Discrete Relaxation for Matching Relational Structures

LES KITCHEN AND AZRIEL ROSENFELD, FELLOW, IEEE

Abstract—**Local constraint analysis ("discrete relaxation") is used to reduce ambiguity in matching pairs of relational structures. It is found empirically that if the set of possible local properties is sufficiently large, this generally results in unambiguous identifications after only a few iterations.**

Preface

In the past, "relaxation" methods have usually been applied to the problems of identifying points in a fixed simple structure (namely, the pixels in a digital picture array). This correspondence addresses the more general problem of identifying objects which may be parts of an arbitrary structure. Such a situation arises in the higher level processing of images, where the objects may correspond to regions of an image, extracted by some segmentation process. Regions would be linked by such relations as "is above," "surrounds," or "is larger than."

The first two sections of this correspondence establish some theoretical results concerning the effectiveness of relaxation in such situations. The third section demonstrates the application of the method to some simple but not unreasonable problems.

I. Relational Structures and Monomorphisms

Many authors have used finite relational structures as a formalism for describing visual scenes and for the recognition of known objects in such scenes. (For example, Barrow and Popplestone [2], Barrow *et al.* [1], and Winston [15].) In this section we introduce the notion of a relational structure using a treatment which closely follows Barrow *et al.* [1].

Definition 1: A (finite) *relational structure* is a triple $\langle X, \mathscr{P}, \phi \rangle$. X is a finite set of *nodes*, called the *carrier* of the relational structure. $\mathscr{P}$ is a finite set of *predicates*. Each predicate P has associated with it a positive integer called the *order* of the predicate. We write $P^{(n)}$ if P has n arguments. ϕ is a function which maps each predicate $P^{(n)} \in \mathscr{P}$ to an n-ary relation on X. That is, $\phi(P^{(n)}) \subseteq X^n$.

For $x_1, x_2, \cdots, x_n \in X$, $P^{(n)} \in \mathscr{P}$ we write $P^{(n)}(x_1, x_2, \cdots, x_n)$ precisely when $(x_1, x_2, \cdots, x_n) \in \phi(P^{(n)})$. Conventionally, a relational structure is referred to by the name of its carrier, its two remaining components being left implicit.

Definition 2: Let M and W be two relational structures with the same predicate set $\mathscr{P}$. A one to one mapping f: $M \to W$ is called a *monomorphism* of M into W (written f: $M \leq W$) when for all $P^{(n)} \in \mathscr{P}$ and $x_1, x_2, \cdots, x_n \in M$, $P^{(n)}(x_1, x_2, \cdots, x_n)$ implies $P^{(n)}(f(x_1), \cdots, f(x_n))$. That is, f preserves the structure of M. If there exists a mapping f: $M \leq W$, we will often write merely $M \leq W$, the mapping f being left implicit.

The set M can be thought of as a *model* which describes some object of interest, while W can be thought of as a description of a *world*, or *universe*, in which we are searching for an instance of this object. Thus the problem of finding the monomorphisms between two relational structures is of some importance. In general, this problem is computationally very difficult. (It is, in fact, nonpolynomial (NP)-complete, since it is a generalization of the subgraph matching problem for graphs, a problem which is known to be NP-complete [9].) However, for those relational structures which are commonly encountered, we would hope that the problem is not so intractable. That is, while the worst case behavior of an algorithm for finding monomorphisms may be exponential, "on the average" it may find solutions quite rapidly.

Manuscript received October 4, 1978; revised August 31, 1979. This work was supported by the U.S. Army Night Vision Laboratory under Contract DAAG53-76C-0138 (DARPA Order 3206).

The authors are with the Computer Science Center, University of Maryland, College Park, MD 20742.

