COVER PAGE

Title :
Towards a General Model for Heuristics and Heuristic-Preserving Problem Reductions

Authors :
Fernando Náufel do Amaral and Edward Hermann Haeusler

Affiliation :
Dept. of Informatics,
Pontifícia Universidade Católica,
R. Marquês de São Vicente, 225,
CEP 22453-900 – Rio de Janeiro, RJ, Brazil
Phone no.: +55-21-9768-9751
email: {fnaufel,hermann}@inf.puc-rio.br

Keywords :
Heuristics, Problem Theory, Problem Reductions, Formal Models, Category Theory

Track :
Open Discussion Track

Conference Topics :
AI Foundations

Towards a General Model for Heuristics and Heuristic-Preserving Problem Reductions

Fernando Náufel do Amaral and Edward Hermann Haeusler

Dept. of Informatics,
Pontifícia Universidade Católica, Rio de Janeiro, Brazil
{fnaufel,hermann}@inf.puc-rio.br

Abstract. We present the first definitions and results of ongoing work to define a general formal model for heuristic techniques. One of our goals is to use this model to study problem reductions that preserve properties of interest of the heuristics involved. We use Category Theory as our main tool.

1 Introduction

It has been a common task to model and formalize a research field after some set of important and relevant results have been obtained. This is the very case in problem-solving techniques based on Artificial Intelligence/heuristics. As with the NP = P question, the knowledge that the scientific community has on the so-called *hard* problems is of the kind that might be termed exploratory. It is worthwhile mentioning the advances on the creation of heuristics for dealing with certain practical instances of these problems requiring near-optimal solutions, as opposed to exact and optimal ones. Simulated Annealing, Tabu Search and Genetic Algorithms are among the most popular ways of achieving solutions for hard problems with randomness and approximability features. On the other hand, some problems are known that allow quite good approximation algorithms to be built from knowledge obtained from a detailed study of their own nature. In this scenario there is a lot to be learned besides the establishment of a relevant methodology for approaching this issue. These ways of applying heuristics to solve hard problems, however, are often case-driven and success-driven. One can wisely point out that this is only a facet of the essential question of how to solve problems efficiently using a computer. However, there are also some foundational/methodological approaches to this. Here it is worth mentioning the studies that try to mathematically relate problems, solutions, patterns of achieving solutions and so on. Instead of developing specific tools to solve some class of problems, this kind of study aims at the establishment of some characterization result that helps to understand the very nature of the hard problems. The discovery of NP-complete problems is one of the most representative examples of this sort of research. Since the 70's there has been some important work addressing this panoramic approach ([1, 13], among others). In most of this research the notion of reduction between problems plays a central rôle. [12] has

extended this definition of reduction between problems to cope with approximate solutions and quality-preserving reductions.

Category Theory was a milestone in the development of a mathematical language for dealing with the universals of a research area. The seminal work relating mathematical results from topology and algebra (culminating in the cohomology concept), the establishing of a foundational alternative for the mathematical discourse, and its widespread use in the various aspects of computational models helping the elucidation of some basic question regarding concurrency theory, typing discipline, software specification and so on, are quite good examples of the application of Category Theory. This article follows the approach started by [5] in applying categorical tools to the analysis of hard problems and their relationships. Here the focus is on the heuristics. The establishment of some definitions of heuristics and heuristic-preserving morphisms between problems is one of our goals. By relating problems which lend themselves to the application of the same heuristic strategies we hope to gain some knowledge on the intrinsic nature of the hard problems discussed above; perhaps the establishment of some universals in the classes associated to each kind of heuristic will lead to some completeness-like concept pertaining to heuristics. However, we have not achieved the main goal yet. This paper describes our first results and definitions and discusses some future directions.

The paper is organized as follows: Sec. 2 presents the basic definitions of a General Theory of Problems on which we base our own work; Sec. 3 shows that every problem may be viewed as one of search, allowing us to concentrate on the class of search problems but draw general conclusions that apply to problems of any kind; Sec. 4 begins to formalize the process of operationalizing a problem by defining proto-operators as primitive objects that specify how the search graph of a problem may be built; finally, Sec. 6 presents a discussion of the present status and future directions of our work, as well as a brief comparison to related work in the literature.

