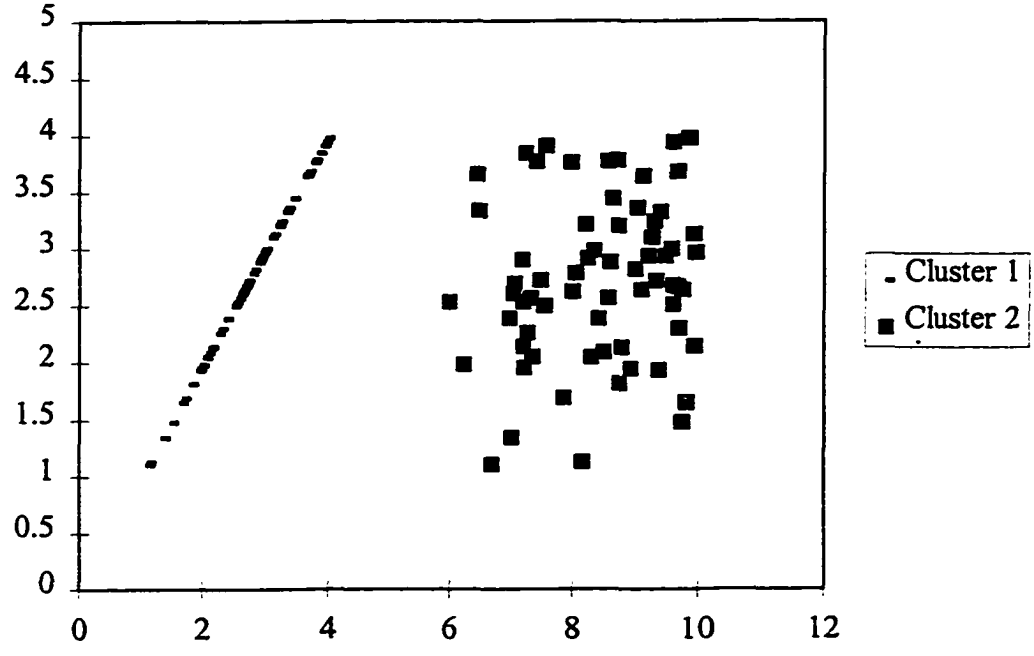


Figure 37: Clusters with Different Shapes, T = 0.99

entropy similarity measure based algorithm also has some problem with this type. While it gathers points based on the next closest point, it tends to add more points than it should; it includes points in one cluster that can be easily seen to be a part of another cluster.

