

# Ordering Genetic Algorithm Genomes With Reconstructability Analysis: Discrete Models

Stephen Shervais  
Eastern Washington University  
College of Business and Public Administration  
Cheney, WA 99004  
sshervais@ewu.edu

Martin Zwick  
Portland State University  
Systems Science Ph.D. Program  
Portland, OR 97201  
zwickm@pdx.edu

**ABSTRACT** – *The building block hypothesis implies that genetic algorithm effectiveness is influenced by the relative location of epistatic genes on the chromosome. We demonstrate this with a discrete-valued problem, based on Kauffman’s NK model, and show that information-theoretic reconstructability analysis can be used to decide on optimal gene ordering.*

**Keywords:** Reconstructability analysis, genetic algorithms, transposition, crossover, optimization, OCCAM

## 1. Introduction

The impact of one gene on the fitness contribution of another is called *epistasis*. (In other contexts, this might be called an “interaction effect” or might be said to exemplify “synergy.”). Holland’s schema theorem and the building block hypothesis suggest that the performance of a genetic algorithm (GA) will be influenced by the relative location on the chromosome of genes exhibiting epistasis.

This paper extends previous work on the topic [1]. Here, after describing the schema and building block hypotheses and their relevance to epistasis (Section II), we further demonstrate the existence of a gene order effect in a Kauffman NK model (Section III). We then show that the methodology of reconstructability analysis can be used to discover preferred gene orders from data obtained by sampling the solution space (Section IV). Finally, we discuss the results of these preliminary experiments and point to areas for future exploration (Section V).

## 2. Schema Theorem, Building Blocks, And Epistasis

The schema theorem was first proposed by Holland [2] as a description of how adaptive systems “persistently test and incorporate structural properties associated with better

performance (p. 66).” Although there is now some doubt as to how well it describes the dynamics of the GA search process [3][4], it is still useful as a conceptual device, and we use it that way here. According to the schema theorem, GAs work by parallel testing of multiple combinations of bit strings made up of the available alleles. In the typical binary chromosome, the alleles may be represented as 1, 0, and \* (don’t care). Thus, 110\*\*\*11 is a schema (call it S1) of defining length seven and a schema order of five. (Note that the term *order* has two meanings in this paper. The order of a schema is the number of non-\* positions; the order of genes on a chromosome refers to their relative physical placement. Most often, and unless otherwise noted, we will be using the word in its second meaning.) S1 also contains a shorter schema (S2), 110\*\*\*\*\*, with a defining length of two and a schema order of three, and a third schema (S3), \*\*\*\*\*11, with a defining length one and schema order of two. In fact, an eight-bit-long schema with a maximum defining length of seven has  $3^8$  possible schemata embedded in it, but we here discuss just these three.

If strings containing S2 have a higher-than-average fitness, they will be preferentially selected, and S2 will act as a *building block* that can be assembled with other building blocks to create longer schemata and higher fitness bit-strings. Since the ratio of the defining length to the schema order is low, S2 is not likely to be broken up by the crossover operator. The same argument applies to S3. Now consider S1. If bitstrings containing this schema have a higher than average fitness, they will be preferentially selected as well. However, since the defining length of S1 is large relative to its schema order, it also stands a higher chance of being broken up during crossover. If S2 and S3 are both important to the fitness of S1, we would be better off changing the representation so that S2 and S3 are close together. In other words, if S1 has high fitness, it would be more likely to survive recombination if we had some good reason to move the 11 alleles over to be adjacent to the 110 alleles, i.e., to recode the genome so that this schema was

11011\*\*\*. Note that this will be the case even if the fitness contributions of the S2 and S3 schemas are independent of each other, but it will be especially true if the fitnesses of S2 and S3 interact in some way, that is, are epistatic. In the schema S1 discussed above, assume that the high fitness of S1 derives from an epistatic interaction between S2 and S3, and not merely from the separate high fitnesses of these two schemas. This would be all the more reason for S2 and S3 to be adjacent to one another and constitute a compact building block.

Although the usefulness of short building blocks has long been understood, only a few researchers have addressed the issue of how changing gene order might facilitate reaching enhanced fitness. Barbara McClintock is credited with discovering the importance of gene transposition in nature [5]. Transposition is thus available as a possible genetic operator available for use by GA researchers. Simoes and Costa [6] examined the usefulness of McClintock's transposons as a replacement for the crossover operator. In their work, randomly selected runs of bitstrings were moved about on the chromosome. While the study displayed the effectiveness of the transposition operator, the experimenters reported no data that would allow identification of the most effective bitstring orders.