2 A General Theory of Problems

We base our formal definitions of problems, solutions and reductions between problems on Veloso's work on a General Theory of Problems ([13]):

Definition 1 (Problem). *A problem P is a tuple $< D, R, p >$, where*

- *D is a nonempty set called the* data domain. *An element d of D is called an* instance*;*
- *R is a nonempty set called the* result domain. *An element r of R will usually be called an* answer*;*
- *p is a relation over $D \times R$ called the* problem condition. *Having $(d, r) \in p$ means that instance d has r as one of its acceptable (correct) answers.*

Definition 2 (Solution). *A solution of P is a function $\phi : D \longrightarrow R$ such that*

$$\forall d \in D \quad (d, \phi(d)) \in p$$

Definition 3 (Link). *Given* $P = < D, R, p >$ *and* $Q = < E, S, q >$, *a link* L *from* P *to* Q *is a pair* (τ, σ), *where*

- $\tau : D \longrightarrow E$ *is the* data transformation function;
- $\sigma : S \longrightarrow R$ *is the* result retrieval function.

Definition 4 (Reduction). *A reduction of* P *to* Q *is a link* L *from* P *to* Q *such that for any solution* ϕ *of* Q, *the composite* $\sigma \circ \phi \circ \tau$ *is a solution of* P.

Proposition 1. *Problems and reductions form a category* $\mathcal{PROB}$.

Example 1 (The TSP). The Traveling Salesman Problem (TSP) is defined as $< D, R, p >$, where

- D is the set of all finite complete directed graphs with a distance associated to each edge;
- R is the set of all hamiltonian circuits over the graphs in D;
- p relates each graph $d \in D$ to its minimum-distance hamiltonian circuit (or circuits).

2.1 Search Problems

An important class of problems is the class of search problems, defined as follows:

Definition 5 (Search Problem). *A search problem is a problem of the form* $< G, N, s >$, *where*

- G *is a set of graphs, each graph having a designed initial node* n_0;
- N *is a set of sets of nodes,* $N \subseteq \bigcup_{g \in G} \wp(nodes(g))$;
- $s \subseteq \{(g, S) \mid S \subseteq nodes(g)$ *and* $\forall n \in S$ *there is a path in* g *from* n_0 *to* $n\}$.

Having the result domain N as a set of sets of nodes allows for search problems where the goal is to find more than one node (possibly all nodes) of the graph having a certain property.

Note that the TSP, as defined in Example 1 above, is *not* a search problem.

3 Every Problem is Isomorphic to a Search Problem

Our first step in the definition of a heuristic for a problem of any kind is seeing the problem as one of search, so that heuristic search techniques may then be used to solve it. This can be accomplished by realizing that solving an instance $d \in D$ consists in searching for a specific element $r \in R$ such that $(d, r) \in p$. For example, the search can be conducted according to a "split-and-prune" strategy (see [7]) where we initially consider all members of R as possible answers and then repeatedly and systematically discard answers until we are left with one that is acceptable for the instance in question. Alternatively, sets of candidate answers can be manipulated according to so-called "modern heuristic techniques"

(Genetic Algorithms, Tabu Search or Simulated Annealing – see [11]) until an acceptable answer is found.

It helps – but is by no means necessary – to associate to each instance $d \in D$ a subset $f(d) \subseteq R$ containing all the answers in R meeting some well-defined criteria that an answer for d must satisfy. This defines a function $f : D \to \wp(R)$, called the *feasible answer function*. We require, for all r with $(d, r) \in p$, that $r \in f(d)$; i.e., every correct answer for an instance d must be a feasible answer for d.

In order to construct the search problem associated to a problem $P = < D,\ R,\ p >$, we start out by defining for each instance $d \in D$ a huge graph g_d where the nodes represent sets of feasible answers and where each edge represents a change in the set of feasible answers currently considered as candidate answers for d. Because some modern heuristic techniques usually add elements to the current set of candidate answers (unlike split-and-prune, where the current set is only partitioned at each step), we define g_d as a complete graph, to account for any kind of change in the set of candidate answers.

Furthermore, we also want to retain information on the instance d which gave rise to this search graph, so we define each node of the graph to be a pair (S, d), with $S \subseteq f(d)$ a set of feasible answers for d and d the instance in question.

