RDS-TR-13-82

THE GRAPH LABELING PROBLEM AND THE RELAXATION LABELING PROCESSES

M. D. Diamond

October 1982

CENTER FOR ROBOTICS AND INTEGRATED MANUFACTURING

Robot Systems Division

COLLEGE OF ENGINEERING

THE UNIVERSITY OF MICHIGAN

ANN ARBOR, MICHIGAN 48109

This work was supported in part by the Ultrasonic Imaging Laboratory and the Robot Systems Division of the Center for Robotics and Integrated Manufacturing (CRIM) at The University of Michigan, Ann Arbor, MI. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the funding agencies.

Abstract

This document presents a survey of the current literature on the graph labeling problem and the relaxation labeling processes. This survey serves as a base from which current research into cooperative solutions to the continuous graph labeling problem can proceed. Our intent is to motivate the reader by presenting herein the development and overview necessary to introduce the topic, as well as the the notation and analysis necessary to create the framework in which current and proposed work can be discussed.

1. Introduction

The relaxation labeling processes (RLPs) refer to a class of cooperative algorithms which can be applied to problems expressed in terms of the labeling of a graph. A graph labeling problem is one in which a unique label, λ, from a set, Λ, of possible labels must be assigned to each node of the graph. The assignment must be performed given incomplete local information about the correct labeling at each vertex, and contextual information about the interaction of labels on adjacent vertices. Two forms of labeling, discrete and continuous, are distinguished by the nature of the local information and the representation of the contextual information. In a discrete labeling problem the local information consists of a subset of labels associated with each vertex from which the correct label for that vertex must be chosen. The contextual information is expressed in terms of constraint relations which make explicit those pairs of labels which can occur simultaneously on adjacent nodes. In a continuous labeling problem, the local information is composed of likelihood measures or figures of merit given for each label on each vertex, and the contextual information is composed of measures of compatibility between pairs of labels on adjacent vertices. In the latter case, both the figures of merit and the compatibilities are assumed to be taken from a continuous scale.

Relaxation techniques for discrete labeling problems were discovered first. They were derived from early work in computer vision, specifically, Waltz's filtering algorithm [Wal72] for the implementation of the Huffman-Clowes line labeling scheme [Huf71,Clo71]. Although Waltz described a sequential search, subsequent work [Mon74,RHZ76] cast Waltz filtering in terms of a set of parallel, iterative equations; that is, as a cooperative process. The fundamental idea in this technique is to reduce the ambiguity in a labeling by removing those labels which are not compatible with the labeling on adjacent vertices. Removing any label at one time step will affect the removal of labels at adjacent vertices at the next time step, so the effect is seen as propagating through the extent of the network. Although not all aspects of the

behavior of discrete relaxation networks are currently understood, it is known that they will converge to a fixed point which can easily be described in terms of the constraint relations and the network topology, and thus related directly to the requirements imposed by a particular problem.

To treat the continuous case, extensions to the form of the iterative equations for the discrete relaxation processes were made, which incorporate the strength measures associated with the labels on adjacent vertices and the compatibility information into an updated strength measure for a given label on a given node [RHZ76,PEL80,KIR80,FaB81]. This extension has, however, proven to be difficult: the convergence properties of existing algorithms have generally been poor, and there is apparently no understanding of the functions they compute. An examination of the literature points to the fact that there has been no real attempt at a formal analysis of the problem domain and no global perspective on what the techniques are designed to accomplish. Thus, the updating algorithms which have emerged so far have been based almost entirely on heuristics.

Despite the apparent lack of theory, the application of continuous relaxation labeling techniques to problems in scene analysis and pattern recognition has generated a large volume of literature in recent years [DaR80,Ros79,Ros81]. In fact, the graph labeling model is quite robust, and suitable for a wide range of problems in pattern recognition. The parallel nature of the solution algorithm, and the implications this has for implementation in hardware seems also to have had some effect on its growth [Wil78]. Finally, evidence that relaxation like mechanisms can be used to explain certain phenomena in human visual perception [WeM78,MoW79,MaP76] has further contributed to the popularity of this topic. It seems evident then that the relaxation labeling techniques will play an important role in the developing fields of artificial intelligence and pattern recognition. The focus of our work, which is the achievement of a formal understanding of the capabilities, and limitations of the relaxation labeling processes, as well as the establishment of a consistent

methodology for their implementation, is therefore, well motivated.

The rest of this document serves to illustrate and expand in detail the ideas expressed above. In section 2 we present a formal development of the relaxation labeling processes. This development includes the introduction of notation, concepts, and results which will be used throughout this report and in subsequent reports on this subject. It will also serve as a survey of the current literature. Section 3 contains an overview of the problem domain. Our intent here is to set the stage for the further work by presenting an analysis of the problem and relating it to problems in well established areas. In section 4 some results related to the discrete relaxation labeling processes and constraint networks will be presented.

2. Development of the Relaxation Labeling Processes

The purpose of this section is to present the theory of relaxation labeling as it currently exist in the literature. We do this by outlining a development of the area, making note of important ideas as well as illustrating the work from which they have been derived. Discrete and continuous relaxation are treated separately here. In subsequent sections issues related to the unification of the two concepts are discussed.

2.1. Discrete Relaxation

The realization that the labeling of segments of an image is equivalent to interpreting the scene which generates that image occurred early in the work in computer vision [Guz68]. A solution to a particular labeling problem, the labeling of line drawings derived from blocks-world scenes was solved by Huffman [Huf71] and (independently) Clowes [Clo71] in 1971. Waltz [Wal72] proposed an algorithm for the implementation of Huffman-Clowes labeling based on the propagation of constraint information. This algorithm is described as an example below. Huffman-Clowes labeling and Waltz's algorithm lead to the development of graph labeling and

constraint propagating networks, which together form the basis for the relaxation labeling processes.

Most of the theory of discrete relaxation which has been established to date appears in a paper by Montanari [Mon74]. Although some further contributions can be found in the work of Freuder [Fre78], Mackworth [Mac77], Rosenfeld et al. [RHZ76], Haralick et al. [HDR78], and Haralick and Shapiro [HaS79,HaS80]. Applications of constraint propagation techniques have been proposed for such areas as game playing [Chu79], problem solving [Sac79,BaT76], theorem proving [Gas74], search strategies [HaE79,HaS79,HaS80], database management [Gro76], graph theory [Mon74,Ull76], syntactic pattern recognition [DaR78], and scene analysis [Wal72,DaR81]. Further applications can also be found in surveys by Davis and Rosenfeld [DaR80], and Haralick and Shapiro [HaS79].

2.1.1. The Huffman-Clowes Line Labeling Scheme

Example 2.1: We consider the situation in which we are viewing a *blocks world scene*, which consists entirely of polyhedral objects on a flat table. We assume that the necessary processing has been carried out in order to extract a *line drawing* or *picture graph* from the given scene. We restrict ourselves further to situations where no more than three edges are incident on a particular vertex of this line drawing. This condition is known as the *trihedral vertex criterion.*

Huffman-Clowes labeling attempts to place one of three possible labels on each line of a line drawing, as is shown in figure 2.1 below. A plus (+) is assigned to edges which separate surfaces forming a convex body, a minus (-) is assigned to edges which separate surfaces forming a concave body, and an arrow (→) is assigned to edges in which one of the intersecting surfaces occludes the other. In the latter case, the arrow is given that direction which imposes a counter-clockwise orientation on the line, taken with respect to the center of the visible face. There are constraints imposed on the labels on adjacent lines, which are determined by the

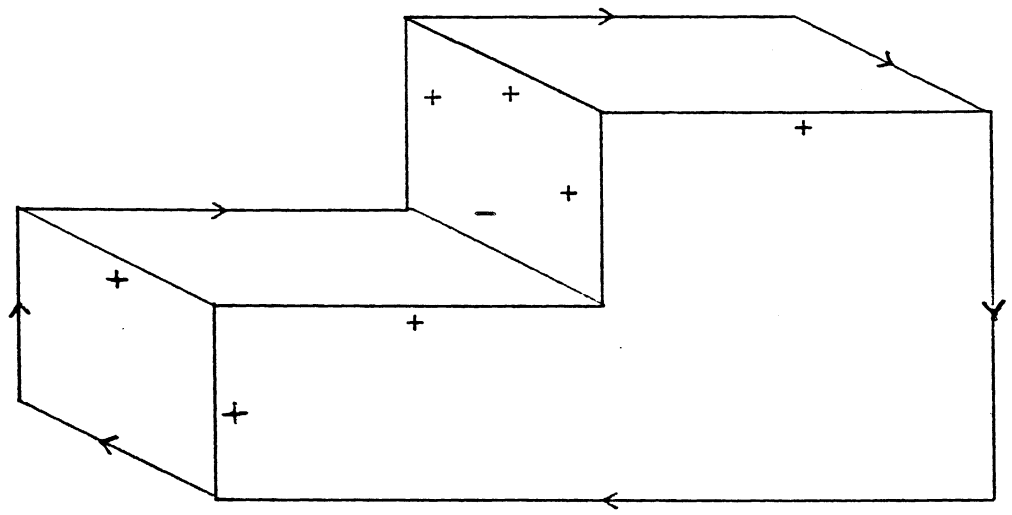

Figure 2.1: Huffman-Clowes labeling of a simple blocks-world scene.

nature of the vertices at which they intersect. To this end the vertices are separated into four classes, based on the angles formed by the lines incident upon them. These classes, which are named the L, fork, arrow, and T joints are illustrated in figure 2.2.

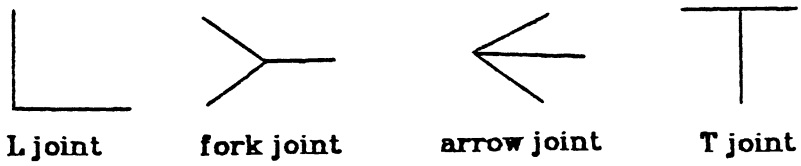

Figure 2.2: Four possible vertices under the trihedral vertex critereon.