Based on this idea, Beasley et al. [7] used *a priori* knowledge to code interactive genes into *sub-problems*, which are subject to separate evolutionary processes and are recombined each generation. This requires that some exogenous process identify the sub-problems.

Goldberg, et al. [8] developed the "fast messy" GA, which, among other things, allows the GA to evolve gene locations on the chromosome. They did this by coding stretches of the chromosome with a gene identifier, which specified the gene that that part of the chromosome represents. A given gene might start out over-expressed in a chromosome, because its identification code appears at two different locations. The program selects the first instance of the gene and ignores the rest. Alternatively, a gene might be under-expressed if it does not appear in the bitstring at all. The program then applies a default template to supply the missing gene values. As evolution proceeds, and the length of the GA is allowed to change from long to short and back to long again, those bitstrings with efficient gene orders will be preferentially selected. The difficulty with this approach is that the search for an efficient gene order proceeds in parallel with the search for an optimal solution.

A somewhat similar approach is Linkage Learning [9][10]. The linkage learning algorithm maintains at least one copy of each allele value, but only evaluates the first value it finds. Thus, the chromosome length  $L$  equals the number of genes plus some number of intron genes times the the product of the allele cardinalities for all genes (in-

cluding intron genes) Instead of crossover, a short section of chromosome is transferred from a donor chromosome, and excess genes are deleted until the new chromosome is back to length  $L$ . Over time, those chromosomes with the best gene orders will do better.

The difficulty with both these approaches is that the search for an efficient gene order proceeds in parallel with the search for an optimal solution. In addition, there is no indication how good the process is at finding the correct gene orders. Since both mGA and LLGA are more efficient than the simple GA, they are presumably adept at making *some* improvements to the gene order. However, in none of the literature that we have reviewed and referenced in this papers do the authors report on the final gene orders found. Our approach, described below, separates the two tasks, explicitly solving for an improved gene order in an efficient manner, and then structuring a simple GA based on that order.

### 3. Gene order effects in genetic algorithms

Prior work in this area [1] demonstrate the possibility of a gene order effect by using continuous fitness functions, including a set of linked DeJong F2 functions (for a summary of GA benchmark functions, see [11]). This work extends the research into the realm of discrete chain models, specifically an example of Kauffman's NK models [12]. In this work,  $N$ , the number of variables, is 9 ( $A$  through  $I$ ), and  $K = 1$ , specifically, the right-hand neighbor. Instead of wrapping the connection from variable  $I$  back to variable  $A$ , we limit our consideration to eight variable pairs:  $AB$  through  $HI$ . The chromosome is  $2N$  bits long, and is normally read from left to right, starting at the left end. Variables are represented by two binary bits, which means individual variables may take on four different values (that is, the Kauffman  $A$ -parameter = 4). Variable pairs are represented by their combined bitstrings, so there are 16 different combinations of bits available to a given variable pair. These 16 combinations are given arbitrary values in the range 0-100. The total space of all possible solutions is  $4^9 = 262,144$ .

The GA is generation-based, using roulette wheel selection for reproduction based on a breeding population of 30. The crossover rate is 1.0, with two mirror-image offspring produced, mutated ( $P_{mu} = 0.03$ ), and inserted in the population until the total is 60, at which time the population is ordered by fitness and the bottom 30 are dropped.