Definition 6 (Search problem associated to a problem). *Given a problem $P = < D,\ R,\ p >$, define the search problem $P' = < D',\ R',\ p' >$, where*

- *Let $f : D \to \wp(R)$ be the feasible answer function for P (as described above). Let $d \in D$. Let g_d be the complete graph whose set of nodes is $\wp(f(d)) \times \{d\}$ and whose initial node is $(f(d), d)$. Then $D' = \{\ g_d \mid d \in D\ \}$;*
- $R' = \{\ \ \{(\{r\}, d)\} \mid r \in f(d),\ d \in D\ \ \}$;
- $p' = \{\ \ (g_d, \{(\{r\}, d)\}) \mid (d, r) \in p\ \ \}$

As there are obvious reductions (τ, σ) from P to P' and (τ^{-1}, σ^{-1}) from P' to P which, when composed, yield the respective identity reductions, we have the following

Proposition 2. *P and P' are isomorphic in $\mathcal{PROB}$.*

Example 2 (The TSP as a search problem).

The feasible answer function $f : D \to \wp(R)$ maps each graph $d \in D$ to the set of all hamiltonian circuits of d.

The search version of the TSP (Example 1) is TSP' $=< D', R', p' >$, with

- $D' = \{\ g_d \mid d \in D\ \}$, where for each instance d of the TSP g_d is the complete graph where each node is a pair (A, d), with A a set of hamiltonian circuits of d; the initial node of g_d is $(f(d), d)$, with $f(d)$ the set of all hamiltonian circuits of d;
- $R' = \{\ \ \{(\{r\}, d)\} \mid r \in f(d),\ d \in D\ \ \}$; i.e., an answer for TSP' is a node corresponding to a pair consisting of an instance d of the TSP and of a singleton set with a hamiltonian circuit of d;
- $p' = \{\ \ (g_d, \{(\{r\}, d)\}) \mid (d, r) \in p\ \ \}$; i.e., a goal node of g_d is a node corresponding to a pair whose first component is a minimum-distance hamiltonian circuit of d.

4 Refining the Complete Graph: Proto-operators

The next step in defining a heuristic for a problem P would be determining the set of operators to be used in solving the associated search problem. Operators are applied to states in a state graph, but the definitions of "operator" and "state" are interdependent; so, we first define the more primitive notion of *proto-operator*.

Given the complete graph g_d for instance d, a proto-operator π will refine g_d, eliminating some of its edges to transform it into another search graph g_d^π.

Informally, we group subsets of the result domain together according to some specified criteria. Then, at each step of the search, we may delete or add (or both) an entire group of feasible answers from/to the current set of candidate answers.

Definition 7 (Proto-operator). *A proto-operator π for $P = <\ D,\ R,\ p\ >$ is a pair of families $(\{\alpha_i\}_{i\in I}, \{\delta_j\}_{j\in J})$ of equivalence relations over $R \times R$.*

The α family specifies how answers in the result domain should be grouped for being added to the current set of candidate answers, whereas the δ family determines how the grouping should occur when answers are being deleted.

Definition 8 (The refined graph g_d^π). *Given graph g_d for an instance d of a problem P and a proto-operator π for P, with $\pi = (\{\alpha_i\}_{i\in I}, \{\delta_j\}_{j\in J})$, the graph g_d^π has the same set of nodes and the same initial node as g_d, but the edges are defined as follows:*

Let (S, d), with $S \subseteq f(d)$, be a node. Each $\alpha_i \in \pi \downarrow 1$ and each $\delta_j \in \pi \downarrow 2$ induces a partition of S. Let $[r]_{\alpha_i}$ and $[r]_{\delta_j}$, with $r \in f(d)$, be the equivalence classes of r by α_i and δ_j, respectively. Then there will be an edge from (S, d) to each of the nodes in

$$\{\ (S \cup [r]_{\alpha_i}, d)\ \mid\ r \in S,\ i \in I\ \}$$

and to each of the nodes in

$$\{\ (S \cap [r]_{\delta_j}, d)\ \mid\ r \in S,\ j \in J\ \}$$

The edges to nodes of the first set correspond to steps where a group of answers is added to the current set S of candidate answers. The edges to nodes of the second set correspond to steps where a group of answers is deleted from S. An operation of a search process (e.g. a genetic algorithm) where both addition and deletion of answers occur simultaneously must be split into two edges in g_d^π, one edge for addition, one for deletion.