Combinatorially, there are 208 ways to label the lines incident on these four types of vertices. In fact, however, only 16 such labelings will occur in natural scenes, as are shown in figure 2.3. This is important for the following reason: A line drawing contains all the information inherent in the blocks world scene from which it was derived. Yet, it is represented by a simple data structure within the computer memory. Inspection of any particular element of this data structure reveals nothing as to the global role it plays:[1] this can be done only once the elements have been given their proper labels. However, this leads to the following paradox: A local labeling can be achieved only by considering the global context in which each

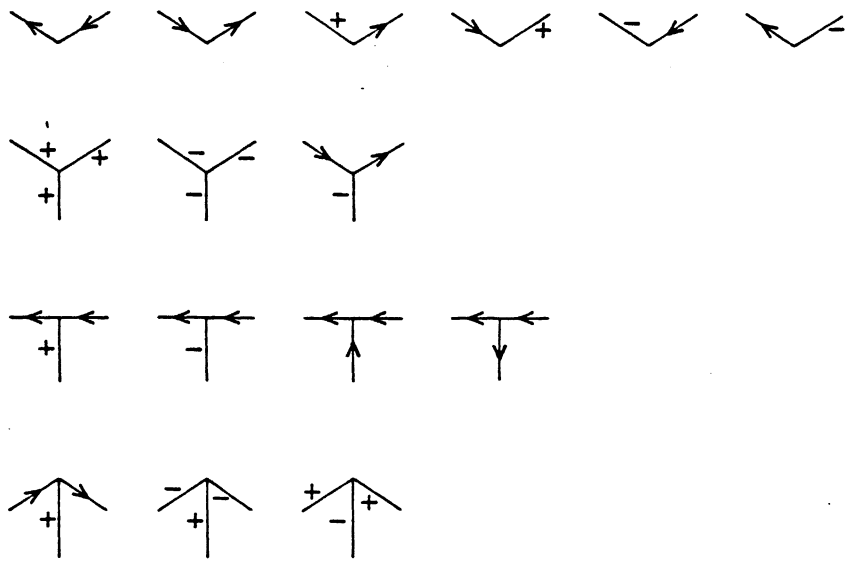

Figure 2.3: Table of the 16 possible vertex labelings for blocks world scenes.

[1] The "role" that is being sought after in Huffman-Clowes labeling is whether a given line separates two distinct objects, or two surfaces of the same object. See [Guz68].

element falls, whereas the global interpretation depends on the local labeling.

This paradox can be resolved, at least in part, by using the constraint information in conjunction with the following procedure:

(1) Start by assigning every possible label to every line incident on each vertex as determined by the vertex type and the list of legal labelings as shown in figure 2.3,

(2) Propagate the labeling information from each vertex to its adjacent vertices and remove labels which are inconsistent with the labeling on these vertices. The removal of a label on a line incident with a given vertex further constrains the labeling of *all* lines incident on that vertex, since it restricts the number of entries in the labeling table (figure 2.3) for that particular *vertex type* which may now be used to describe the current labeling on that vertex.

(3) Repeat step (2) until no more labels can be removed.

The procedure thus described contains the essence of Waltz's implementation of the Huffman-Clowes labeling scheme.

It is important to note how constraints derived from the problem domain are used to restrict the possible interpretation of elements in that domain. Although this idea is relatively simple, its power should not be underestimated. We note, for example, how in the current work towards achieving vision systems capable of human-like performance, the use of natural constraint are continually being brought to bear on a problem that would otherwise be intractable [BaT81,Zuc81].

2.1.2. Labeling, Constraint Networks, and Discrete Relaxation

The intent of this section is to express the ideas inherent in the Huffman-Clowes line labeling scheme and Waltz's filtering algorithm in a formal manner so that they may be further developed and applied to other problem areas. To this end, the concepts of labeling, constraint networks, and discrete relaxation are introduced and the interrelationships of these ideas discussed. Further aspects of discrete

relaxation and constraint networks will be examined in subsequent sections.

A *labeling problem* is a general formalism for a wide class of problems in pattern recognition. The two components of a labeling problem are a set, V, often referred to as *units* or *vertices*, and a set, Λ, known as the *label set*. Our objective is to assign a unique label $\lambda \in \Lambda$ to each vertex $v_i \in V$. In a *graph labeling problem* one is furthermore given a graph, $G = (V,L)$, with vertex set V and edge set $L \subseteq V \times V$. The function of the graph is to impose limitations on the amount of context, or global information, that can be used in a labeling decision at each vertex. In the discrete labeling problem, which is what is being considered here, this limitation is made explicit by associating a *constraint relation*, $R_{ij} \subseteq \Lambda \times \Lambda$ with each edge $v_i v_j \in L$. The constraint relation specifies which pairs of labels (λ,λ') may co-occur on nodes v_i and v_j. Although a more general case may be considered, for the sake of this discussion it is assumed that the constraint relations are symmetric. That is, $(\lambda,\lambda') \in R_{ij}$ if and only if $(\lambda',\lambda) \in R_{ij}$, so that $R_{ij} = R_{ji}$. The graph, label set, and constraint relations are referred to collectively as a *constraint network*. A constraint network, the specification for which is domain dependent, is a model for how information about the problem domain related to a labeling problem is organized. It is often referred to as the *world model*.

Definition 2.1: A *simple constraint network* defined on a label set Λ is a tuple:

$$C_n(\Lambda) = (V,E)$$

where,

$$V = \{v_1, v_2, \ldots, v_n\}$$

is a set of vertices, and,

$$E = \{(e_1,R_1), (e_2,R_2), (e_s,R_s)\},$$

where $e_i \in V \times V$ is an edge, and $R_i \subseteq \Lambda \times \Lambda$ is a constraint relation associated with e_i.

Definition 2.1a: Let $C_n(\Lambda) = (V,E)$ be a constraint network. The associated *coarse graph* is the graph $C_H = (V_H,E_H)$ with $V_H = V$ and $E_H = \{e_1, e_2, \ldots, e_s\}$.

The associated coarse graph defines the network topology. The term *coarse topology* will also be used, the reference to the associated coarse graph being understood.

Definition 2.1b: Let $C_n(\Lambda) = (V,E)$ be a constraint network with $E = \{(e_1,R_1), \ldots, (e_m,R_m)\}$. The *associated fine graph* is a graph $C_h = (V_h,E_h)$ where,

$$V_h = \Lambda \times V$$

and,

$$E_h \subseteq V_h \times V_h = (\Lambda \times V) \times (\Lambda \times V)$$

with $(\lambda,v_i) \times (\lambda',v_j) \in E_h$ if and only if (1) $v_iv_j \in E_H$, *and* (2) $(\lambda,\lambda') \in R_{ij}$.

The associated fine graph has an edge between two vertices $(v_i,\lambda)\times(v_j,\lambda')$ when an edge exists between the corresponding vertices v_i and v_j in the coarse graph and the labels (λ,λ') are in the constraint relation for the edge v_iv_j. As such it contains the same information as the specification for the constraint network as given by definition 2.1. Fine graphs will, however, be useful both as a model and in illustrating the explicit relationships between labels on adjacent nodes in the some of the examples discussed below.[2]

Note the use of the term "simple" in the definition for a constraint network. In the sequel, the issue of the amount of "context" required to solve a particular problem will be raised. To this end, we will want to extend the concept of the constraint network to the case where the coarse topology is represented by an hypergraph [Ber73], that is, as a graph where a given edge may have an arbitrary number of vertices.

A constraint network is applied to a specific problem by associating with each vertex v_i a label set $\Lambda_i \subseteq \Lambda$ which represents the current labeling of that vertex. In

[2]It is suggested that examples 2.1 and 2.2 and figures 2.4 through 2.7 be referred to to illustrate these definitions.

the relaxation algorithm to be described below, the labeling at each node will change with time. Thus, the labeling at time t will be denoted by Λ_i^t and $\{\Lambda_i^t, i = 1,...,n\}$ will denote the labeling on, or the *state* of the constraint network at time t. An *initial labeling*, $\{\Lambda_i^0\}$ is assumed to have been generated by some process which can detect the presence of one of several features, but cannot unambiguously determine which of those features is the correct one for a given node. A labeling, $\{\Lambda_i\}$, is called *ambiguous* if $|\Lambda_i| > 0$ for all i, and there exists an i such that $|\Lambda_i| > 1$. The purpose of the relaxation algorithm is to refine, or disambiguate, this initial estimate by using the context of a labeling Λ_i to remove those labels which are not consistent with the world model.

Definitions 2.2: Given a constraint network $C_n(\Lambda)$, a labeling Λ_i on vertex v_i is called *consistent* with the labeling Λ_j on vertex v_j if for all $\lambda \in \Lambda_i$ there exists a label $\lambda' \in \Lambda_j$ such that $(\lambda,\lambda') \in R_{ij}$. Furthermore, Λ_i is called *consistent with respect to its neighborhood* if for all v_j adjacent to v_i, Λ_i is consistent with respect to Λ_j. Finally, a *labeling* $\{\Lambda_i\}$ is called *consistent* if Λ_i is consistent with respect to its neighborhood, for all i.

Note that consistency can be defined for the individual labels $\lambda \in \Lambda_i$ in a similar manner.

A consistent labeling will be denoted as $\{\Lambda_i^C\}$. Given a world model, and an initial labeling, our goal is to find the maximum consistent labeling contained therein. That is, *for all* i, $i = 1,...,n$ find the largest set Λ_i^C, *such that* $\Lambda_i^C \subseteq \Lambda_i^0$ and the labeling $\{\Lambda_i^C\}$ is consistent. If $\Lambda_i^C = \phi$ for some i, then the initial labeling is inconsistent with respect to the world model. In the case of the Huffman-Clowes model, this would mean that the line drawing represents an impossible blocks world scene. Note that a maximum consistent labeling may still be ambiguous. However, this is the best we can hope to do given the information contained in the world model. In the latter case, another means, such as a sequential search procedure, would have to be used to

completely disambiguate the labeling.