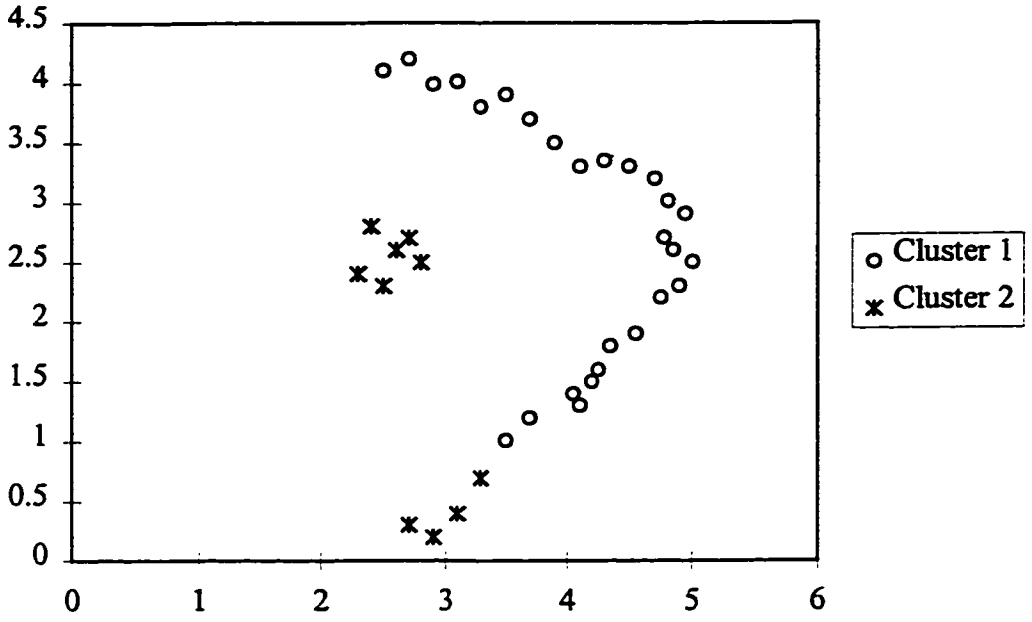


Figure 38: Sparse Linearly Non-Separable Clusters, T = 0.984

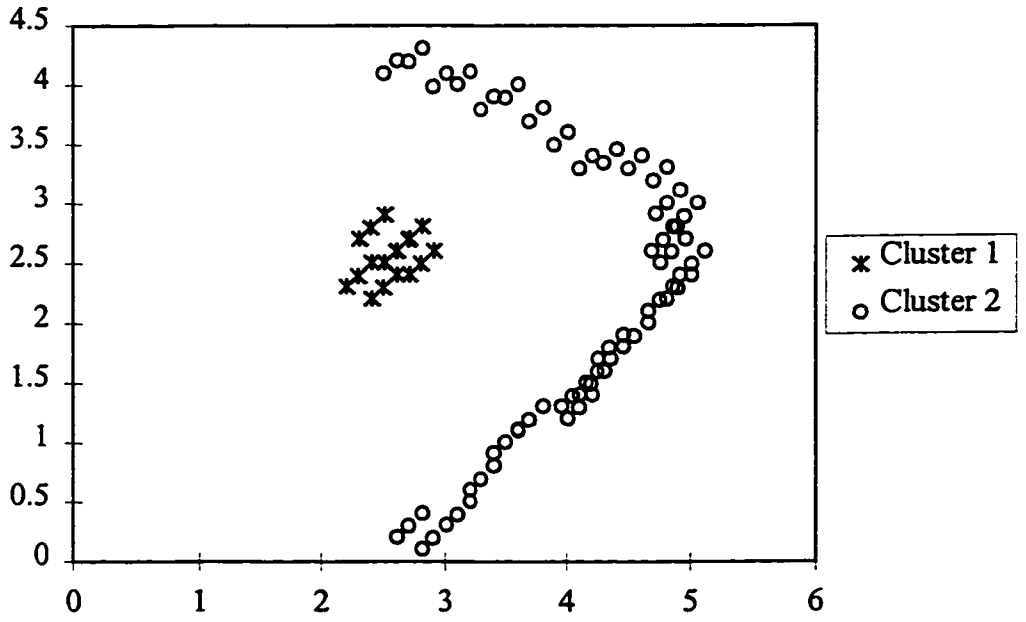


Figure 39: Dense Linearly Non-Separable Cluster, T = 0.998

The next two examples expose a limitation of using the entropy similarity measure in the taxmap algorithm. This limitation is related to the behavior of the similarity measure for points that share exactly the same coordinate. Previously, it was shown that when the two clusters include points that share a coordinate that the algorithm required a very high value of T to distinguish the clusters. As shown in figure 33, there are four clusters and each of the four clusters shares coordinates with the other clusters. The algorithm is unable to correctly distinguish the clusters, but in figure 34, the clusters have been rotated so that they no longer share coordinates. The density of each cluster and the relationship of the points within the cluster remain the same.