II. Discrete Relaxation

The term "relaxation" has been used to describe a class of parallel iterative methods for classification or constraint analysis (Davis and Rosenfeld [4], Rosenfeld [10], Rosenfeld *et al.* [11], and Zucker *et al.* [16]). All of these methods have the following in common: we wish to calculate a value of some sort for each point in a structure. This is done by first assigning to each point an initial approximation to this value. Then the approximation at each point is improved (in some sense) by examining the values of the approximations at its neighbors in the structure. This improvement step can be repeated until some condition is satisfied, typically until no further improvement can be made. The "values" referred to need not be numeric; they may be symbolic labels of some sort, and an "approximation" to such a value might be a list of possible labels, each with some sort of likelihood measure attached. The "structure" referred to is often the grid of pixels which represents a digital picture, and in this case the "neighbors" of a point would be its adjacent pixels on the grid. This is by no means intended as a definitive description of relaxation methods, but merely to indicate informally the type of processing which all these methods have in common.

The same type of approach can be applied to the problem of matching relational structures and is developed below. Assume that all relational structures mentioned have the same predicate set $\mathscr{P}$.

Definition 3: An *assignment* of a relational structure M to a relational structure W (the order being material) is any subset of $M \times W$. Thus an assignment of M to W is a pairing of elements of M with elements of W. It is in a sense a binary relation between M and W but should not be confused with those relations that are responsible for the internal structures of M and W.

Definition 4: Let M be a relational structure and $x \in M \times W$. The *neighborhood* of x (written $\mathrm{Nbd}(x)$) is the set

$$\{y \in M: \ \exists P^{(n)}, x_1, x_2, \cdots, x_n, i, j \text{ such that } P^{(n)}(x_1, x_2, \cdots, x_n) \text{ and } x = x_i \text{ and } y = x_j\}.$$

The neighborhood of x consists of all those nodes which are directly related to x in any way by a predicate.

Definition 5: Let R be an assignment of M to W. A pair $(x, y) \in M$ is said to be *locally consistent with respect to R* when there exists a monomorphism f: $\mathrm{Nbd}(x) \leq \mathrm{Nbd}(y)$ such that $f(x) = y$, and $f \subseteq R$, considered as a set of ordered pairs. In other words, R permits a mapping which preserves the local structure around x and y.

Definition 6: An assignment R is said to be *locally consistent* when every pair in R is locally consistent with respect to R. The notion of a locally consistent assignment is considerably weaker than that of a monomorphism. The directed graphs of Fig. 1 furnish an example. The assignment which pairs every node of the "triangle" with every node of the "square" is locally consistent, but there can be no monomorphism between the two. However, local consistency is a "natural" generalization of monomorphism, as the following theorem shows. Any mapping from M to W, where

0018-9472/79/1200-0869$00.75

Fig. 1. Many locally consistent assignments, but no monomorphism.

M and W are relational structures, can be regarded as an assignment of M to W, since a mapping is merely a particular type of pairing. Monomorphisms are precisely those mappings which are injective (i.e., one-to-one) and are consistent as assignments.

Theorem 1: Let M and W be relational structures, and let f: $M \rightarrow W$ be injective. The f is a monomorphism if and only if f is locally consistent.

Proof: Sufficiency—If f is a monomorphism, then clearly f must be locally consistent. Necessity—Assume that f is locally consistent. Let $P^{(n)} \in \mathscr{P}$, and let $x_1, x_2, \cdots, x_n \in M$, such that $P^{(n)}(x_1, x_2, \cdots, x_n)$. Since f is locally consistent, in particular about x_1, there exists a monomorphism f_1: $\text{Nbd}(x_1) \leq \text{Nbd}(f_1(x_1))$ where $f_1 \subseteq f$. Thus we have $P^{(n)}(f_1(x_1), f_1(x_2), \cdots, f_1(x_n))$. However, f is a function, so the image of every point in M is uniquely determined. Therefore, $f_1(x_1) = f(x_1)$, $f_1(x_2) = f(x_2)$, etc., and hence $P^{(n)}(f(x_1), f(x_2), \cdots, f(x_n))$. We conclude that f is a monomorphism. QED

We now display an abstract version of the discrete relaxation algorithm, applied to two relational structures M and W:

$R_0 := M \times W$; $\quad i := 0$;
repeat
$i := i + 1$;
$R_i :=$ {all pairs in R_{i-1} which are locally consistent with respect to R_{i-1}}
until $R_i = R_{i-1}$;
$R^* := R_i$.