Definitions 2.3: Let $p_i(\lambda)$ be the *indicator function* for the set Λ_i, that is,

$$p_i(\lambda) = \begin{cases} 1 & \text{if } \lambda \in \Lambda_i \\ 0 & \textit{otherwise} \end{cases} \tag{2.1}$$

Similarly, let r_{ij} be the *indicator function* of the constraint relation R_{ij}:

$$r_{ij}(\lambda,\lambda') = \begin{cases} 1 & \text{if } (\lambda,\lambda') \in R_{ij} \\ 0 & \textit{otherwise} \end{cases} \tag{2.2}$$

We can use the definitions for the membership functions to express the Waltz filtering algorithm as a set of parallel, iterative equations, which comprise of the *discrete relaxation operator* [RHZ76] by:

$$p_i^{t+1}(\lambda) = p_i^t(\lambda) \wedge \bigwedge_{j \in N(i)} [\bigvee_{\lambda' \in \Lambda} [p_j^t(\lambda') \wedge r_{ij}(\lambda,\lambda')]], \quad \lambda \in \Lambda, \ i=1,...,n. \tag{2.3}$$

The discrete relaxation is a process which acts upon the initial labeling of a constraint network in order to solve the underlying labeling problem. Here, $\wedge$ and $\vee$ are used to denote logical conjunction and disjunction, respectively, and $N(i)$ denotes the set of vertices adjacent to v_i. In this algorithm, a label λ will be in the label set for a vertex v_i at time $t + 1$, $\lambda \in \Lambda_i^{t+1}$, if it is in the label set for that vertex at time t and if for every vertex v_j in the neighborhood of v_i there exists a label λ' such that the tuble (λ,λ') is in the constraint relation R_{ij} for edge v_iv_j. Thus any label which is not consistent with its neighborhood at a given time step will be removed at the next iteration of the process. The network is guaranteed to converge to a fixed point in at most $|\Lambda| \times |V| - 1$ time steps, since labels can only be removed from the label sets of their respective vertices.

Example 2.2: Consider a network the coarse topology shown in figure 2.4. Four vertices, v_1, v_2, v_3, and v_4 are connected by edges v_1v_2, v_2v_3, and v_3v_4. Constraint relations R_{12}, R_{23}, and R_{34} are shown associated with these edges.

Assume:

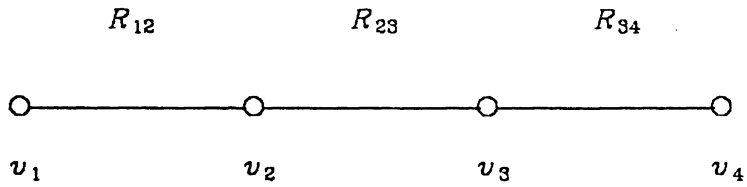

Figure 2.4: topology of constraint network for example .

(1) $\Lambda = \{a, b, c\}$ is the label set.

(2) The initial labeling $\{\Lambda_i^0\}$ is given as:

$\Lambda_1^0 = \{ a, c\}$,

$\Lambda_2^0 = \{ a, c\}$,

$\Lambda_3^0 = \{ a, b, c\}$,

$\Lambda_4^0 = \{ b, c\}$.

(3) The constraint relations are given as:

$R_{12} = \{(a,a), (b,b), (c,c)\}$,

$R_{23} = \{(a,c), (b,b), (c,a)\}$,

$R_{34} = \{(a,a), (b,b), (b,c), (c,c)\}$,

The associated fine graph shown in figure 2.5 is used to illustrate the behavior of this network in four successive time steps. The darkened circles denote those labels which are in the label set for the corresponding vertices, that is, the circle for vertex (λ,v_i) has been filled in if and only if $\lambda \in \Lambda_i$, or equivalently, if $p_i(\lambda) = 1$. Note that at time t=3 the network has converged to a fixed point which gives a unique

label to each vertex.

2.1.3. Input-Output Aspects of Constraint Networks

The previous section illustrated some the structural and behavioral aspects of constraint networks and their application to the graph labeling problem. In this section some of the functional aspects will be considered. In particular, we will be interested in viewing constraint networks as a mapping from one set to another independent of how that mapping is computed.

A (possibly empty) subset, ρ, of the n^{th} order Cartesian product of a set Λ:

$$\rho \subseteq \Lambda \times \Lambda \times \cdots \times \Lambda \text{ (n times)},$$

is called an *n-ary relation* defined on Λ. Let P_n denote the set of all n-ary relations defined on Λ^n (note[3]). An initial labeling $\{\Lambda_1^0, \Lambda_2^0, \cdots, \Lambda_n^0\}$ can be represented by the relation

$$\rho^0 = \Lambda_1^0 \times \Lambda_2^0 \times \cdots \times \Lambda_n^0$$

the same holding for a consistent labeling

$$\rho^C = \Lambda_1^C \times \Lambda_2^C \times \cdots \times \Lambda_n^C$$

so that the input-output description of a constraint network can be given by a function which maps from n-ary relations to n-ary relations. The two issues that we will discuss are (1) to describe the function computed by a given constraint network, and (2) the design of a constraint network to realize a given function. Both can be addressed independently of the behavior of the network.

We start with a more basic view towards a constraint network $C_n(\Lambda)$, which is as an acceptor of strings of symbols of length *n*.

Definition 2.4: Let $\Sigma_n(\Lambda)$ be the set of all strings of length *n* defined on the symbol set Λ and let $\sigma_n \in \Sigma_n(\Lambda)$ be a string of length *n*

---

[3] There is an unfortunate conflict in notation here. In this case we intend Λ^n to denote the n^{th} order Cartesian product of the set Λ, instead of the state of a label set at time $t=n$. It is hoped that in general the meaning will be clear from the context.

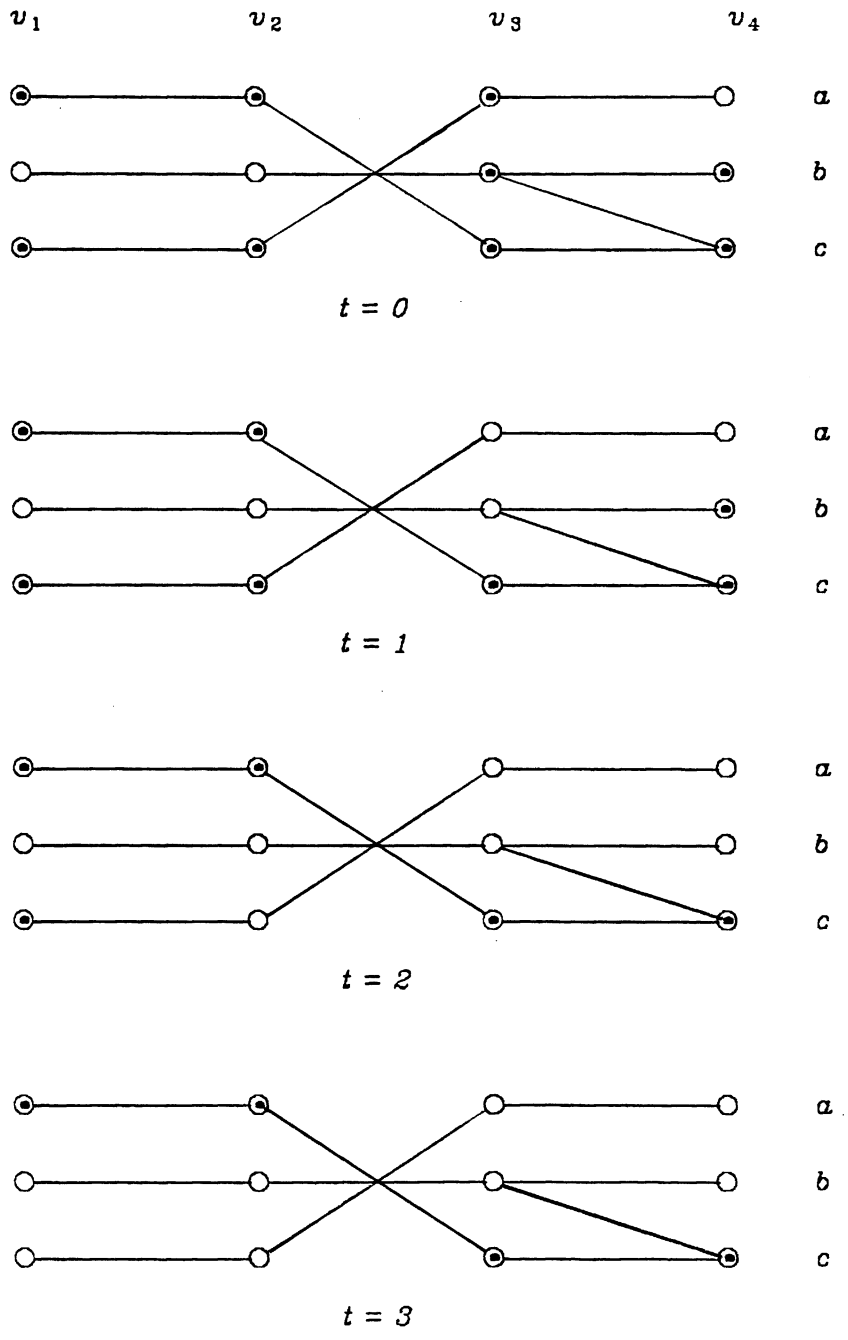

Figure 2.5: Behavior of constraint network at 4 sequential time intervals.

$$\sigma_n = \lambda_1 \lambda_2 \cdots \lambda_n.$$

Then a constraint network $C_n(\Lambda)$ *accepts* σ_n if and only if

$$\Lambda_1 = \{\lambda_1\}, \Lambda_2 = \{\lambda_2\}, \ldots, \Lambda_n = \{\lambda_n\}$$

is a consistent labeling for C_n.

Given a string σ_n it is not difficult to specify a necessary and sufficient condition that a given constraint network will accept it. To do so, we will need the following definition:

Definition 2.5: Denote by $[n]$ the set of integers $\{1, 2, \ldots, n\}$. Let I be an index set such that

$$I = \{i_1, i_2, \ldots, i_m\} \subseteq [n].$$

with $|I| = m \leq n$. Then the *string projection function* is a mapping

$$\pi_I : \Sigma_n \rightarrow \Sigma_m$$

defined by:

$$\pi_I(\lambda_1 \lambda_2 \cdots \lambda_n) = \lambda_{i_1} \lambda_{i_2} \cdots \lambda_{i_m}.$$

Claim 2.1: A constraint network $C_n(\Lambda) = (V,E)$ accepts a string σ_n if and only if

$$\pi_{\{i,j\}} \in R_{ij}, \text{ for } all\ v_i v_j \in E_H.$$

The claim is obvious and will be left without proof.

The set of all strings accepted by a constraint network C_n is a relation ρ which is of central importance because, in pattern recognition applications, it amounts to an assertion about the set of all possible events. The definitions given above of acceptance and projection for strings can easily be extended to relations as follows:

Definition 2.6: A constraint network C_n accepts a relation ρ if it accepts every element of ρ.