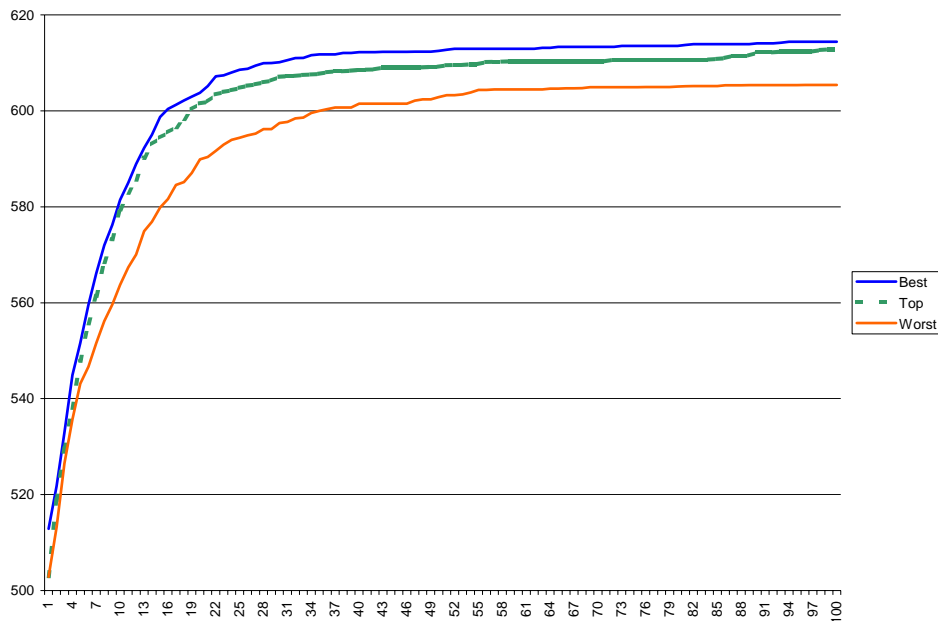
Our hypotheses is that placing functionally related genes close together enhances GA performance. This can be illustrated with a 4 variable problem. Suppose that  $A$  and  $B$  are functionally related, and  $C$  and  $D$  are also functionally related, then an order such as  $ABCD$  places the related variables *adjacent* to one another, and makes it less likely

that they will be separated by crossover. Permuting the variables within each pair, and permuting the pairs of variables produces equivalent orders, i.e., ABDC, BACD, BACD, BADC, and their four inverses are all equivalent to ABCD with respect to these considerations. If the gene order were ACBD, however, the functionally related variables would be *separated* and not adjacent to one another, and would be more easily separated by crossover. Associated with the above hypothesis is the null hypothesis that gene position has no impact on the effectiveness of the GA search.

The fitness functions used in this study involve eight variables and are of the form  $f = f_1(A,B) + f_2(B,C) + f_3(C,D) + f_4(D, E) + f_5(E, F) + f_6(F, G) + f_7(G, H) + f_8(H,I)$ . Thus according to the hypothesis above, variable order ABCDEFGHI is the natural order for this fitness function and should yield optimum GA performance. Orders are tested with 100 initializations of the GA, and differences between adjacent and separated orders are tested via a t-Test applied to the output from each generation. Results are presented and discussed in Section V.

#### 4. Detecting optimal gene order by reconstructability analysis

If gene order matters for GA performance, it is desirable



**Figure 1 Impact of Gene Order on Genetic Algorithm Performance.** The ability of a GA to maximize output of an N-K model is demonstrated for three different gene orderings. The “Best” order is the known-best ABCDEFGHI ordering. The “Top” order is the model identified as the best via reconstructability analysis. The “Worst” order is a model handcrafted to have a maximally separated gene structure. The results are the average of one hundred data runs of each. The x-axis represents number of generations, while the y-axis represents the fitness score.

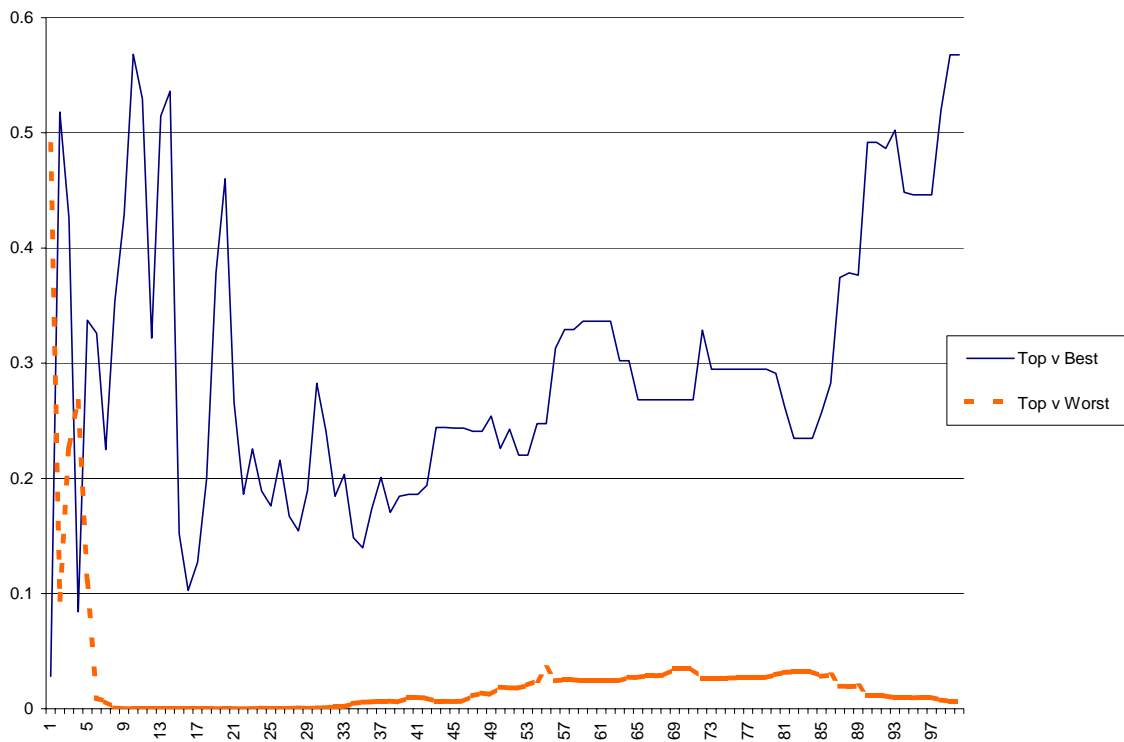
to be able to find out what the optimum gene order is. In this section we show that this determination is achievable, at least for the cases examined here, using the methods of reconstructability analysis.