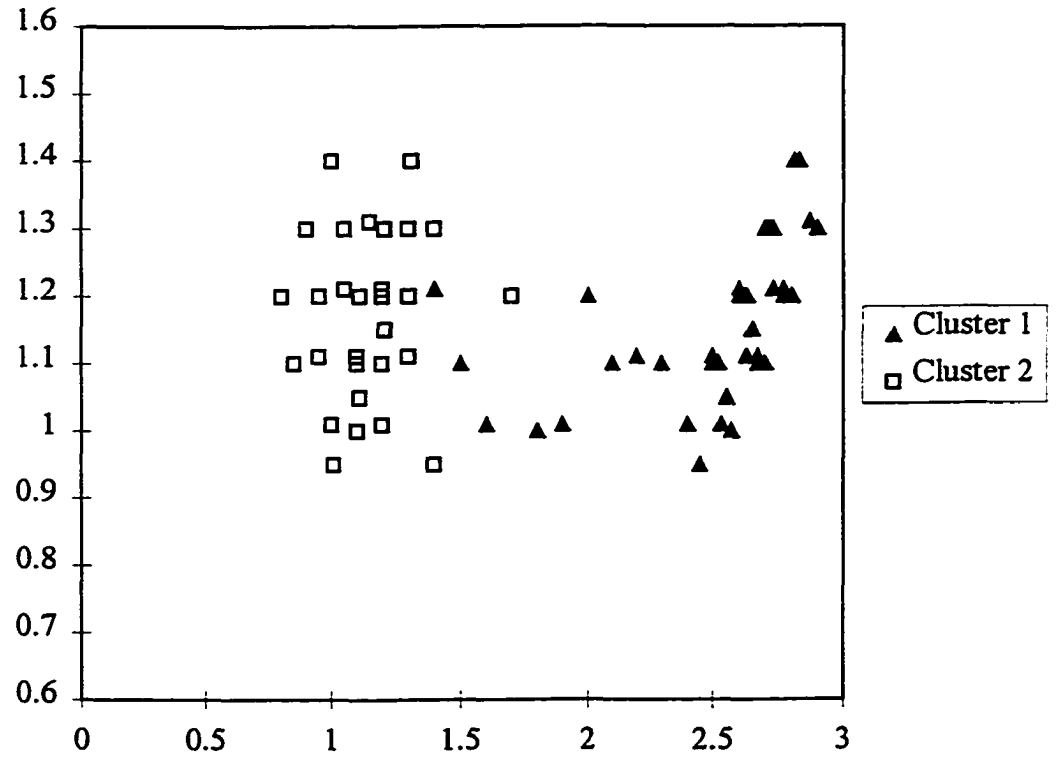


Figure 40: Clusters with a Bridge, T = 0.997

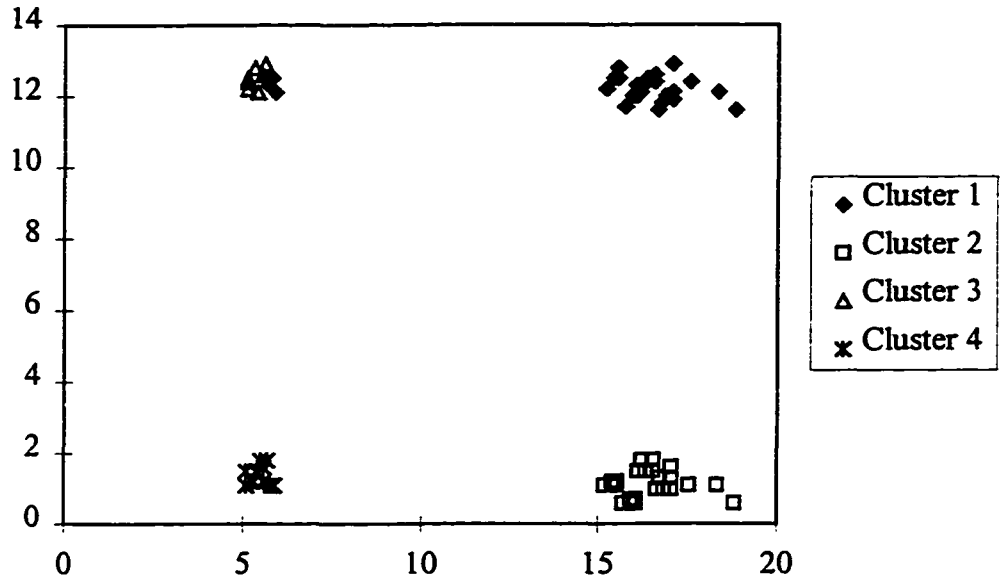


Figure 41: Four Clusters Sharing Coordinates, T = 0.991

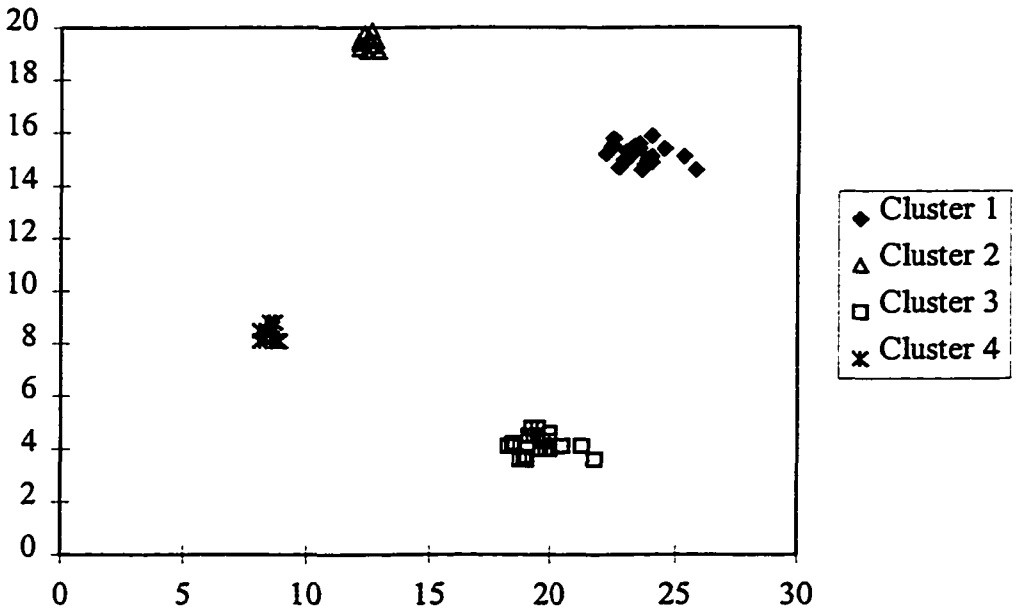


Figure 42: Four Clusters, T = 0.999

Chapter 4. Summary and Conclusions

Reconstructability analysis is a potent tool for the analysis of multivariate systems that has been applied in a wide variety of fields (for example, see [KLIR 86], [KUMA 89], [TRIV 93]). These applications were limited to small systems that were, for the most part, completely defined. In cases where the knowledge of the system was incomplete, various techniques were used to induce the complete system. These techniques include the existing techniques of the entropy fill and the one dimensional clustering, as well as, ad hoc techniques used that were based on expert knowledge of the system being analyzed. The author has presented a more complete understanding of the problems of incomplete systems and provided new techniques that yield answers that are superior to existing techniques and are consistent with the underlying theory and algorithms of K-systems analysis.

The rest of this chapter will present the preceding results as a comprehensive whole. First, the author will outline a methodology for using these new techniques to achieve results which are comprehensible and consistent with the underlying theory of K-systems analysis. Finally, future areas of research will be identified within the field of K-systems analysis and possible applications of these new techniques will be briefly explored outside the field.

4.1 A Methodology for Resolving Incomplete Systems

The understanding of incomplete systems is one of determining whether there is simply missing data or whether the variable values are scattered either through

faulty control of the variables or imprecise observation. The best resolution of an incomplete system is likely to come from a source of information that is external to the data submitted for analysis. If the analyst knows that the variable values are from a discrete set of values and that slight variations are due to imperfect control or observation, then the incompleteness of the system is known to be solely due to missing data; some of the possible states have not been observed. In effect, the clustering is done manually by the analyst with knowledge external to the data set before the data are submitted for analysis. The system can then be submitted for analysis by the missing data algorithms and the results can be directly submitted for K-systems analysis.

Alternatively, if the analyst has no prior knowledge of the system structure, determining whether the incompleteness of the data is due to missing states or scattered variable values is more problematic. Ideally, there is sufficient data so that the missing states can be addressed using the closest state algorithms, since there is no loss of information that is associated with applying a clustering algorithm. By imputing values for the missing states, all the existing information about the system is used and only minimal assumptions are made about the missing states. If clustering is performed, information, in some sense, is potentially lost by clumping together known information into a single information entity.