Definition 2.7: Let I be an index set with $I \subseteq [n]$ and $|I| = m$, the *relation projection function* $\bar{\pi}_I$ is a mapping,

$$\overline{\pi}_I : P_n \rightarrow P_m$$

defined by

$$\overline{\pi}_I(\rho_n) = \bigcup_{\sigma_n \in \rho_n} \pi_I(\sigma_n).$$

Claim 2.2: A constraint network C_n accepts a relation ρ_n if and only if $\overline{\pi}_{\{I,J\}}(\rho_n) \subseteq R_{IJ}$ for all $v_I v_J \in E_H$.

Assume that a particular application specifies a graph $C_H = (V_H, E_H)$ and a relation (set of events) ρ_n. A constraint network is to be constructed which accepts ρ_n. By claim 2.2 it would seem reasonable to set $R_{IJ} = \overline{\pi}_{\{I,J\}}(\rho)$ for all $v_I v_J \in E_H$.

Definition 2.8: Let $C_n(\Lambda, C_H)$ be the set of all constraint networks defined on the label set Λ and with coarse graph $C_H = (V_H, E_H)$. The *network projection* function Π_{C_H} is a mapping,

$$\Pi_{C_H} : P_n \rightarrow C_n(\Lambda, C_H)$$

defined by

$$\Pi_{C_H}(\rho_n) = C_n(\Lambda), \textit{ where } R_{IJ} = \overline{\pi}_{\{I,J\}}(\rho), \textit{ for all } v_I v_J \in E_H.$$

Unfortunately, the constraint network $C_n(\Lambda) = \Pi_{C_H}(\rho_n)$ is likely to accept strings other than those contained in ρ_n. Thus, given a coarse graph C_H, there will be some relations which will not be *accepted uniquely* (that is, those strings and only those strings in the given relation will be accepted) by a constraint network with the given topology. This fact may be of some importance in the research proposed below.

Example 2.3: Let $\rho_3 = \{aab, baa\}$ and $C_H = (V_H, E_H)$ with $V_H = \{v_1, v_2, v_3\}$ and $E_H = \{v_1 v_2, v_2 v_3\}$. The coarse graph G is shown in figure 2.6.[4] Projecting ρ_3 onto the two edges in the graph C_H results in a constraint network with $R_{12} = \{aa, ba\}$, and $R_{23} = \{aa, ab\}$, as shown in figure 2.7. The set of strings accepted by this network is, however,

[4] Note that the coarse topology of most of the constraint networks shown as examples in this document are relatively simple, for the most part being linear. This need not be the case in general.

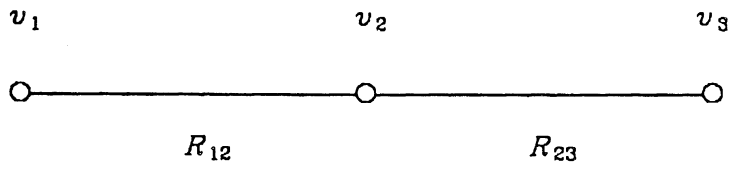

Figure 2.6: coarse graph for constraint network of example 2.3.

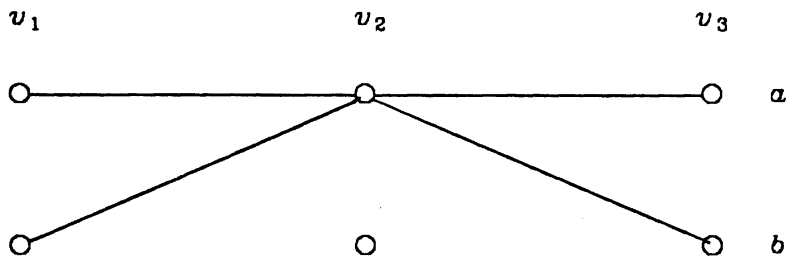

Figure 2.7: fine graph for constraint network of example 2.3.

$$\rho' = \{aaa, aab, baa, bab\}.$$

Note that in the examples given above, the tuples in the constraint relations are represented by strings of length 2. From this standpoint a constraint network can be seen as accepting all strings whose substrings are represented in every constraint relation associated with each edge. This being both a necessary and sufficient condition for acceptance we can characterize those strings accepted by a given constraint network using the formalism described below:

Definition 2.9: Let $\sigma_m \in \Sigma_m$ be a string defined on the symbol set Λ and $I \in [n]$ be an index set. The *string embedding function* ψ_I is a mapping,

$$\psi_I : \Sigma_m \rightarrow P_n$$

defined by

$$\psi_I(\sigma_m) = \{\sigma_n \mid \pi_I(\sigma_n) = \sigma_m\}$$

The string embedding function maps a string $\sigma_m \in \Sigma_m$ to the relation $\rho_n \in P_n$ which is the preimage of σ_m under the mapping π_I.

Definition 2.10: Let $I \subseteq [n]$ be an index set. The *relation embedding function* $\overline{\psi}_I$ is a mapping,

$$\overline{\psi}_I : P_m \rightarrow P_n$$

defined by

$$\overline{\psi}_I(\rho_m) = \bigcup_{\sigma_n \in \rho_m} \psi_I(\sigma_n)$$

Definition 2.11: Let $C_n(\Lambda) = (V,E)$ be a constraint network and $C_H = (V_H,E_H)$ the associated coarse graph. The *network reconstruction function* Ψ is a mapping,

$$\Psi_{C_H} : C_n(\Lambda,C_H) \rightarrow P_n$$

defined by

$$\Psi_{C_H}(\rho_m) = \bigcap_{v_i v_j \in E_H} \overline{\psi}(R_{ij}).$$

The reconstruction function, applied to a given constraint network is the the relation of all strings accepted by that network. Clearly, for any graph C_H, if $\rho' = \Psi(\Pi_{C_H}(\rho))$ then $\rho \subseteq \rho'$.

Definition 2.12: A relation ρ^* for which $\Psi(\Pi_{C_H}(\rho^*)) = \rho^*$ is called *closed* with respect to C_H. Furthermore, the network $\Pi_{C_H}(\rho^*)$ is said to be *sufficient* for ρ^*.

The set of closed relations is important because, given a coarse topology C_H, it specifies those relations for which a constraint network which uniquely accepts that relation can be constructed. Note the dependency on C_H. In fact, this dependency is

only poorly understood and is suggested as a research topic.

Define $F_{C_H} = \Psi_{C_H} \bigcirc \Pi_{C_H}$, to be the composition of the network reconstruction function and the network projection function. Furthermore, let the set of relations $P_n(\Lambda)$, partially ordered by set inclusion be the *lattice* $\langle P_n(\Lambda), \subseteq \rangle$. Then the set $P^C_{C_H} \subseteq P_n(\Lambda)$ of relations which are closed with respect to F_{C_H} is a *closure system* [DDT78] and F_{C_H} is a *closure operator* since,

(1) if $\rho_n^1 \subseteq \rho_n^2$ then $F_{C_H}(\rho_n^1) \subseteq F_{C_H}(\rho_n^2)$

(2) $\rho_n \subseteq F_{C_H}(\rho_n)$

(3) $F_{C_H}(F_{C_H}(\rho_n)) = F_{C_H}(\rho_n)$

Claim 2.3 [Mon74]: if $\rho_n^1, \rho_n^2 \in P_n^C(\Lambda)$, then $\rho_n^1 \cap \rho_n^2 \in P_n^C(\Lambda)$.

Thus the set of relations closed with respect to a given coarse topology is a sup semilattice.

The issue of characterizing those strings which are closed with respect to a constraint network with coarse graph C_H was addressed above. The dual problem is to characterize the minimal coarse graphs (in terms of the number of edges) with respect to which a given relation is closed. Given *any* relation ρ and *any* coarse graph C_H a constraint network with the specified topology can be constructed to accept ρ_n as $C_n(\Lambda) = \Pi_{C_H}(\rho_n)$. Although, as noted above, $C_n(\Lambda)$ may not accept ρ_n uniquely. Note that adding an edge to the edge set E_H of C_H increases the "discriminating power" of the resulting graph C'_H. That is, in general a greater number of relations will be closed with respect to C'_H. Therefore, if we wanted to guarantee that a network will accept a given relation ρ_n uniquely, the obvious solution would be to specify that the coarse graph be the complete graph with n vertices, K_n. Unfortunately, this condition is not sufficient.

Example 2.4: Let C_H be the complete graph with three vertices, K_3, as shown in figure 2.8. Let the relation ρ_3 be given by:

$$\rho_3 = \{abb, bab, bba\}$$

Then $C_n(\Lambda) = \Pi_{C_H}(\rho_3)$ with associated fine graph shown in figure 2.9 accepts the relation

$$\rho'_3 = \{bbb, abb, bab, bba\} \neq \rho_3$$

One way to increase the discriminating power of a constraint network is to increase the *order* of the edges in the coarse graph C_H.

Definition 2.13: An hypergraph is a tuple $G = (V,E)$ such that

(1) $V = \{v_1, v_2, \ldots, v_n\}$ is a set of edges, and

(2) $E = \{e_1, e_2, \ldots, e_s\}$ is a set of hyperedges, where each hyperedge e_i is a subset (of arbitrary order) of V, $e_i \subseteq V$.

Definitions 2.2 through 2.12 above extend directly to the case where the associated coarse graph C_H is , more generally, an hypergraph. The order of the

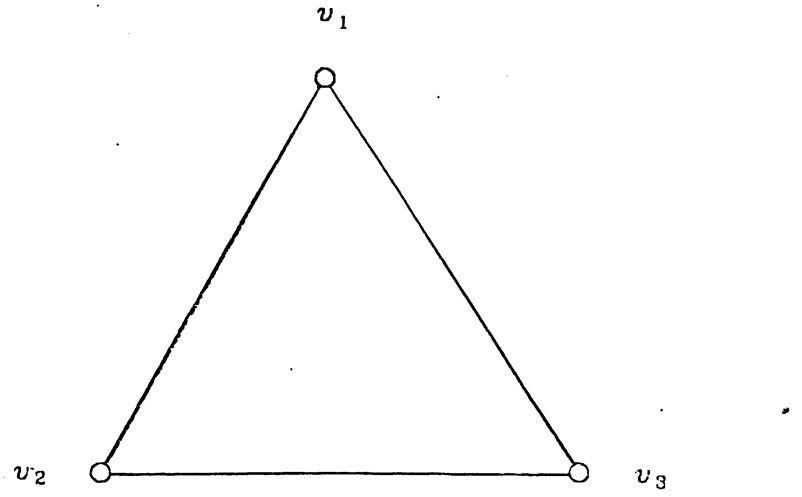

Figure 2.8: coarse graph for example 2.4.