This algorithm at each step removes all pairs from the local assignment which are not locally consistent.

It is easy to see how this fits into the relaxation framework. The structure with which we are dealing is the relational structure W. Each assignment can be regarded as attaching a (possibly empty) set of model nodes from M as labels on each world node in W. The improvement step consists of removing those labels which are locally inconsistent. The following results are generalizations of those in Rosenfeld *et al.* [11].

Theorem 2: The above algorithm always terminates.

Proof: At the end of each iteration of the algorithm either $R_i = R_{i-1}$ or $R_i \subset R_{i-1}$. If $R_i = R_{i-1}$ the algorithm terminates. Since $R_0 = M \times W$ is a finite set, there can only be a finite number of iterations for which $R_i \subset R_{i-1}$. QED

Theorem 3: The assignment R^* produced by the above algorithm is locally consistent.

Proof: Suppose the algorithm terminates after k iterations, that is, $R^* = R_k$. However, $R_k = \{$all pairs in R_{k-1} which are locally consistent with respect to $R_{k-1}\}$.

But $R_k = R_{k-1}$, since the algorithm terminated at this step. So all pairs in R_k are locally consistent with respect to R_k, hence $R_k = R^*$ is locally consistent. QED

Theorem 4: R^* is the maximal locally consistent assignment of M to W. That is, if R is any locally consistent assignment of M to W, then $R \subseteq R^*$.

Proof: (by induction on the number of iteration steps).

Basis: $R \subseteq R_0 = M \times W$, by definition.

Induction: Suppose $R \subseteq R_{i-1}$ for some $i > 0$. Let (x, y) be any pair in R. This pair is locally consistent with respect to R, therefore it is locally consistent with respect to R_{i-1} since $R \subseteq R_{i-1}$. Thus $(x, y) \in R_i$ by the iteration step of the algorithm.

Conclusion: We see that $R \subseteq R_i$ for all i, and in particular $R \subseteq R^*$.

Corollary: If f is any monomorphism f: $M \leq W$, then $f \subseteq R^*$.

Thus the discrete relaxation algorithm is guaranteed to capture any monomorphisms that exist between two relational structures. In general, R^* can contain other pairings as well. Fig. 1 again illustrates a pathological case where $R_0 = M \times W$ is locally consistent. However, it seems that in practice the only locally consistent assignments are usually in fact monomorphisms (or trivially the empty assignment). This is borne out by the experiments described below. In those cases where ambiguity remains, a simple case analysis would suffice to sort out the monomorphisms embedded in R^*. Even though discrete relaxation *may* gain nothing at all, it seems in practice to assist considerably in finding monomorphisms between relational structures. The reader should note that if there are several distinct monomorphisms between two relational structures M and W, then at best R^* will contain the union of all these monomorphisms, and there will still remain the problem of disentangling the individual monomorphisms. (However, this problem is trivial if the monomorphisms are disjoint.)

Note that in order to determine whether a given pair (x, y) is locally consistent with respect to an assignment R_{i-1}, we must still conduct a search for a monomorphism f: $\text{Nbd}(x) \leq \text{Nbd}(y)$, with the other required properties, $f \subseteq R_{i-1}$ and $f(x) = y$, and this search must be repeated for every pair in R_{i-1}. Compared to a tree-search procedure for detecting monomorphisms, we have traded one large combinatorial search for many small combinatorial searches. This would seem to make discrete relaxation considerably faster. Furthermore, the local consistency checks for all the pairs in R_{i-1} are independent and could, therefore, be carried out in parallel, if suitable hardware were available. It is possible to use a weaker characterization of consistency which obviates the search for a monomorphism between neighborhoods.

Definition 7: Let M and W be relational structures, and let R be an assignment of M to W. A pair $(x, y) \in M \times W$ is said to be *weakly consistent* with respect to R when there exists an assignment Q of $\text{Nbd}(x)$ to $\text{Nbd}(y)$ with the following properties.