Formally, as we do not want g_d^π to be a multigraph, if the same edge is created by more than one equivalence relation in the α and δ families, we include only one such edge in g_d^π.

Definition 9 (Search problem with proto-operator). *Given a problem $P = <\ D,\ R,\ p\ >$ and a proto-operator π for P, we define the search problem with proto-operator $P_\pi = <\ D_\pi,\ R_\pi,\ p_\pi\ >$, where*

- $D_\pi = \{\ g_d^\pi \mid d \in D\ \}$;
- $R_\pi = R$;
- $p_\pi = \{\quad (g_d^\pi, \{(\{r\}, d)\}) \mid (d, r) \in p$ *and there is a path in* g_d^π *from* $(f(d), d)$ *to* $(\{r\}, d)\quad \}$.

Example 3 (A proto-operator for the TSP). Given an instance d of the TSP, the approach of finding a minimum-distance hamiltonian circuit for d by choosing one unvisited city at a time is represented by the following proto-operator π for the TSP:

$\pi = (\{\alpha_i\}_{i \in I}, \{\delta_j\}_{j \in J})$, with

- $I = \emptyset$; i.e. there will be no edges corresponding to the addition of answers in the refined graph;
- $J = \mathbb{N}^\star$
- For all feasible answers r_1, r_2 and for all $j \geq 1$

$$(r_1, r_2) \in \delta_j \quad \text{iff} \quad \text{prefix}(r_1, j) = \text{prefix}(r_2, j)$$

where prefix(r, j) is a function returning the path consisting of the first j cities of the circuit r. If r has fewer than j cities, then prefix$(r, j) = r$, making all of the δ_j copies of the identity relation for j greater than the number of cities in the instance.

In order to illustrate the effect of this proto-operator, consider a very small instance d of the TSP consisting of a complete graph with 4 nodes (named A, B, C and D). The distances associated to the edges are not relevant in this discussion. The feasible answer function $f(d)$ yields the set of all 24 hamiltonian circuits for this graph. The complete graph g_d has 2^{24} nodes, one for each subset of $f(d)$. The initial node of g_d is $(f(d), d)$ (recall that we "store" the identity of the instance d in every node of g_d).

The proto-operator π defined above transforms g_d into the graph g_d^π depicted in Fig. 1. Recall that the nodes of g_d^π correspond to pairs of the form (S, d), where S is a set of feasible answers and d is the problem instance that gave rise to the graph. In the figure, however, only the set S is shown for each node; furthermore, the common prefix characterizing a set S is used to represent the set, followed by "$\star$" (e.g. $AB\star$ stands for all hamiltonian circuits with the prefix A–B). Furthermore, the graph g_d^π is reflexive (there is an edge from every node to itself) and transitive (given an edge from n_1 to n_2 and an edge from n_2 to n_3, there is an edge from n_1 to n_3). These reflexive and transitive edges are not shown in the figure.

A proto-operator is considered *complete* if all correct answers for all instances of the problem are reachable in the refined graph.

Definition 10 (Completeness). *Given a problem* $P = <\ D,\ R,\ p\ >$, *a proto-operator* π *for* P *is* complete *iff*

$$\forall (d, r) \in p \quad \exists \text{ a path in } g_d^\pi \text{ from } (f(d), d) \text{ to } (\{r\}, d)$$

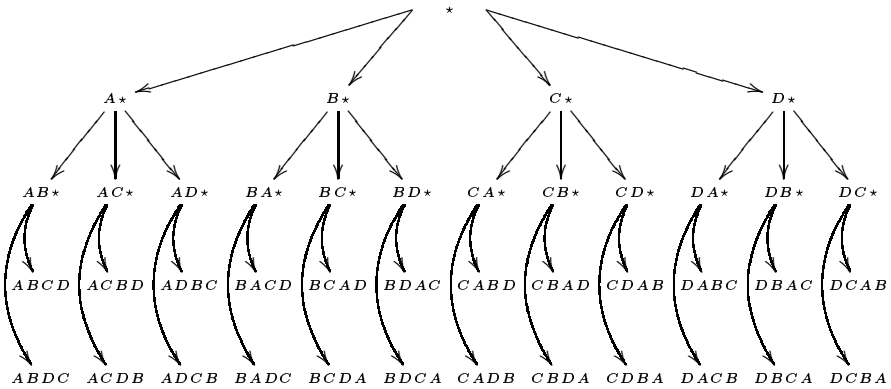

Fig. 1. The graph g_d^π for an instance d of the TSP

Proposition 3. *For any problem P and proto-operator π,*

- *There is a reduction from P to P_π;*
- *If π is complete, P is isomorphic to P_π in $\mathcal{PROB}$.*