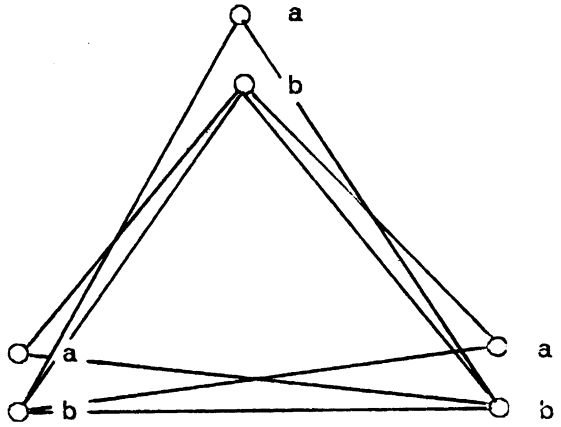

Figure 2.9: fine graph for example 2.4.

largest edge in C_H is considered to be the order of the associated constraint network. Clearly, any relation ρ_n can be uniquely accepted by a constraint (hyper) network of order n. The smallest order α of a constraint network which uniquely accepts a given relation ρ_n is important because it can be used as a measure of the intrinsic globality or complexity of the relation. Determining α for an arbitrary relation is strongly suspected to be n-p complete, and no reasonable polynomial time bounds have been published. If we associate a processor with each hyperedge in the network then the labeling on the set of vertices in an hyperedge forms the *context* of the associated processor. The context is all the information that is directly observable for making a labeling decision. All other information must be propagated through another hyperedge.[5]

We have at this point covered some of the fundamental issues relevant to discrete relaxation. The model is evidently very powerful and has only begun to be

[5] If we consider the Huffman-Clowes line labeling scheme to be applied to the relation consisting of all legal blocks world scenes, then the context defined by the line drawing is not sufficient to uniquely accept this relation. That is, there are line drawings for which no real blocks world scene exists, that will nonetheless, be Huffman-Clowes labelable.

understood. We suggest that one could easily dedicate an entire thesis to stating the research issues which arise in conjunction with this topic.

2.2. Continuous relaxation labeling

Given an n-ary relation, the network projection function described in the previous section creates a constraint network which implements a necessary condition, and in some situations a necessary and sufficient condition, for the recognition of the strings in that relation, given an initial labeling. The process can be implemented in parallel hardware and applied to real world problems. In the discrete case it is assumed that the feature detector responsible for the initial labeling can detect, with certainty, the presence or absence of a feature. It is often more realistic to assume that the feature detector is limited to assigning some measure of strength or figure of merit, to each feature for each vertex of the graph. This would be the case if, for example, the actual labeling was observed in the presence of noise. To this end a continuous analog of the constraint network formalism has been proposed [RHZ76] and has subsequently undergone several stages of development.

2.2.1. Extension to the Continuous Labeling Problem

The initial extension of discrete relaxation labeling to the continuous domain was made by giving the label sets Λ_i, and the constraint relations R_{ij} fuzzy set interpretations [DDT78]. This is done by letting the membership functions for the label sets and the constraint relations take on values in the range [0,1], that is, $p_i(\lambda) \in [0,1]$ and $r_{ij}(\lambda,\lambda') \in [0,1]$ (refer to equations 2.1 and 2.2 above). The *iterative updating equations* for the fuzzy set extension is equivalent to those for the discrete case (eqn. 2.3):

$$p_i^{t+1}(\lambda) = p_i^t(\lambda) \wedge \bigwedge_{j \in N(i)} [\bigvee_{\lambda' \in \Lambda} [p_j^t(\lambda') \wedge r_{ij}(\lambda,\lambda')]] \qquad (2.4)$$

except that $\Lambda\{\ \}$ and $V\{\ \}$ are now interpreted as the infimum and supremum, respectively, of the set $\{\ \}$. These iterative equations must converge to a fixed point

in no more than $(|\Lambda|\times|V|)\times(|\Lambda|\times|V|-1)/2$ time steps since this is the number of discrete differences that exist in the network.

The $p_i(\lambda)$ are now to be interpreted as strength measure or figures of merit for the label λ on vertex v_i. "Likelihood estimate" and "probability estimate" are terms which appear quite often in the literature, although their use has never been justified in any formal sense. The $r_{ij}(\lambda,\lambda')$ are often referred to as *compatibility coefficients* and are intended to represent some measure of compatibility or mutual reinforcement between the label λ on vertex v_i and λ' on vertex v_j.

At the same time the extension to fuzzy sets was made an *algebraic updating rule* was proposed [RHZ76]. The basis of this rule is to calculate the strength measure $p_i^{t+1}(\lambda)$ as a function of the "support" $s_{ij}(\lambda)$ for the label λ on node v_i due to the labeling, Λ_j on node v_j. The (linear) support is defined to be the average of the label strengths on the adjacent node weighted by compatibility coefficients $r_{ij}(\lambda,\lambda')$:

$$s_{ij}^t(\lambda) = \sum_{\lambda'\in\Lambda} r_{ij}(\lambda,\lambda')\, p_j^t(\lambda), \tag{2.5}$$

and the iterative updating equations for the algebraic updating rule are given in terms of the linear support by:

$$q_i^{t+1}(\lambda) = p_i^t(\lambda)[1 + \sum_{j\in N(i)} s_{ij}^t(\lambda)], \tag{2.6}$$

and,

$$p_i^{t+1}(\lambda) = \frac{q_i^{t+1}(\lambda)}{\sum_{\lambda'\in\Lambda} q_i^{t+1}(\lambda')}. \tag{2.7}$$

Equation 2.6 is an ad hoc procedure for combining the supports to produce a new labeling estimate, q_i^{t+1}. It is based on the heuristic that the more support the labeling on a neighborhood gives to a particular label, the more likely it is to be the correct label. Equation 2.7 serves to normalize these estimates so that they sum to 1 for a given node:

$$\sum_{\lambda \in \Lambda} p_i(\lambda) = 1, \quad \textit{for all } i, \tag{2.8}$$

as would be required if the strength measures were to be interpreted as probability estimates. The hope is, that the process driven by such equations will converge to a fixed point where the label with the highest measure of support is a "good" choice for the unique label to be assigned to that node. In most applications of algebraic updating rules, the compatibility coefficients are assumed to take on values in the range [-1,1].

In order to simplify further discussion we introduce the following notation: Denote by $\bar{p}_i$, the vector of labeling values at a particular vertex:

$$\bar{p}_i = (p_i(\lambda_1), p_i(\lambda_2), \ldots, p_i(\lambda_m)), \quad \bar{p}_i \in R^m$$

Furthermore, define the *labeling vector*, $\bar{p}$ by:

$$\bar{p} = (\bar{p}_1, \bar{p}_2, \ldots, \bar{p}_n), \quad \bar{p} \in R^{nm}.$$

Thus, $\bar{p}$ contains all the information about the current labeling. If we require equation 2.8 to hold, then $\bar{p}$ must be an element of the set K of *weighted labeling assignments* given by:

$$K = \{\bar{p} \in R^{nm} \mid \sum_{\lambda \in \Lambda} p_i(\lambda) = 1, \quad \textit{for all } i\}.$$

Furthermore, we can express the set of *unambiguous labelings* by

$$K^* = \{\bar{p} \in R^{nm} \mid \bar{p} \in K \text{ and } p_i(\lambda) \in \{0,1\}, \textit{ for all } i,\lambda\}.$$

As in the discrete case, the labeling is unambiguous if for each i, $i = 1,\ldots,n$, $p_i(\lambda) = 1$ for exactly one $\lambda \in \Lambda$, so that $p_i(\lambda') = 0$, $\lambda' \neq \lambda$. It is easy to show that K is the convex hull of K^* [HuZ80]. Finally, the set of compatibility coefficients can be extended to allow a "compatibility" measure between every vertex in the graph by setting $r_{ij}(\lambda,\lambda') = 0$ if v_i is not adjacent to v_j. The resulting set can be expressed in terms of an $nm \times nm$ compatibility matrix R. Note that given this notation, equation 2.5 can be expressed very easily as

$$\bar{s}^t = R\,\bar{p}^t \tag{2.9}$$

where $\bar{s}$ is defined in a manner similar to $\bar{p}$.

Recently, two additional algebraic updating formulas have been proposed: Let the support $s_i^t(\lambda)$ be given, as above, by equation 2.5. In the first approach, due to Peleg [Pel80a], the updated strength measures are expressed as:

$$q_i^{t+1}(\lambda) = p_i^t(\lambda) \sum_{j \in N(i)} s_i^t(\lambda). \tag{2.9}$$

In the second, known as the product updating rule [Kir80], they are given as:

$$q_i^{t+1}(\lambda) = p_i^t(\lambda) \prod_{j \in N(i)} s_i^t(\lambda), \tag{2.10}$$

In both cases the updated strength measures are normalized as in equation 2.7 above. There was in fact some attempt to justify equations 2.9 and 2.10 on formal grounds. Both are based, at least in part, on the assumption [Pel80] that

$$Pr(\bar{p}_i^t, \bar{p}_j^t \mid \lambda, \lambda') = Pr(\bar{p}_i^t \mid \lambda) \cdot Pr(\bar{p}_j^t \mid \lambda'), \tag{2.11}$$

where Pr is used to denote the probability density function, and $\bar{p}_i^t$ denotes the current labeling (i.e. set of strength measures) considered as an event. It can be shown, however, that equation 2.11 implies,

$$Pr(\lambda \mid \bar{p}_i^t, \bar{p}_j^t) = Pr(\lambda \mid \bar{p}_i^t). \tag{2.12}$$

In other words, the probability that the actual label on node v_i is λ is independent of the labeling $\bar{p}_j^t$ on the adjacent node v_j. This, of course, is in contradiction to the heuristic arguments which justify the incorporation of the support due to the labeling on the neighborhood.