1) $Q \subseteq R$.
2) $(x, y) \in Q$.
3) $(x, z) \in Q$ implies $y = z$.
4) For all $P^{(n)}$, and for all $x_1, x_2, \cdots, x_n \in \text{Nbd}(x)$, $P^{(n)}(x_1, x_2, \cdots, x_n)$ implies that there exist $y_1, y_2, \cdots, y_n \in W$ such that $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n) \in Q$ and $P^{(n)}(y_1, y_2, \cdots, y_n)$. That is, for every predicate instance in the model which mentions x, R permits a renaming of the arguments so that the predicate holds in the world. This renaming need not be one-to-one, nor even a function, but x must always be replaced by y.

Weak consistency can be used instead of local consistency in a discrete relaxation process, and corresponding theorems can be proved. However, the results it produces are more likely to be ambiguous, in that weak consistency will permit pairings which are locally inconsistent.

The basic technique just described is not new. Calling it "refinement," Ullman [13] applied it to the simpler problem of subgraph isomorphism. Recently, Haralick and Shapiro [6] have presented a general theory for constraint satisfaction problems. Our approach differs from theirs in that 1) it admits the treatment of distinct predicates of different orders; 2) the mutual constraint between nodes is treated in terms of variable-sized neighbor-

hoods, rather than fixed-sized tuples; and 3) the consistency of a labelling is implicitly determined by the current assignment and the local structure of the model and world, not by an explicit set of permitted tuple labellings. It can be fitted into their framework by choosing a tuple size equal to that of the largest neighborhood in the world and extending smaller neighborhoods with arbitrary world nodes whose labelling is unrestricted. However, this construction is rather artificial since it ignores the natural structure of such problems and is thus mainly of theoretical interest. (For weak consistency, one must choose a tuple size equal to the greatest predicate order.)

As was previously mentioned, if ambiguity remains after the application of relaxation, it may be necessary to use a case analysis. Relaxation can be usefully integrated into such an analysis in the following way: select a world node with several possible labels; partition these labels into several disjoint subsets (which may be singletons); for each subset create a new assignment identical to the original except that the chosen world node is paired only with labels from the subset; and apply relaxation to each of these new assignments. If ambiguity remains, this procedure can be applied recursively. Using such a technique, Mackworth [7] and Haralick and Shapiro [6] have integrated relaxation-like methods with a backtracking or depth-first tree search. MSYS (Barrow and Tenenbaum [3]) combines a fuzzy relaxation method with a best-first tree search.

III. Experiments

A computer program has been written which implements the discrete relaxation algorithm described. There are, however, several minor differences. The initial iteration is anomalous. Since $R_0 = M \times W$, we can check the local consistency of pairs in R_0 "on the fly" by examining all pairs systematically, without the need to store them. Furthermore, since on this iteration many pairs are examined, the program uses the simpler weak consistency condition. This does not invalidate any of the results proved in the last section but is easier to test. All subsequent iterations use the full local consistency condition. Obviously, the whole sequence of assignments $R_1, R_2, \cdots, R^*$ need not be stored; only the current version is stored. On each iteration those pairs in the current assignment which are not locally consistent are flagged, and at the end of each iteration these flagged pairs are deleted from the assignment. This means that the algorithm terminates whenever no deletions are made on a given iteration.

At first the program was tried in an *ad hoc* fashion on the matching of various relational structures, some concocted by hand, others based on the adjacencies of states in political maps. On these the program generally performed quite well, and encouraged by these results we conducted the more systematic tests described below.

The program was modified to generate random substructures of a relational structure read as input. The following method was used. An element of the structure is chosen at random, and used to start a list. Then repeatedly an element on the list is chosen at random which has neighbors *not* on the list; next, one of its neighbors *not* on the list is chosen at random and placed *on* the list. This continues until the list reaches some prespecified size. The list is then used as the carrier of a relational structure which inherits from the original relational structure all those predicates which have as arguments only elements which appear on the list. This process tends to generate compact substructures, embedded in a larger relational structure, and thus corresponds to the intuitive notion of compact objects embedded in a large visual scene. The program then attempts to match the substructure generated as model with the larger parent structure as world. In this case a monomorphism certainly exists, so we can measure how often the program finds it and how much computation the relaxation takes.