The general question about incomplete systems is whether the data should be considered missing and whether the data should be considered scattered. If there is no other knowledge about the source of the system data, there is uncertainty as to what

approach should be used to induce a complete system. It seems that a perfect answer for all situations is unattainable. If the system is incomplete, there must be some reason that this is so and it seems likely that this answer will be found external to the data that has been submitted. The most general answer when no external information is known is to assume that there is information about this system that is missing. Then applying the missing data algorithms will provide answers that make the fewest assumptions about the nature of the missing data. Unfortunately, it is possible that the amount of information that is missing from the system is so great that it is unreasonable to assume that any algorithm, no matter how good, could impute the missing values that truly capture the behavior of the system. In this case, attempting to group or cluster the existing data may be the best approach, especially since it may reduce the amount of missing data.

Since a completely general answer to this problem seems highly unlikely, a number of possible approaches are suggested. First, one reasonable approach is to determine whether any of the closest state sets of a missing state are empty. If so, the closest state algorithms will still provide an answer, but the answer will actually consist of estimates based on states that are not in the closest state set. We have shown that as the distance between states becomes greater, the amount of shared information becomes far less and we would expect that the estimates calculated would be proportionally more biased.

For systems that consist of a large number of variables and states, the previous approach may be too restrictive due to the large amount of data that is required about

the system. An alternative approach may be that the missing data may be imputed so long as the maximum gap between existing states is less than half of the maximum distance between possible states defined for the system. This allows closest state sets to be empty, but requires that the values used to impute the missing values share at least half of their information with the missing states.

If neither of the two previous criteria are met, it is clear that some other method for inducing a complete system must be tried. If the data are experimental and the system is relatively small, it may be possible to gather more information about the system with the express purpose of filling in some of the missing data. If the data is observational and not experimental or if the system is large and gathering more experimental data is costly or difficult, it may be impossible to gather more information about the system. For these types of cases and others, we can assume that the only information about the system is that which already exists and additional information to reduce the amount of missing data is not available; the analysis must proceed with only the existing data.

Clustering the data in some fashion may reduce the amount of states that are missing in the system. If the amount of missing data can be eliminated by clustering or reduced so that the system meets one of the previous criteria for imputing missing data, a meaningful analysis may be performed. An essential feature of the clustering is that the clusters themselves must be meaningful. Creating clusters based on the variable values themselves along with the system function assures that, at least, the clusters are relevant to the context of the system, that is the system function. These

two dimensional clusters map directly back to the original system and are easily understood to relate to the original system. The clustered data may then be re-analyzed to determine if there is still missing data and to what extent. The closest states algorithms may then be used to impute any remaining missing values and the results of the analysis may be readily mapped back to the original system. In addition, predictions of the effects of previously unknown variable values may be determined as was described previously.

4.2 Conclusions and Final Remarks

The author has proposed the use of the closest states algorithm and the entropy similarity measure as the methods for performing the imputation and clustering of the data; applied together, these techniques can be used to induce a complete system. In general, these methods use the same principles and mathematics that are already embodied within the existing K-system algorithms and techniques. They are based on well known principles that enable a consistent approach to K-system analysis. While use of other techniques yield systems that have properties sufficient for the use of the RA algorithms, these new techniques provide solutions that are either superior to previous techniques or more meaningful and more easily understood when applied within the context of K-systems analysis.

Additionally, a general methodology for inducing a complete system has been introduced. This includes criteria for determining when to address an incomplete system as solely missing data and when to address it as including scattered data as well. After this determination has been made, new algorithms for their resolution have

been developed. The new algorithms are based on the same principles and mathematics as the existing techniques that allow the probabilistic reconstructability analysis algorithms to be applied to g-systems.

The basic methodology for performing K-systems analysis is as follows:

1. Determine if the system is complete by assessing whether all possible states have an associated system function value.
2. If the system is complete, apply the K-systems algorithms. If not, determine the extent of the missing data as whether or not the system has either a) an empty closest state sets or b) existing states separated by more than half the maximum distance between states.
3. If neither of the two criteria are met, the closest states algorithm may be immediately applied and the results submitted for K-systems analysis. No further processing is required to assess or use the results of this analysis.
4. If one of these criteria is met, perform two dimensional clustering (using each variable and the system function as the two dimensions) using the entropy similarity taxmap method. Determine whether the resulting system is complete. If it is complete, apply the K-system algorithms. If not, use the closest states algorithm and then complete the analysis.
5. Results generated using the two dimensional clustering require some additional computations if predictions of previously unknown variable values are desired. This is done by projecting the two dimensional clusters into the variable axis and calculating the probability that a particular variable value falls into one or more of

the clusters. Predictions are made for all non-zero probabilities and these predictions include the probability products of the possible clusters.