Once the updating formula has been specified, the function computed by the relaxation labeling algorithm is completely determined by the matrix of compatibility coefficients. The specification of the compatibility matrix becomes, therefore, an issue of primary importance. But, since there is no formal theory on which the derivation of either the updating rules or the compatibility coefficients can be based, one is left to heuristics or ad hoc rules for extracting the compatibility matrix from training samples.

The use of heuristics is domain dependent, so that no general principles can be stated. We note, however, that (consistent with the purpose that they are presumed

to serve) we would expect that the compatibility coefficients would take values approaching 1 if it is desirable to have the labels they represent on adjacent nodes, and values values approaching -1 (or 0, depending on the particulars of the updating formula) if the associated labels should never co-occur on adjacent nodes. For an early example of a set of compatibility coefficients for a continuous relaxation labeling process based on heuristics, see [ZHR77].

Various methods for deriving compatibility coefficients from training samples have been proposed. The most common approach is to use the sample correlation coefficient [PeR78,Pel80][6]. Another approach is to use a statistic derived from the sample (Shannon) mutual information between given labels [PeR78]. And various other schemes have been proposed [Yam79].

2.2.2. Structural and Behavioral Aspects of Continuous Relaxation Processes

Unlike the discrete relaxation processes, neither the behavior nor the fixed points of the algebraic updating rules are well understood. With the exception of a few concepts which will be discussed below, the survey of the previous section covers most of the current theory. The dangers inherent in designing an iterative process of many variables in an ad hoc manner are apparent from some of the results that have been reported. Perhaps the most consistent observation has been that the continuous relaxation processes will produce the best results after three or four iterations, after which the performance degrades as noise points become dominant and are propagated [ZHR77]. Furthermore, in some cases the process failed to converge, showing a wandering or behavior instead. This lead to some research into stopping criteria [RLS80,Pel80b], an area which was active for a short period of time.

Note that the algebraic updating rules have been derived, more or less, as an extension of the discrete relaxation operator of equation 2.7. Central to the

[6] In fact, the use of correlation coefficients can be "derived" or justified, in an informal sense, for the updating rules of equations 2.9 and 2.10 if one were to accept the assumptions expressed by equation 2.11.

concept of discrete relaxation is the idea of consistency as given in definition 2.2. A equivalent idea can be expressed for weighted labeling assignments as follows [HuZ80]:

Definition 2.14: Let $\bar{p} \in K$ be a weighted labeling assignment, and let the support $s_i(\lambda)$ be given as in equations 2.5, 2.10, and 2.9 above:

$$s_i(\lambda) = \sum_{j \in N(i)} \sum_{\lambda' \in \Lambda} r_{ij}(\lambda,\lambda')\, p_j(\lambda')$$

Then $\bar{p}$ is *consistent* in K if

$$\bar{p}\, \bar{s}^T \geq \bar{v}\, \bar{s}^T \tag{2.13}$$

for all $\bar{v} \in K$.

Since the fixed points of the discrete relaxation operator defined on a constraint network are the consistent labelings, it would seem reasonable to use consistency (in the continuous sense) to *define* the fixed points of a continuous relaxation operator and then examine those operators and circumstances in which these fixed points are achieved [ZLM81].

This does not, of course, explain the meaning of the fixed points, and in particular, their dependency upon the compatibility matrix, in terms of the original problem. That is, there is no clear way to relate the concept of consistency in the continuous case with the problem that is being solved. This is one of the major draw backs with this approach.

2.2.3. Other Approaches to the Continuous Labeling Problem

Other approaches to the continuous labeling problem have begun to appear in order to circumvent some the problems with the algebraic updating rules described above. Several workers have suggested increasing the information inherent in the compatibility coefficients by making them a function of the labeling on several adjacent vertices [HuZ80] and some results on experiments have been reported

[EkR80,FER81]. For example, a coefficient such as $r_{i;jk}(\lambda;\lambda_1,\lambda_2)$ would be a measure of the compatibility of label λ on vertex v_i with the label λ_1 on vertex v_j *and* the label λ_2 on the vertex v_k. However, this runs into problems due to the memory requirements for the large compatibility matrix and extra computational requirements, and generally is as unfounded as the approaches suggested above.

Another means which has been explored for controlling the convergence and fixed points of the algebraic updating rules is to enhance the control structure of the process. Zucker [Zuc78] has reported on several experiments in which multiple levels or copies of a relaxation network are run in parallel, with information flow between levels for the purpose of controlling the performance of the process at the lower level. Perhaps the most intriguing scheme for enhancing control is found in the *augmented relaxation labeling* and *dynamic relaxation labeling* processes proposed by Kuschel and Page [KuP81,Kus80]. The essence of this approach is to incorporate a global broadcasting mechanism by which the labeling at one vertex is used to bias the compatibility matrix associated with another vertex, possibly at some distance in the network. The broadcasting/biasing mechanism is model driven, thus adding an hierarchical component to the process. By biasing the compatibility coefficients, the network becomes sensitive to patterns whos existence can be predicted from models and the global state of the network. Since in most current approaches the networks are relatively insensitive to specific patterns and since the ability of relaxation networks to "propagate" information beyond a few vertices before that information becomes corrupted is open to question [Kus81] it is felt here that this scheme is worthy of further investigation. We point out that a dynamic programming model would probably be useful in this context.

Another class of approaches is based on optimizing a function of the strength measures $p_i(\lambda)$. Although solution algorithms thus derived are well behaved, it leaves several issues still to be addressed: In the first place, the function to be optimized for a given application may difficult, if not impossible to specify. If on the other hand,

a function is determined on some ad hoc basis, it is usually difficult to relate the meaning of the fixed points of the process to the original application.

In order to illustrate this latter case, consider the optimization approach of Faugeras and Berthod [FaB81]: If the compatibility coefficients are viewed as being conditional probabilities;

$$r_{ij}(\lambda,\lambda') \equiv p_{ij}(\lambda \mid \lambda')$$

and the strength measures $p(\lambda)$ as probabilities, then we may estimate the probability of the label λ on vertex v_i from the labeling on vertex v_j by:

$$q_{ij}^t(\lambda) = \sum_{\lambda' \in \Lambda} r_{ij}(\lambda,\lambda')\, p_j^t(\lambda') \tag{2.14}$$

the estimates due to the labeling on all adjacent labels may be averaged to yield an estimate due to the labeling on the neighborhood as:

$$q_i^t(\lambda) = \frac{1}{|N(i)|} \sum_{j \in N(i)} q_{ij}^t(\lambda) = \frac{1}{|N(i)|} \sum_{j \in N(i)} \sum_{\lambda' \in \Lambda} r_{ij}(\lambda,\lambda')\, p_j^t(\lambda'). \tag{2.15}$$

Let $\bar{q}_i$ be the vector of label strength estimates for a the vertex v_i:

$$\bar{q}_i = (q_i(\lambda_1), q_i(\lambda_2), \ldots, q_i(\lambda_m)).$$

Then we can define another measure of consistency as being the difference between the current labeling and the labeling predicted by the neighborhood:

$$C_1 = \| \bar{q}_i^t - \bar{p}_i^t \| \tag{2.15}$$

We can, furthermore, specify a measure of ambiguity of the labeling at a vertex by:

$$C_2 = \sum_{\lambda \in \Lambda} p_i^t(\lambda) \log p_i^t(\lambda). \tag{2.16}$$

and then minimize the function given as the linear combination:

$$f(\bar{p}) = \alpha C_1 + (1 - \alpha) C_2. \tag{2.17}$$

where α is an arbitrarily specified number in the range [0,1]. Note that as α

approaches 0 in equation 2.17 the optimization procedure will tend to ignore the contextual information as expressed in the compatibility coefficients (refer to equation 2.12 above). On the other hand as α approaches 1, there will be no explicit tendency to disambiguate the labeling. Subjective evaluation of results with this technique have favored values for α approaching 1. Note, however, there is no way to relate these results to the original problem. In fact, the nature of the original problem has never been analyzed.

2.2.4. Applications

A survey of recent applications for continuous relaxation labeling techniques would require an unreasonable amount of space. We make note of a few applications here and refer to surveys by Rosenfeld [Ros79,Ros81] and Davis and Rosenfeld [DaR80] for a more complete review. Results relating to this topic appear regularly in the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, the *IEEE Transactions on Systems, Man and Cybernetics*, *Pattern Recognition* (Pergamon, London), and *Computer Graphics and Image Processing* (Academic, New York). Papers on relaxation labeling can be found at conferences relating to areas in computer vision and pattern recognition: the *International Conference on Pattern Recognition*, the *IEEE Conference on Pattern Recognition and Image Processing*, the *DARPA image understanding workshop*, and the *International Joint Conference on Artificial Intelligence*, to name a few. Finally, the topic is addressed in several recent books [Pav77,HaR78].

One of the earliest applications of relaxation labeling was to curve and line enhancement [ZHR77]. We describe this application as an example below. Marr et al. [MaP76,MPP77] employs relaxation in the correspondence problem for stereo images. Ullman [Ull79a,b] applies relaxation techniques to time varying imagery. Other applications include multispectral pixel classifications [EYR80], template matching [DaR77], segmentation [HaR78], and shape matching [Dav79]. For areas other than

computer vision, applications include handwriting recognition [Hay79], the solution to substitution ciphers [PeR79], and matrix reconstruction [KrN81].

Example 2.5: the application continuous relaxation labeling to curve and line enhancement:

A computer vision system typically involves a T.V. camera, linked to a computer, which is observing some scene of interest. We assume that the necessary processing can be carried out to store a digitized representation of that scene in the computer memory. This representation is expressed as an $n \times n$ (512×512 is a typical value) matrix in which each element is an intensity value sampled from the original image. One of the goals of low level in computer vision is to extract a *line drawing* or *picture graph* from the scene, since the closed regions in this description often represent objects, or object surfaces - which are generally considered to be scene primitives. The problem of extracting a line drawing is usually approached by first detecting edges - reing the edges into straight lines, or smooth contours. Althoughcan be performed sequentially, the relaxation techniques are used to enhance or smooth out the edges in a parallel, iterative manner over the entire image.