The data for the first group of tests were derived from the political map of Africa, which has 46 countries. The map is described in terms of the adjacencies of the countries, together with some of the following properties:

a) whether a country is coastal or inland,
b) the number of letters in its name,
c) the first letter of its name,
d) its color on the globe (National Geographic Society, 1976),
e) the first letter of the name of its capital city.

We thus have a relational structure with one binary predicate, *adjacent* (196 instances), and a large set of possible unary predicates (46 instances for each of the properties a)–e)).

Tests were made using the adjacency data together with various subsets of the properties. All tests were run on the same randomly generated models of sizes ranging from four to 40 nodes using five examples of each size.

The results of these tests are summarized in Tables I–V. In each of these tables the column headed "Iterations" gives the average number of iteration steps required for convergence (including the final iteration which determines that the assignment has stabilized). The column headed "Comparisons" gives the average number of times (000 omitted) that it was necessary to compare one predicate with another in the matching; thus it is an indication of the computational effort required by the relaxation process. The column headed "Ambiguity" measures the average number of excess pairs remaining after convergence is complete; it is zero if there is exactly one pair for every element in the model. All of these averages are taken over the five instances of each model size. (The predicates used are indicated in the table captions; the adjacency relation was also used in every test.)

A second similar group of tests was run, using corresponding data derived from the 48 contiguous states of the U.S.A. (203 instances of the binary predicate *adjacency*, and for each of the properties a)–e), 48 instances of unary predicates).

A version of the program which uses weak consistency for all iterations was also tried. The results of using this version on the same data as used in Table I are shown in Table II.

Limitations on computer resources precluded systematic experiments with larger or more varied problems. However, single tests were run with three larger problems. Two were isomorphism problems: to match the entire relational structure of Africa as used for Table III against itself (46 nodes, 92 unary and 196 binary predicate instances) and similarly for the U.S.A. structure used for Table VIII (48 nodes, 96 unary and 203 binary predicate instances). For the third problem the same structures were matched against each other (Africa as model, U.S.A. as world). This was a problem for which no monomorphism was expected, and as anticipated, the relaxation converged quickly to the empty assignment, indicating that, in fact, no such monomorphism existed. Two other problems were isomorphism problems for relational structures drawn from other sources. The first used the chemical structure of β-codeine $C_{18}H_{21}O_3N$ [14]. The atoms were represented by 43 nodes and classified by 43 unary predicate instances as being carbon, hydrogen, oxygen, or nitrogen. The 40 single chemical bonds were represented by 40 instances of a binary relation. The one double bond was represented by one instance of another binary relation, and the benzene ring was represented by an instance of an order-six relation. The second problem was based on the switching circuit diagram of a binary coded decimal

TABLE I
FIRST LETTER OF NAME (AFRICA)

Model Size	Iterations	Comparisons	Ambiguity
4	3.4	4.2	0
5	3.6	7.0	0.6
6	4.0	8.6	0.2
7	3.8	11.1	1.2
8	4.0	11.0	1.0
9	3.8	14.4	0.4
10	4.0	17.2	1.2
20	4.2	42.1	1.2
30	4.0	67.2	1.2
40	4.0	93.6	1.6

TABLE II
FIRST LETTER OF NAME—COASTAL/INLAND (AFRICA)

Model Size	Iterations	Comparisons	Ambiguity
4	3.4	5.2	0
5	3.4	8.1	0
6	3.4	10.3	0
7	3.4	12.7	0.4
8	3.6	13.4	0.6
9	3.4	16.9	0.2
10	3.6	18.7	0.8
20	3.4	43.0	0.4
30	3.4	68.9	0.4
40	3.8	96.0	0.8

TABLE III
FIRST LETTER OF NAME—COLOR ON GLOBE (AFRICA)