Reconstructability analysis (RA) is a methodology for multivariate modeling of discrete, typically nominal (qualitative, unordered) variables; where variables are continuous, they must first be discretized (“binned”) to be analyzed. RA derives from Ashby [13], and was developed by Broekstra, Cavallo, Cellier, Conant, Jones, Klir, Krippendorff, and others; an extensive bibliography is available in [14], and a compact summary of RA is available in [15] and [16]. RA resembles log-linear (LL) methods, used widely in the social sciences, and where RA and LL methodologies overlap they are equivalent ([17],[18]). In RA [19], a probability or frequency distribution or a set-theoretic relation is decomposed (compressed, simplified) into component distributions or relations. The most common approach is typically the decomposition of frequency or probability distributions, where RA does statistical analysis. RA can model problems both where “independent variables” (inputs) and “dependent variables” (outputs) are distinguished (*directed* systems) and where this distinction is not made (*neutral* systems). In the present case, we have a directed system, with nine inputs A-I, and one output, the fitness value. In standard RA calculations [20] and in the prior work of the authors, the output is treated as an

explicit variable and is discretized, but in this study we use the “k-systems” approach of Bush Jones [21], where the output is linearly rescaled so that it can be treated as a frequency distribution, which is then analyzed as if it were the distribution of a neutral system.

Consider an observed frequency distribution  $g(A, B, C, D, E, F, G, H, I)$ , written simply as ABCDEFGHI. This observed distribution is some sample of a fitness function  $f(A, B, C, D, E, F, G, H, I)$ , and constitutes our data. RA decomposes the observed distribution into a set of projected frequency distributions which define a model. For example, the model ABCDE:FGHI specifies two projected distributions,  $g_1(A, B, C, D, E)$  and  $g_2(E, F, G, H, I)$ , which when put together yield a distribution,  $g'$ , that approximates  $g$ . The distributions are put together by a maximum-entropy (uncertainty) method. The model distri-

distance,  $\sum p \log(p/p')$ , normalized to a [0, 100%] range. By definition, the data itself, ABCDEFGHI has 100% information (0% error). The “independence model,” A:B:C:D:E:F:G:H:I, has 0% information. The model of the data is less complex (has fewer degrees of freedom) than the data, and models are assessed for statistical significance, usually with the Chi-square distribution. In the present problem, with the fitness function defined above, RA is expected to find that the “chain model,” AB:BC:CD:DE:EF:FG:GH:HI, captures a high percentage of the data. The critical limitation that RA faces is the size of the sample relative to a complete specification of the fitness function. Note that this model uniquely corresponds to the variable order ABCDEFGHI (or its inverse). All other chain models will correspond to different variable orders. Thus the optimal variable order is selected as the order corresponding to the chain model with the highest



**Figure 2 t-Test Comparison Of The Top Occam Model And The Best and Worst Gene Orderings.**

The solid line shows that, after the confusion of the first five generations, the Best model and the Occam-selected Top model are not statistically different. The dashed line shows that, after the 5<sup>th</sup> generation, differences between the Top model and the hand crafted Worst model are all statistically significant. The x-axis represents number of generations, while the y-axis represents the fitness score.

bution is compared to the observed distribution, and the difference (error) represents loss of information in the model. Information is defined as follows: let  $p$  be the probability distribution obtained by normalizing  $g$  by the sample size, and let  $p'$  be the model probability distribution corresponding to  $g'$ . Information is the Kullback-Liebler

information content.