5 Problem Reductions that Preserve Proto-operators

Although we have yet to obtain a precise definition of what heuristics are in our model, we discuss in this section when and how properties of the proto-operators we have defined may be preserved by problem reductions. In future developments of this work, ideas and results from this early discussion will hopefully be refined and applied to heuristics in the full model.

The one property of interest we mentioned above for proto-operators is completeness. This motivates the following definition:

Definition 11 (Proto-operator-preserving Problem Reduction). *Given two problems $P = <\ D,\ R,\ p\ >$ and $Q = <\ E,\ S,\ q\ >$, a proto-operator π for P and a proto-operator χ for Q, and a reduction (τ, σ) from P to Q, we say that the reduction* preserves proto-operators *iff*

$$\begin{aligned}\forall d \in D: \quad & \forall r \in R \text{ with } (d,r) \in p \text{ and } (\{r\}, d) \text{ reachable in } g_d^\pi : \\ & \exists s \in S \text{ with } (\tau(d), s) \in q \text{ and } (\{s\}, \tau(d)) \text{ reachable in } g^\chi_{\tau(d)} \\ & \quad \text{such that } r = \sigma(s)\end{aligned}$$

Informally, we say a reduction from P to Q preserves the proto-operators involved if, given any instance of P, every answer to this instance that is obtainable by P's proto-operator π can still be obtained by Q's proto-operator χ (modulo the data transformation and result retrieval functions τ and σ, respectively).

From a category-theoretical viewpoint, this gives rise to a category whose objects are problems (as in $\mathcal{PROB}$) and where a morphism from P to Q is a tuple $(\pi, \chi, \tau, \sigma)$, with π a proto-operator for P, χ a proto-operator for Q, and (τ, σ) a reduction from P to Q that preserves π and χ.

6 Concluding Remarks

6.1 Discussion

The search techniques found in the literature can be divided into two basic categories: the first consists of algorithms that attempt to construct an answer for an instance one step at a time, usually tracing the steps of this construction as a path in a suitable state graph associated with the instance being solved. An example of this approach would be an algorithm to solve the Traveling Salesman Problem by repeatedly choosing an unvisited node of the graph until a minimum-cost hamiltonian circuit is built. The second category is comprised of algorithms that start out with one or more candidate answers, applying repeated changes to them in order to generate new answers to replace some of the previous ones. Typical examples of this strategy are Genetic Algorithms and Tabu Search ([11]).

In the model described in the previous sections, the elements of the result domain are organized in a search graph whose nodes correspond to sets of answers and whose edges (induced by a proto-operator) represent the addition or deletion of answers to/from a set. This model is adequate for representing search techniques of both categories mentioned in the previous paragraph: a proto-operator that only induces edges representing deletions (i.e. a proto-operator with an empty α family) corresponds, by the split-and-prune paradigm, to the step-by-step generation of answers. On the other hand, a proto-operator having a nonempty α family is able to represent the operation of an algorithm of the second category, where the set of candidate answers is subjected both to additions and deletions.

This paper, as a communication of partial results of ongoing research, has explained the model only up to the definition of proto-operators and the statement of some related propositions. Parts of the model not treated here include the following:

- The definition of the notion of *state* as an equivalence relation between sets of answers. For the techniques in the first category, this equivalence relation is based on the "partial problem" that remains to be solved at the given point of the process of constructing the answer to the instance at hand. The definition of state may identify some nodes in the search graph induced by the proto-operator, transforming the search graph into a *state graph*. Note, however, that techniques falling into the second category described above do not usually make use of any definition of state. In this case, the search graph remains unchanged, for the states correspond exactly to the sets of answers in the graph.
- After "state" is defined, it becomes possible to speak of *pre-conditions*, and some edges of the state graph may be deleted because they correspond to the application of actions in states where such actions are not allowed. It is only at this point that we may speak of *operators* having been defined for the problem. Thus, in our model, the definition of a set of operators for a problem encompasses the definition of a proto-operator, of a notion of state and of a set of pre-conditions.