Model Size	Iterations	Comparisons	Ambiguity
4	2.6	4.9	0.0
5	3.0	7.6	0.0
6	2.8	9.8	0.0
7	3.0	12.0	0.2
8	3.0	12.5	0.0
9	3.0	16.1	0.2
10	3.0	17.5	0.4
20	3.0	42.0	0.6
30	3.0	66.4	0.4
40	3.0	92.1	0.4

TABLE IV
FIRST LETTER OF NAME OF CAPITAL CITY (AFRICA)

Model Size	Iterations	Comparisons	Ambiguity
4	3.2	4.3	1.4
5	3.6	6.7	1.0
6	3.6	8.3	0.2
7	3.8	10.4	0.6
8	3.6	11.8	1.2
9	4.0	16.2	1.2
10	3.8	16.3	1.8
20	4.2	44.2	1.8
30	4.0	85.8	2.0
40	4.0	109.5	2.2

TABLE V
FIRST LETTER OF NAME OF CAPITAL CITY—LENGTH OF NAME (OF COUNTRY) (AFRICA)

Model Size	Iterations	Comparisons	Ambiguity
4	2.4	4.9	0
5	2.6	7.6	0.2
6	2.8	9.8	0
7	3.6	12.4	0.2
8	3.4	12.8	0.2
9	3.4	16.6	0.2
10	3.2	17.8	0.4
20	3.4	43.1	0.0
30	3.8	70.4	0.0
40	4.0	97.3	0.0

TABLE VI
FIRST LETTER OF NAME (U.S.A.)

Model Size	Iterations	Comparisons	Ambiguity
4	4.0	4.9	0.2
5	3.4	5.8	1.0
6	4.0	7.9	0.2
7	4.2	9.3	0.2
8	3.6	12.1	0.6
9	4.0	14.0	1.4
10	3.8	14.4	1.0
20	4.0	37.0	0.6
30	4.0	60.6	0.2
40	4.0	80.2	1.2

TABLE VII
FIRST LETTER OF NAME—COASTAL/INLAND (U.S.A.)

Model Size	Iterations	Comparisons	Ambiguity
4	3.8	6.0	0.0
5	3.2	7.0	1.0
6	3.6	9.5	0.2
7	3.8	11.2	0.2
8	3.6	14.7	0.4
9	3.8	16.8	0.8
10	3.6	17.3	1.0
20	4.0	45.2	0.2
30	4.0	73.8	0.0
40	4.0	97.4	1.2

TABLE VIII
FIRST LETTER OF NAME—COLOR ON GLOBE (U.S.A.)

Model Size	Iterations	Comparisons	Ambiguity
4	2.8	5.7	0.2
5	3.2	6.8	0.0
6	3.0	9.2	0.0
7	3.2	10.6	0.0
8	3.2	14.3	0.2
9	3.0	15.9	0.0
10	3.0	16.8	0.2
20	3.0	41.9	0.4
30	3.0	68.5	0.2
40	3.0	90.3	0.0

TABLE IX
FIRST LETTER OF NAME OF CAPITAL CITY (U.S.A.)

Model Size	Iterations	Comparisons	Ambiguity
4	3.4	4.8	0.4
5	3.6	5.8	1.0
6	3.6	7.7	0.2
7	3.6	8.6	0.4
8	3.6	12.1	0.6
9	3.8	13.3	0.0
10	4.2	14.3	0.4
20	4.2	36.7	0.8
30	3.4	58.3	0.2
40	4.0	79.2	0.0

TABLE X
FIRST LETTER OF NAME OF CAPITAL CITY—LENGTH OF NAME (OF STATE) (U.S.A.)