One approach to edge detection is to correlate *edge masks*, with the intensity values in a neighborhood of a given pixel. The output of the correlation operation is the response of an edge detector which is sensitive to edges at a specific orientation. Masks for edges at eight orientations are shown in figure 2.10. We can associate labels λ_0 through λ_7 with these edge masks, as assertions of edges at the given orientation through the pixel at the center of the neighborhood. The label λ_8 will correspond to an assertion that no edge exists in the given neighborhood.

A graph structure is imposed on the image raster by connecting every pixel to its eight immediate neighbors, as shown in figure 2.11. The specification for the label set and the graph given above defines a continuous graph labeling problem. The initial labeling values, $p_i^0(\lambda)$, are established by the outputs of the mask correlation

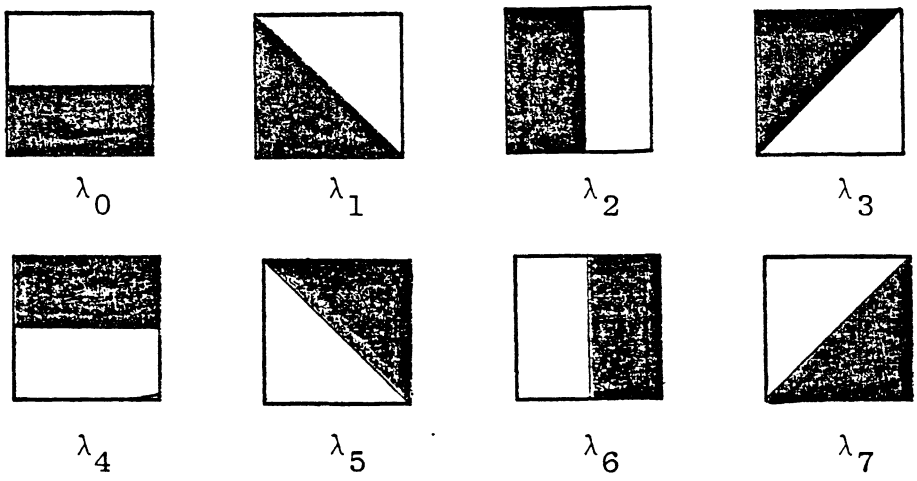

Figure 2.10: edge masks for the application of example 2.5.

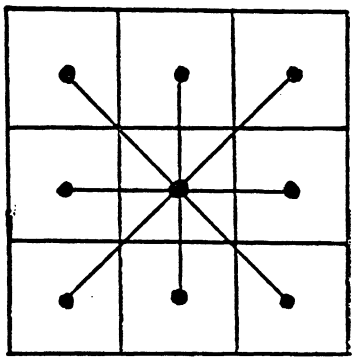

Figure 2.11: neighborhood of a single pixel in the graph defined for example 2.5.

set and the graph given above defines a continuous graph labeling problem. The initial labeling values, $p_i^0(\lambda)$, are established by the outputs of the mask correlation operations for each neighborhood of each pixel in the image.

The contextual information, as represented by the compatibility coefficients, is derived by considering the objectives of the relaxation process: Because we would like to enhance edges that lie on a straight or smoothly curving line it would make sense to give high compatibility values to neighboring edges that are, or are almost, collinear. For example, for two pixels in an horizontal configuration, as shown in figure 2.12a, we might assign the compatibility coefficients for the two pairs of collinear edges as:

$$r_{ij}(\lambda_0,\lambda_0) = 1, \quad r_{ij}(\lambda_4,\lambda_4) = 1.$$

Whereas for the anti-collinear edges of figure 2.12b we would assign the compatibility coefficients of:

$$r_{ij}(\lambda_0,\lambda_4) = -1, \quad r_{ij}(\lambda_4,\lambda_4) = -1,$$

And finally, for edges representing intermediate degrees of curvature we would assign various other compatibilities as shown in figure 2.12c.

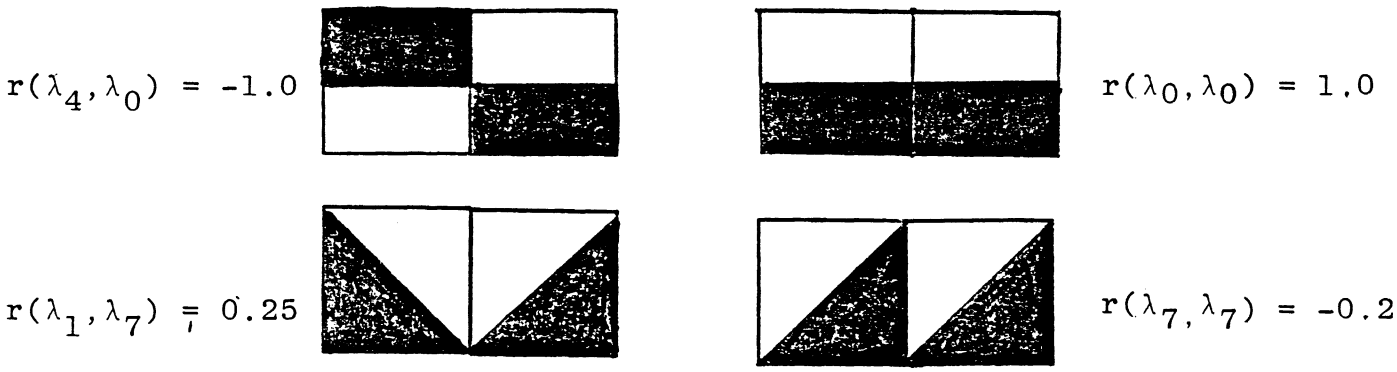

Figure 2.12: Compatibility coefficients for various configurations of pixels.

Then given an initial labeling, the iterative equations of 2.11 through 2.17 could be applied in attempt to remove noise points and straighten curved lines.

3. An Overview of the Problem Domain

The previous section discussed the theory of relaxation labeling both to orient the reader to current approaches and the general philosophy towards its use in the labeling problem, and to organize the notation and other material which we will expect to use in the research which we are proposing. By necessity the treatment was cursory. Our goal was to touch upon some of the major aspects of the topic leaving the references as a guide to the various extensions that exist in the literature. We now present an overview of the problem domain, with an emphasis on the continuous graph labeling problem. Our intent is to use the discussion of the previous section as a starting point for an analysis of the objectives and assumptions which seem to be inherent in the current literature. Clearly defined approaches to the labeling problem will then be discussed, with a view towards research into solution algorithms which can be justified in a more reasonable way than those which currently exist.

3.1. The Nature of the Problem

By labeling what is meant is the association of a unique label λ from a label set Λ to each component X_i of a random vector X. The assignment of a label λ to a component X_i is to be viewed here as being equivalent in every sense to *classifying* X_i into class λ. As such, the problem could be approached from the point of view of statistical decision theory and need not be addressed further here. It is the constraint on the solution algorithm imposed by the graph structure - implying the existence of a processor at each node to make labeling decision for the associated component, and edges representing data paths between processors - which makes this problem unique.

Even so, the problem needs to be refined further before it becomes meaningful: In the first place, as Ullman [Ull79] points out, if each node was represented by a universal processor, or if each processor was connected to every other processor in

the network, then *any* function could be computed by the network, and we would be left with the general labeling problem of the previous paragraph. Therefore, we will want to constrain the implementation to Ullman's [Ull79] *simple local processes* which are defined in the spirit of (informal) concept of cooperative processes by the following four criteria:[7]

(1) *Locality:* Let r, the "radius of computation," be the maximum degree of a node in a network. Then we would like r to be as "small" as possible.

(2) *Simplicity:* We would like the function computed at each processor to be relatively "simple". No attempt has been made to formally define simplicity, we add on our own however, that the processor at each node must have a limited amount of memory.

(3) *Uniformity:* We would like the topology of network to be as uniform as possible, and the procedure executed by each processor at each node to be the same.

(4) *Efficiency and Convergence:* The iterative process should have good convergence properties.

An example of a uniform network with a small radius of computation is an image raster where each pixel is considered to be adjacent to its eight neighbors ($r = 8$).

In the second place, one must specify the context within which the problem is defined: what the underlying processes are, what is being estimated and how, what prior information is available, and so forth. In fact, many different systems may include a labeling problem. The system shown in figure 3.1 which we will use as our model is motivated by considering examples of applications of relaxation techniques which appear in the literature. By comparison, the conventional model for symbol transmission through a channel is given in figure 3.2. The similarity suggests that some of the theory and technique from coding and information theory may be applied to our problem. It also suggests that relaxation labeling techniques may find

[7] In fact Ullman specifies six criteria.

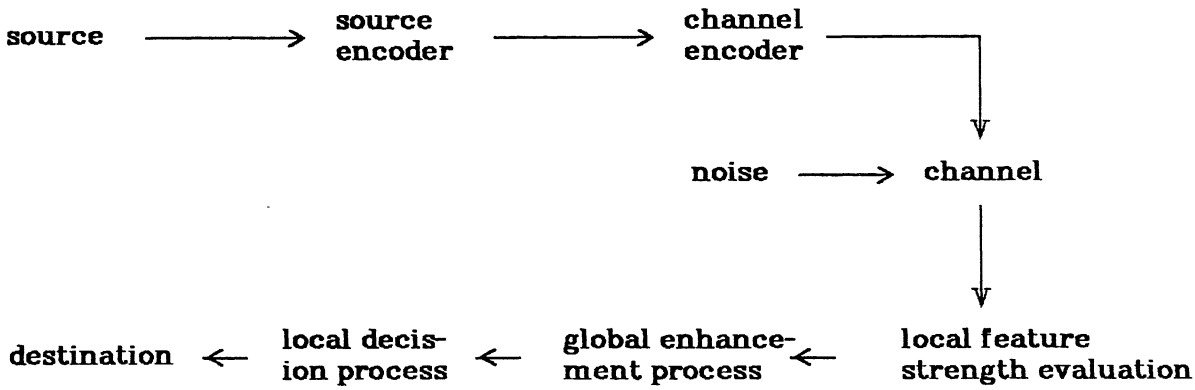

Figure 3.1: the context of the labeling problem.