- For optimization problems, it becomes necessary to map the original problem to one of near-optimization, where the problem condition p has been extended to account for answers that are not exact, but rather approximations produced by the chosen heuristic strategy.
- Finally, in this setting, specific heuristic techniques may be modeled. These may include the definition of a strategy such as hill-climbing, best-first, means-ends analysis etc., as well as the heuristic functions used in the evaluation of states.

After the model is completely defined, it becomes possible to study problem reductions that preserve properties of interest that heuristics in general may possess, as was quickly illustrated by the proto-operator-preserving reductions presented in Sec. 5 above.

6.2 Related Work

Heuristics have been the object of study of many researchers since the 70's. The approaches vary widely, however. Lenat ([6]), for example, does not concern himself with the explicit construction of state spaces and operators for the operation of his AM program; he points out that it is the heuristics themselves (defined as IF-THEN rules) that dictate all the actions the program may perform. On the other extreme, Banerji ([1]) considers the state space and the operators as given right from the start, including them in his definition of problem. As indicated in Secs. 3 and 4 above, we are interested not only in giving a formal account of the construction of the state space, but also in basing this construction on the very structure of the problem.

Defining and analyzing formal properties of heuristics has also been common among researchers whose goal is to automatically generate heuristics (which is a natural objective to be pursued, since what has been formalized usually lends itself easily to automation) – see [3, 4], for example. To this end, it is common to manipulate and relate state graphs of different problems as we are interested in doing. But rather than studying the generation of heuristics, we are interested in exploring how problem reductions preserve properties of heuristics associated to the source problem.

As far as modern heuristic techniques are concerned, a unified model for Simulated Annealing, Tabu Search and Genetic Algorithms is found in [9].

Our work may be viewed as a generalization of the work on approximation-preserving reducibilities ([12]).

References

[1] Ranan B. Banerji. *Artificial Intelligence: a Theoretical Approach.* North Holland, 1980.

[2] Ranan B. Banerji, editor. *Formal Techniques in Artificial Intelligence: a Sourcebook.* Studies in Computer Science and Artificial Intelligence. North Holland, 1990.

[3] István T. Hernádvolgyi and Robert C. Holte. The automatic creation of memory-based search heuristics. Available from http://citeseer.nj.nec.com/296915.html.

[4] J.P.E. Hodgson. *Automatic Generation of Heuristics*, pages 123–171. In Banerji [2], 1990.

[5] Liara Aparecida Santos Leal, Paulo Blauth Menezes, Dalcidio Moraes Claudio, and Laira Vieira Toscani. Optimization problems categories. In R. Moreno-Diaz and A. Quesada-Arencibia, editors, *EUROCAST 2001 – Eighth International Conference on Computer Aided Systems Theory – Formal Methods and Tools for Computer Science – Extended Abstracts*, pages 93–96. Universidad de Las Palmas de Gran Canaria, 2001.

[6] Douglas B. Lenat. The nature of heuristics. *Artificial Intelligence*, 19(2):189–249, October 1982.

[7] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. The Addison-Wesley Series in Artificial Intelligence. Addison-Wesley, 1985.

[8] V.J. Rayward-Smith, editor. *Applications of Modern Heuristic Methods*. Alfred Waller Limited, Henley-on-Thames, UK, 1995.

[9] V.J. Rayward-Smith. *A Unified Approach to Tabu Search, Simulated Annealing and Genetic Algorithms*, pages 17–38. In [8], 1995.

[10] V.J. Rayward-Smith, I.H. Osman, C.R. Reeves, and G.D. Smith, editors. *Modern Heuristic Search Methods*. John Wiley and Sons, 1996.

[11] Colin R. Reeves. *Modern Heuristic Techniques*, pages 1–25. In Rayward-Smith et al. [10], 1996.

[12] Luca Trevisan. *Reductions and (Non-)Approximability*. PhD thesis, Universitá degli Studi di Roma "La Sapienza", 1997.

[13] P.A.S. Veloso and S.R.M. Veloso. Problem decomposition and reduction: Applicability, soundness, completeness. In R. Trappl, editor, *Progress in Cybernetics and Systems Research*, volume 8, pages 199–203. Hemisphere Publ. Co., 1981.