Model Size	Iterations	Comparisons	Ambiguity
4	2.8	5.7	0.0
5	2.8	6.7	0.0
6	3.0	9.2	0.0
7	3.0	10.5	0.0
8	3.0	14.2	0.0
9	3.0	15.8	0.0
10	3.0	16.8	0.0
20	3.0	41.5	0.0
30	3.0	67.8	0.0
40	3.0	89.5	0.0

TABLE XI
FIRST LETTER OF NAME (WEAK CONSISTENCY—AFRICA)

Model Size	Iterations	Comparisons	Ambiguity
4	3.4	4.1	0.0
5	3.8	6.4	0.6
6	4.0	8.2	0.2
7	4.0	10.2	1.2
8	4.0	10.4	1.0
9	4.2	13.5	0.4
10	4.4	14.9	1.2
20	5.0	35.8	1.2
30	4.4	55.5	0.8
40	4.8	77.8	1.6

(BCD) to seven-segment decoder, SN5446A [12]. The 58 nodes were components, classified by 59 instances of various unary predicates as input terminals, output terminals, drivers, inverters, and gates, nor gates, or nand gates (the last three were subclassified according to the number of inputs). Note that one component served as both an input and output terminal. The connections were represented by 105 instances of a binary predicate indicating that the output of its first argument is connected to an input of its second argument.

The results of these tests are shown in Table XII (using local consistency) and in Table XIII (using weak consistency). It can be seen from the tables that, for these problems, the results of the relaxation usually show very little ambiguity and convergence tends to be quite rapid (three to four iterations). What ambiguity remained was most often due to the existence of multiple monomorphisms for the chosen model. (That is, the relaxation almost always produced the theoretically best possible answers.) The experiments with Africa and the U.S.A. indicate that the number of iterations does not significantly depend on model size. The

TABLE XII
ASSORTED STRUCTURE MATCHING PROBLEMS USING LOCAL CONSISTENCY

Model	World	Iterations	Property Comparisons	Ambiguity
AFRICA	AFRICA	3	34·3	0
U.S.A.	U.S.A.	3	31·5	1
AFRICA	U.S.A.	3	24·6	0
β-CODEINE	β-CODEINE	5	17·9	20
SN5446A	SN5446A	5	361·3	2

TABLE XIII
ASSORTED STRUCTURE MATCHING PROBLEMS USING WEAK CONSISTENCY

Model	World	Iterations	Property Comparisons	Ambiguity
AFRICA	AFRICA	3	33·3	0
U.S.A.	U.S.A.	3	30·9	1
AFRICA	U.S.A.	3	24·4	0
β-CODEINE	β-CODEINE	5	11·4	31
SN5446A	SN5446A	8	50·8	2

number of comparisons required does increase with the model size but only at an essentially linear rate. Relaxation using weak consistency is a little cheaper but can produce more ambiguous results (see β-codeine in Tables XII and XIII).

IV. Discussion and Extensions

The results presented here should be considered to be of a preliminary and exploratory nature. We have shown that relaxation performs well for matching problems similar to those encountered in scene analysis. While we have made no attempt to compare relaxation with any other methods, such a study would be worthwhile, especially considering the empirical results of Gaschnig [5]. He found that a Waltz-type filtering algorithm performed more poorly than backtracking methods when applied to the N-queens problem. However, as Gaschnig points out, the N-queens problem is very tightly constrained (in that the position of one queen can be affected by all the other queens on the board). The structures considered here tend to be loosely constrained in that each node is directly constrained by only a few others. Also, Gaschnig was primarily interested in the effort required to find a first solution, rather than all solutions. These differences suggest that there may be many problems for which relaxation is the preferred technique.

Some improvements could be made in the present program. First of all, attention should be given to the order in which predicates are checked when testing for local consistency. If rarer predicates are examined first, it is more likely that a local inconsistency will be discovered early. Second, since unary predicates can affect local consistency only on the first iteration of relaxation, they should be disregarded on all subsequent iterations. Third, provisions should be made for the special treatment of commonly occurring types of relations. For example, each instance of a symmetric binary relation (like *adjacency* in the previous section) must be stored twice, once in each sense ("A is adjacent to B" and "B is adjacent to A"). Special handling of symmetric relations could reduce space and processing requirements considerably.

Beyond this, extensions can be made both to the notion of relational structures and to the application of relaxation techniques. A first step is the introduction of quantitative predicates, which will be treated in a subsequent paper. This extension is crucial if our methods are to be applied to real-world scene analysis problems.