The author has provided a comprehensive methodology for resolving incomplete systems so that they may be submitted for K-systems analysis. An algorithm for imputing missing values has been presented that is superior to existing techniques. A unifying methodology for distinguishing between missing data and data scattering has been presented. A new similarity measure based on the mathematics of information theory has been discovered and its use illustrated within an existing clustering algorithm. Note that more research is needed related to both missing data and clustering as they apply to K-systems analysis. In particular, a comprehensive comparison of variants of the closest state algorithm may provide further insight into the use applicability of the general algorithm and the variants that are possible. Also, the entropy similarity and corresponding dissimilarity have been characterized in terms of the properties they possess and the relationship to the Euclidean distance. The use of the entropy similarity has been demonstrated within an existing algorithm, but additional applications using other algorithms or the development of an algorithm specific to the measure may yield additional benefits. One algorithm in particular that is based on the principle of maximum entropy and a pairwise similarity matrix seems especially promising for application of the entropy similarity measure [HOFM 97].

Bibliography

[ANDE 73] Anderburg, Michael R., (1973). Cluster analysis for applications. Acedemic Press, New York, New York.

[BEZD 80] Bezdek, James C. (1980). A convergence theorem for the fuzzy ISODATA clustering algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2, 1, 1-8.

[BEZD 92] Bezdek, James C. and Pal, Sankar K. (1992) Fuzzy Models for Pattern Recognitions, The Institute of Electrical and Electronic Engineers, New York, New York.

[BENI 94] Beni, Gerardo and Lu, Xiaomin (1994). A least bias fuzzy clustering method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16, 9, 954-960.

[CARM 68] Carmichael, J. W., George, J.A. and Julius, R.S. (1968). Finding Natural Clusters. *Syst. Zool.*, 17, 144-150.

[CARM 69] Carmichael, J.W. and Sneath, P.H.A. (1969). Taxometric maps. *Syst. Zool.*, 18, 402-415.

[CAVA 81a] Cavallo, Roger E. and Klir, George J. (1981). Reconstructability Analysis: Overview and Bibliography. *International Journal of General Systems*, 7, 1-6.

[CAVA 81b] Cavallo, Roger E. and Klir, George J. (1981). Reconstructability Analysis: Evaluation of Reconstruction Hypothesis. *International Journal of General Systems*, 7, 7-32.

[CAVA 82] Cavallo, Roger E. and Klir, George J. (1982). Decision Making in Reconstructability Analysis. *International Journal of General Systems*, 8, 243-255.

[COVE 91] Cover, Thomas M. and Thomas, Joy A. (1991). Elements of Information Theory. John Wiley and Sons, Inc. New York, New York.

[EVER 93] Everitt, Brian S. (1993). Cluster Analysis. John Wiley and Sons, Inc. New York, New York.

[GOUW 96] Gouw, Deky and Jones, Bush (1993). The Interaction of K-Systems Theory. *International Journal of General Systems*, 24, 163-169.

[GUIA 85] Guiasu, Silviu, and Shenitzer, Abe (1985). The Principal of Maximum Entropy. *The Mathematical Intelligencer*, 7, 42-48.

[HART 71] Hartley, H. O. and Hocking, R.R. (1971). The analysis of incomplete data, *Biometrics*, 27, 783-808.

[HART 75] Hartigan, John A. (1975). Clustering Algorithms. John Wiley and Sons, Inc. New York, New York.

[HOFM 97] Hofmann, Thomas and Buhmann, Joachim M. (1997). Pairwise data clustering by deterministic annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19, 1, 1-14.

[JONE 82] Jones, Bush (1982). Determination of Reconstruction Families. *International Journal of General Systems*, 8, 225-228.

[JONE 85a] Jones, Bush (1985). Determination of Unbiased Reconstructions. *International Journal of General Systems*, 10, 169-176.

[JONE 85b] Jones, Bush (1985). A Greedy Algorithm for a Generalization of the Reconstruction Problem, *International Journal of General Systems*, 11, 63-68.