In order to determine the best model, we generated a 10% sample of the problem space, a total of 26,214 unique solutions, selected at random. (Note: the proportion of the problem space that needs to be searched is an on-going

research question.) This random solution set is processed by the Occam software package at Portland State University, which provides us with models of the structure in the data, models that can be used to infer the best gene order. We selected three gene order models for testing in the GA. First, we used the natural order, the model that is known to have the best structure for the problem. Second, we used the top model, as chosen by Occam, based on its information content. Finally, we used a model that was hand-crafted to have maximally separated genes.

Calculations were made using the RA software programs developed at Portland State University, now integrated into the package OCCAM (for the principle of parsimony and as an acronym for “Organizational Complexity Computation And Modeling”). The earliest of these programs was developed in [22]; a review of RA methodology is offered in [23] and [16]; a list of recent RA papers in the PSU group is given in [15]. A description of the OCCAM architecture is given in [24].

## 5. Results, discussion, and future work

Figure 1. shows the effect of gene order on GA performance for epistatic genes in a Kauffman N-K model. The topmost of the two solid lines shows the performance of the GA when the chromosome uses the natural (Best) gene order – ABCDEFGHI, while the lower line shows the performance of a GA using a chromosome deliberately designed to be maximally separated – ACEGIBDFH. The natural order has a clearly visible difference, and a t-Test shows this difference to be statistically significant from the fifth generation on (Figure 2.).

The first ten of the 181441 possible models identified through RA are shown in Table 1. Because the dataset does not exhaust the state space, it is possible that RA will

**Table 1 Occam Output Models.** Occam generates thousands of models when dealing with this many variables. These are the top ten models, in terms of information content, based on a 10% sample of the state space. In terms of impact on GA performance, there is no statistical difference between the Top model and the Known Best model.

	MODEL	Information
<b>Top Model</b>	AB:AC:CD:DE:EF:FG:GH:HI	0.0036
	AB:BD:CD:CE:EF:FG:GH:HI	0.0036
	AC:AI:BD:CD:EF:FG:GH:HI	0.0035
	AB:AF:BD:CD:CE:FG:GH:HI	0.0035
	AB:AC:BD:DE:EF:FG:GH:HI	0.0035
	AB:AC:BI:CD:DE:EF:FG:GH	0.0035
	AB:AI:BD:CD:CE:EF:FG:GH	0.0034
	AC:AE:BD:CD:EF:FG:GH:HI	0.0034
	AC:BD:BI:CD:EF:FG:GH:HI	0.0034
	<b>Best Model</b>	AB:BC:CD:DE:EF:FG:GH:HI

not identify the exact model that created the dataset, and that is what happened in this case. The topmost model, the one that carried the most information, was model BACDEFGHI. Note that this differs from the actual model by only one letter pair, and the information content of both is similar – 0.0036 compared with 0.0034. The similarity between the two models means their impact on the performance of the GA should be similar, and that is what we find. In Figure 1, the center curve is that produced by use of the Top model. In Figure 2. the lower, dashed, curve, shows the results of the t-Test for one hundred datapoints in each of the one hundred generations. After the 5<sup>th</sup> generation, there is no statistical difference between the Top model and the known-Best model.

We have shown that Reconstructability Analysis allows one to find the models that retain high information about the data. Using simple chain models, one can obtain an indication of how to order the variables on the GA chromosome, and it appears that the adjacent orders would be better than the separated orders. Moreover the use of disjoint models might be a way to solve Beasley et al.’s problem of a priori identification of subproblems for expansive coding. Breaking the current problem into subproblems AB, AC, CD, DE, EF, FG, GH, HI, solving the subproblems separately, and merging the answers would probably give a good result. Using RA to decompose optimization problems into subproblems might of course also be useful for optimization methods other than the GA.

This work represents a preliminary examination of the possibility of optimizing discrete-valued problems using reconstructability analysis to prestructure a GA. Several areas remain to be explored, among them, the impact of sample size, and problem difficulty. In addition, the current stand-alone OCCAM code will need to be linked to the GA, to allow future work to do head to head comparison with other gene-ordering techniques, such as linkage learning and estimation of distribution algorithms.