Acknowledgment

The authors would like to thank Mrs. Virginia Kuykendall for her help in preparing this correspondence.

References

[1] H. G. Barrow, A. P. Ambler, and A. M. Burstall, "Some techniques for recognizing structures in pictures," in *Frontiers of Pattern Recognition*, S. Watanabe, Ed. New York: Academic, 1972, pp. 1–29.
[2] H. G. Barrow and R. J. Popplestone, "Relational descriptions in picture processing," in *Machine Intelligence 6*, B. Meltzer and D. Michie, Eds. New York: Elsevier, 1971, pp. 377–396.
[3] H. G. Barrow and J. M. Tenenbaum, "MSYS: A system for reasoning about scenes," S.R.I., A.I. Center, Menlo Park CA, Tech. Note 121, Apr. 1976.
[4] L. S. Davis and A. Rosenfeld, "Hierarchical relaxation for waveform parsing," Comput. Sci. Center, Univ. Maryland, College Park, Tech. Rep. 568, 1977.
[5] J. Gaschnig, "Experimental case studies: Backtrack vs. Waltz-type vs. new algorithms for satisficing assignment problems," in *Proc. 2nd Nat. Conf. Can. Soc. for Computational Studies of Intelligence*, Univ. Toronto, Toronto, Ont., pp. 268–277, July 1978.
[6] R. M. Haralick and L. G. Shapiro, "The consistent labeling problem: Part I," *IEEE Trans. Pat. Anal. and Mach. Intel.*, vol. PAMI-1, no. 2, pp. 173–184, Apr. 1979.
[7] A. K. Mackworth, "Consistency in networks of relations," *Art. Intel.*, vol. 8, pp. 99–118, 1977.
[8] *National Geographic Political Globe*. Washington, DC: National Geographic Society, 1976.
[9] R. C. Read and D. G. Corneil, "The graph isomorphism disease," *J. Graph Theor.*, vol. 1, pp. 339–363, 1977.
[10] A. Rosenfeld, "Iterative methods in image analysis," *IEEE Conf. Pat. Recog. and Image Proc.*, pp. 14–18, 1977.
[11] A. Rosenfeld, R. Hummel, and S. Zucker, "Scene labeling by relaxation operations," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, pp. 420–433, 1976.
[12] *The TTL Data Book*, 2nd ed. Dallas, TX: Texas Instruments, Inc., p. 25, Section VII 1976.
[13] J. R. Ullman, "An algorithm for subgraph isomorphism," *J. Assoc. Comput. Mach.*, vol. 23, pp. 31–42, 1976.
[14] R. C. Weast, Ed., *Handbook of Chemistry and Physics*, 49th ed. Cleveland, OH: Chemical Rubber, 1968, p. C-257.
[15] P. H. Winston, "Learning structural descriptions from examples," in *The Psychology of Computer Vision*, P. H. Winston, Ed. New York: McGraw-Hill, 1975, pp. 157–210.
[16] S. W. Zucker, E. V. Krishnamurthy, and R. L. Haar, "Relaxation processes for scene labeling: Convergence, speed and stability," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, pp. 41–48, 1978.

The Chinese and Japanese Abaci

ALLAN G. BROMLEY

***Abstract*—An analysis is made of the number of finger movements in addition operations upon the Chinese and Japanese forms of the Abacus. This shows that the additional and redundant digit representations possible on the Chinese abacus by virtue of the greater number of beads on each rod yield no significant reduction in finger movements over the Japanese abacus. An informal analysis of the less common multiplication and division operations shows the Japanese abacus with modern methods of usage to be no less efficient than the Chinese. The sociological pressures that led to the evolution of the modern Japanese abacus have not detracted in any way from its computational efficiency.**

Manuscript received April 16, 1979; revised August 20, 1979.

The author was with the Basser Department of Computer Science, University of Sydney, N.S.W. Australia. He is now with the Science Museum, South Kensington, London, England, SW7 2DD.