[JONE 85c] Jones, Bush (1985). Reconstructability Analysis for General Functions. *International Journal of General Systems*, 11, 133-142.

[JONE 85d] Jones, Bush (1985). Reconstructability Considerations with Arbitrary Data. *International Journal of General Systems*, 11, 143-151.

[JONE 85e] Jones, Bush (1985). The Cognitive Content of System Substates. IEEE Workshop on Languages for Automation.

[JONE 86] Jones, Bush (1986). K-systems Versus Classical Multivariate Systems. *International Journal of General Systems*, 12, 1-6.

[JONE 89] Jones, Bush (1989). A Program for Reconstructability Analysis. *International Journal of General Systems*, 15, 199-205.

[KLIR 86] Klir, George J. (1986). The Role of Reconstructability Analysis in Social Science Research. *Mathematical Social Sciences*, 12, 205-225.

[KUMA 89] Kumar, Vinod, Kumar, Uma, and Hoshino, Kyuoji (1989). An Application of the Entropy Maximization Approach in Shopping Area Planning. *International Journal of General Systems*, 16, 25-42.

[LITT 87] Little, R.J.A. (1982). Models for non-reponse in sample surveys, *Journal of the American Statistical Association*, 77, 237-250.

[ORCH 72] Orchard, T. and Woodbury, M.A. (1972). A missing information principle: Theory and applications. *Proceedings of the 6th Berkeley Symposium on Math, Statistics and Probability*, 1, 697-715.

[PITT 89] Pittarelli, Michael (1989). Uncertainty and Estimation in Reconstructability Analysis. *International Journal of General Systems*, 15, 1-58.

[RUSP 69] Ruspini, Enrique H. (1969). A new approacch to clustering. *Inform. Control*, 15, 1, 22-32.

[SAND 83] Sande, I.G. (1983). Hot deck imputation procedures, in *Incomplete Data in Sample Surveys, Vol. III: Symposium on incomplete data, Proceedings*, New York: Acedemic Press.

[SHAN 48] Shannon, Claude E. (1948). A mathematical theory of communication. *Bell Systems Technical Journal*, 27, 379-423, 623-656.

[TRIV 93] Trivedi, Sudhir K. Reconstructability Theory for General Systems and it's Application to Automated Rule Learning. Ph.D. dissertation, Louisiana State University, Baton Rouge, LA, 1993.

[XIE 91] Xie, Xuanli L. and Beni, Gerardo (1991). A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13, 8, 841-847.

[ZADE 71] Zadeh, Lofti A. (1971). Similarity relations and fuzzy orderings. *Information Science*, 3, 177-200.

[ZADE 65] Zadeh, Lofti A. (1965). Fuzzy Sets. *Inform. Control*, 8, 338-353.

Vita

Gary J. Asmus was born in Hampton, Virginia, and grew up in Wisconsin and then Missouri where he graduated from Washington High School in Washington, Missouri. After beginning his college career in Missouri, he completed his undergraduate studies at Louisiana State University and Agricultural and Mechanical College in December 1992. He returned to Louisiana State University in 1994 as a Board of Regents Fellow and completed his doctoral work in 1998.

His interests center on the role of analogy in human and artificial intelligence. He pursues this interest through the study of numerical and K-systems analysis, clustering or unsupervised learning, and genetic and evolutionary algorithms. In particular, he is interested in application and development of simple techniques that can be applied to model complex systems of all types, from meteorology to the human mind.

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: Gary J. Asmus

Major Field: Computer Science

Title of Dissertation: Techniques for Resolving Incomplete Systems in K-systems Analysis

Approved:

Major Professor and Chairman

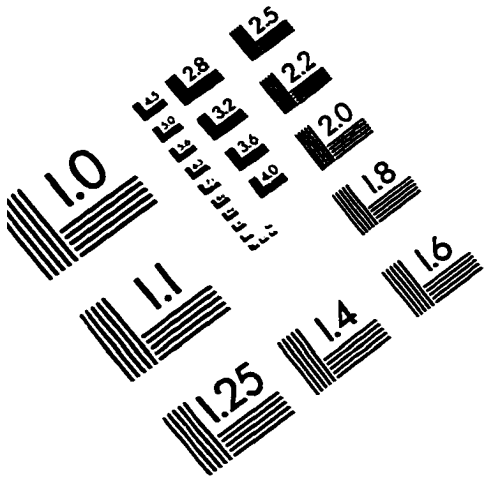
Dean of the Graduate School

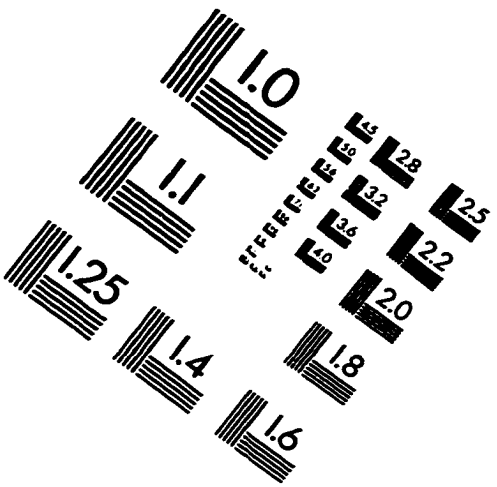
EXAMINING COMMITTEE:

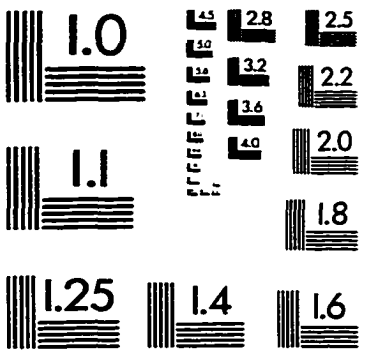
Date of Examination:

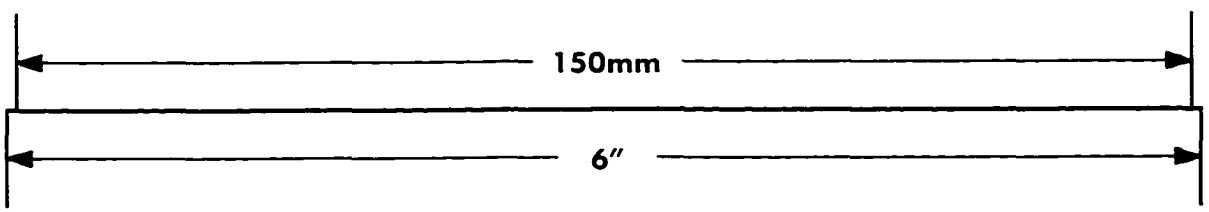
October 15, 1998

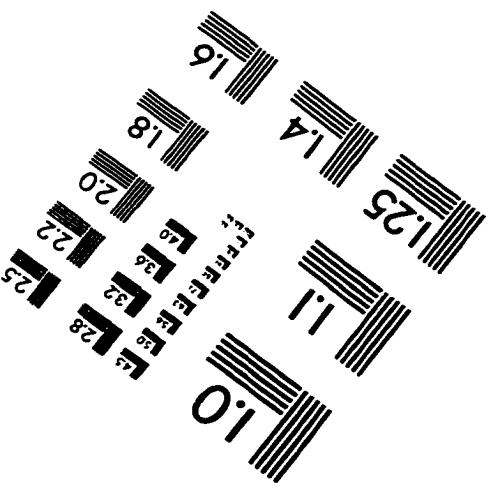
IMAGE EVALUATION TEST TARGET (QA-3)

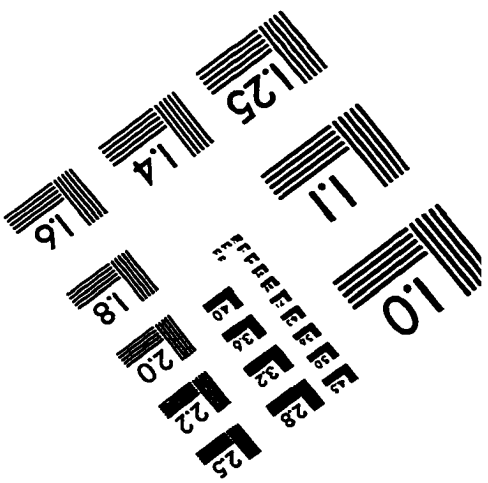












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989