## References

- [1] M. Zwick, and Shervais, S. Reconstructability Analysis Detection Of Optimal Gene Order In Genetic Algorithms. In Proceedings of 12th International World Organization of Systems and Cybernetics and 4th International Institute for General Systems Studies Workshop, Pittsburgh, March 24-26, 2002. In *Kybernetes*, Vol 33, No 5/6 (2004), 1053-1062.
- [2] J. Holland, *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, MI, 1975.
- [3] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.

- [4] C. Thornton, The building block fallacy. *Complexity International* 4. (1997), <http://www.csu.edu.au/ci/vol04/thornton/building.htm>
- [5] B. McClintock, *The discovery and characterization of transposable elements: the collected papers of Barbara McClintock*, Garland, New York, NY, 1987.
- [6] A. Simoes, and E. Costa, Transposition: {A} Biologically Inspired Mechanism to Use with Genetic Algorithms, In: *Proceedings of the Fourth International Conference on Neural Networks and Genetic Algorithms (ICANNGA 99)*, Portoroz, Slovenia. Springer Verlag, Berlin, 1999, 178-186.
- [7] D. Beasley, D. Bull, and R. Martin,. Reducing Epistasis in Combinatorial Problems by Expansive Coding. In: *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufman Publishers, San Mateo, 1993, 400-407.
- [8] D. Goldberg, K. Deb, H. Kargupta, and G. Harik, *Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms*. IlliGAL Report No. 93004, University of Illinois, 1993.
- [9] F. Lobo, K. Deb, D. Goldberg, G. Harik, and L. Wang, *Compressed intron in a linkage learning genetic algorithm*. ILLIGAL Technical Report No. 97010, December 1997.
- [10] G. Harik, and D. Goldberg, *Learning Linkage*, ILLIGAL Technical Report No. 96006, August 1996.
- [11] J. Digalakis, and K. Margaritis, An experimental study of benchmarking functions for Genetic Algorithms. In *Proceedings of the IEEE International Conference On Systems, Man & Cybernetics*, Nashville, TN, October 2000, 3810-3815.
- [12] S. Kauffman, *The Origins of Order*, Oxford University Press, Oxford, 1993.
- [13] W. Ashby, Constraint Analysis of Many-Dimensional Relations, *General Systems Yearbook*, 9, (1964), 99-105.
- [14] G. Klir, Reconstructability Analysis: An Offspring of Ashby's Constraint Theory. *Systems Research*, 3 (4), 1986, 267-271.
- [15] M. Zwick, *Discrete Multivariate Modeling*, 2001. [http://www.sysc.pdx.edu/res\\_struct.html](http://www.sysc.pdx.edu/res_struct.html)
- [16] M. Zwick, An Overview of Reconstructability Analysis. In Proceedings of 12th International World Organization of Systems and Cybernetics and 4th International Institute for General Systems Studies Workshop, Pittsburgh, March 24-26, 2002. <http://www.sysc.pdx.edu/download/papers/ldlpitfabstract.htm>
- [17] Y. Bishop, S. Feinberg, and P. Holland, *Discrete Multivariate Analysis*. MIT Press, Cambridge, MA, 1978.
- [18] D. Knoke, and P. Burke, *Log-Linear Models*. (Quantitative Applications in the Social Sciences Monograph # 20. Sage, Beverly Hills, CA, 1980.
- [19] G. Klir, *The Architecture of Systems Problem Solving*. Plenum Press, New York, NY, 1985.
- [20] K. Krippendorff, *Information Theory: Structural Models for Qualitative Data*. (Quantitative Applications in the Social Sciences #62.) Sage, Beverly Hills, CA, 1986.
- [21] B. Jones, "Reconstructability Analysis for General Functions." *International Journal of General Systems*, Vol. 11 (1985), 133-142.
- [22] J. Hosseini, R. Harmon, and M. Zwick, Segment Congruence Analysis Via Information Theory. In *Proceedings, International Society for General Systems Research*, Philadelphia, PA, May 1986, G62 - G77.
- [23] M. Zwick, Wholes and Parts in General Systems Methodology. In *The Character Concept in Evolutionary Biology*. Academic Press, New York, 2001, 237-256. <http://www.sysc.pdx.edu/faculty/Zwick/research.html#wholes>
- [24] K. Willett, and M. Zwick, A Software Architecture for Reconstructability Analysis. In Proceedings of 12th International World Organization of Systems and Cybernetics and 4th International Institute for General Systems Studies Workshop, Pittsburgh, March 24-26, 2002, <http://www.sysc.pdx.edu/download/papers/softarcabstract.htm>