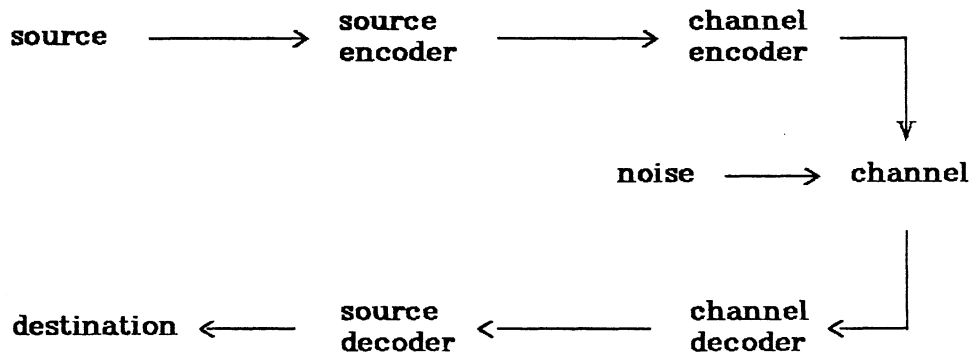

Figure 3.2: standard model for transmission of symbols through a noisy channel.

application to problems in communication.

In this model, the processes up through the *local feature strength evaluation* describe how the initial information to the graph labeling problem is generated.

Consider the application of relaxation labeling to curve and line enhancement. The source consists of a string of symbols representing the underlying labeling which we wish to estimate. In the case under consideration, this would be the "ideal line drawing" which describes the scene. Source coding entails the implicit relationships that occur among adjacent symbols. For most applications being considered, it is a passive process; the "encoding" occurs as a result of constraints imposed by the problem domain. Note that the underlying code closely resembles a convolutional code, except that the generating process is a system of productions. Channel coding involves the encoding of the ideal labeling into an observable signal, in this case, the (ideal) grey levels in the image. The channel and the incorporation of noise would correspond to the light reflected from the image, the sensor, digitizer - in fact everything between the image intensities and the digitized image in memory. Local feature strength evaluation is used to assign strength measures to each symbol or label; in this case, the response of local edge detectors. The result comprises the initial information available to the relaxation labeling algorithm.

It is the *global enhancement process* to which the relaxation techniques are being applied. The goal of this process is to incorporate all the information as to the correct labeling at a particular node, which is available in the initial labeling of the entire network, into the strength measures on the labels at that node. This being done, the process of *maxima selection* (choosing that label with the largest figure of merit), which is typical of all classification procedures, can be performed on a local basis. The result is an estimate of the initial labeling.

3.2. The Integration of Global Information to Make Local Decisions

There are two ways in which the resulting graph labeling problem can be approached. In the first place we may assume the existence of an underlying constraint network and a relation to be accepted and attempt to derive a meaningful extension to the continuous case. This is the approach which was implied by the

discussion of section 2. Alternatively, we may attempt to solve the labeling problem independent of the constraints imposed by the graph, and then attempt to implement the labeling algorithm in terms of a network.

3.2.1. In the case of the latter approach, we must first derive a function, F, which incorporates all (or perhaps a "reasonable" amount) of the labeling information over the extent of the network into the labeling at each node. To illustrate some of the issues pertaining to this, consider the vector valued random vector X such that each component X_i takes on values in the labeling vector $(\lambda_1, \lambda_2, \cdots, \lambda_m)$. If the probability distribution is such that the labeling at a component X_i is statistically independent of the labeling on the rest of the network then all the information concerning the correct labeling at that component is contained within its own labeling and a local labeling decision can be made. Otherwise we would have to apply some function $F:\Lambda\times X \rightarrow \Lambda\times X$ such that $F(\lambda\times X_i)$ is statistically independent of $F(\lambda'\times X_j)$ for all $j\neq i$. This would be a maximum a posteriori estimation process. For most applications statistical independence will be too strong a requirement so that uncorrelatedness may be a reasonable substitution. The function F may be determined by using the classical restoration/enhancement techniques in conjunction with system identification procedures.

Once the mapping has been determined, the function would have to implemented in the spirit of cooperative processes described above. The requirement that each node have limited memory means that at each iteration the updated label strengths must be an algebraic combination of the set of label strengths at the previous time steps only. That is, it would have to be computed by a set of iterative equations,

$$g_{v_i\lambda}, \text{ for all } v_i \in V, \ \lambda \in \Lambda$$

where the support[8] of each function corresponds to only those labels on nodes adjacent to v_i. The problem related to this decomposition can be expressed formally

[8]For a function of several variables, the *support* of that function is defined to be the set of those variables upon which it is dependent.

as follows:

Given: $F{:}R^n \rightarrow R^n$ with $F(\bar{x}) = \bar{y}$

where: $\bar{x} = (x_1, x_2, \ldots, x_n)$ and $\bar{y} = (y_1, y_2, \ldots, y_n)$,

Derive: $G{:}R^n \rightarrow R^n$ where,

$G = (g_1, g_2, \cdots, g_n)$ *such that*

if $\bar{y}^0 = \bar{x}$, and $\bar{y}_l^{t+1} = g_l(\bar{y}^t)$

then $\lim_{t \to \infty} \bar{y}^t = \bar{y} = F(\bar{x})$

where the components g_l are subject to the following constraint: each $g_l(\bar{y}^t)$ is dependent only on a subset, N(i), of the components of $\bar{y}^t$.

This is a classical problem in numerical analysis which was solved by Kolmogorov and Arnold in the early 1960's.[9]

3.2.2. Now consider an extension from the discrete to the continuous relaxation labeling processes. Assume we are given a discrete relaxation labeling network $C_n(\Lambda)$, that accepts the strings in a relation ρ_n. We may identify the string $\sigma_n = \lambda_1\lambda_2 \cdots \lambda_n$ with that point $\bar{p} \in K^*$ (the space of all unambiguous labelings, see previous section) such that

$$\alpha \in \Lambda,\ q_l(\alpha) = \begin{cases} 1 & \text{if } \alpha = \lambda_l \\ 0 & \textit{otherwise} \end{cases}$$

Furthermore, we can identify the relation ρ_n with the set of points represented by all strings $\sigma_n \in \rho_n$. A reasonable extension of the discrete labeling problem to the continuous case could be expressed as follows: given an ambiguous labeling $\bar{p} \in K$, select an element $\sigma_n \in \rho_n$, with the associated unambiguous labeling $\bar{q} \in K^*$, such that the "distance" between $\bar{p}$ and $\bar{q}$ is minimal. In other words, the underlying

[9] Unfortunately, the published results have not been translated from Russian into English, making the reference difficult to locate.

constraint network specifies, in terms of the set of all consistent labelings, the set of legal strings, and the aim is to find a labeling that is (1) consistent, and (2) such that the average of the initial values given to the symbols in that labeling is maximal. Furthermore, the labeling decision must be made on a local basis at each node, and if an iterative updating algorithm is used, information can flow only along the edges of the associated coarse graph.

Optimization techniques may be used to solve this problem. As such the method appears similar to the optimization approaches suggested by other researchers above. There is, however, an important difference: the fixed points of the algorithms will be related directly to a specific set of strings in the underlying discrete relaxation process, rather than to a set of consistent labelings, as defined in equation 2.13.

Example 3.1: It is not difficult to find an updating algorithm which is guaranteed to make an optimal labeling decision according to the above criterion when the coarse graph does not contain any cycles. Consider, for example, a constraint network with a linear topology and assume that the vertices are given labeling values taken from a continuous scale. The optimal labeling can be derived by making two "copies" of the network and using a Viterbi algorithm to assign labeling values to each vertex in the associated fine graph. In the first copy of the network the Viterbi algorithm is initiated at the left most node and moves to the right, and in the second copy of the network the algorithm moves to the left from the right most node. The algorithm is described formally as follows:

Let the vertices in the coarse graph be labeled as $v_1, v_2, \ldots, v_n$, with v_1 on the left and v_n on the right and with v_i being adjacent to v_{i-1} and v_{i+1}. Let $L_i^t(\lambda)$ represent the labeling values for label λ at vertex i at time t corresponding to the algorithm running from the left and similarly, let $R_i^t(\lambda)$ represent the labeling values for label λ at vertex i at time t corresponding to the algorithm running from the right. Let $r_{ij}(\lambda,\lambda')$

be the indicator function for the constraint relations of the underlying constraint network. Then the iterative procedure described by:

$$R_i^0(\lambda) = L_i^0(\lambda) = p_i^0(\lambda), \quad \textit{for all } i, \lambda, \tag{3.1a}$$

$$R_i^{t+1}(\lambda) = p_i^0(\lambda) + \max_{\lambda' \in \Lambda} \{r_{i,j+1}(\lambda,\lambda') \cdot R_{i+1}^t(\lambda')\}, \tag{3.1b}$$

$$L_i^{t+1}(\lambda) = p_i^0(\lambda) + \max_{\lambda' \in \Lambda} \{r_{i-1,j}(\lambda,\lambda') \cdot L_{i-1}^t(\lambda')\}, \tag{3.1c}$$

$$p_i^{t+1}(\lambda) = R_i^{t+1}(\lambda) + L_i(\lambda) - p_i^0(\lambda), \tag{3.1d}$$

will converge after no more than n time steps. The $r_ij(\lambda,\lambda')$ in equations 3.1a and 3.1b refer to indicator functions for the associated constraint network. The labeling values, $p_i^t(\lambda)$ for $t \geq n$ will reflect the sum of the initial labeling values for the optimum consistent labeling which has the label λ at vertex i. Thus, one can select, at each vertex, that label with a maximum labeling value and be guaranteed to be making an optimum decision.

Note that this algorithm can only be applied if the the set of legal strings is closed with respect to the coarse topology.

4. Summary

This report has given an overview and development of the relaxation labeling processes and the graph labeling problem. The material discussed here represents the initial stages of a research project aimed at establishing a formal basis for the relaxation labeling processes, or cooperative approaches to the continuous graph labeling problem.

Several important issues were covered. These issues include the development of the discrete relaxation labeling processes with respect to applications in computer vision. The development of the continuous graph labeling processes as a direct extension of the discrete relaxation operator, and some of the problems inherent in the method by which this extension was made were also discussed. An argument for the redefinition of the problem was made. This redefinition specified the way that the mapping from the original input feature vector to the output labeling is specified explicitly at the outset of the problem definition. Given the definition of this mapping, the problem of implementing the computation of that function in terms of an iterative, parallel scheme was treated only as a secondary consideration.

Finally, several suggested approaches to the problem definition and the associated solution were discussed. Our current research is towards further investigation of these suggested approaches.