THE UNIVERSITY OF MICHIGAN

COMPUTING RESEARCH LABORATORY

THE GRAPH LABELING MODEL AND ITS APPLICATION TO THE PROBLEM OF EDGE LINKING

Marc David Diamond

CRL-TR-30-83

AUGUST 1983

Room 1079, East Engineering Building
Ann Arbor, Michigan 48109
USA
Tel: (313) 763-8000

TABLE OF CONTENTS

LIST OF FIGURES

Figure

CHAPTER I

INTRODUCTION

1.1. Computer Vision and The Edge Linking Problem

The inability to obtain and interpret sensory feedback rapidly and accurately is often considered to be one of the biggest obstacles in the further development of automated manufacturing systems [DoR79]. Although computer vision is considered to be an important potential source of such sensory feedback, current vision systems are generally restricted to the processing of binary images, which are usually of practical value only when obtained under structured lighting conditions. It is generally accepted, however, that vision systems based on grey level images are necessary in order to satisfy the requirements of most flexible automation systems [TBB79].

Major difficulties exist with the use of grey level imagery in a real time environment. Most of these difficulties can in one way or another be related to the amount of data which must be processed in the typical visual recognition task. This fact has motivated research into parallel algorithms to apply to the various stages of a typical visual information processing task [DaR80], as well as the proposal, design, and implementation of several SIMD, and other special purpose architectures for computer vision tasks [COM83].

There is a sequence of the three basic processes, including edge detection, edge linking, and the recognition of objects from their shape, which are

incorporated into most approaches to the computer vision problem. Edge detection is, by its nature, a local operation, and hence there are no major computational barriers in performing this operation in real time on a cellular architecture. Although the recognition of objects from shape requires global knowledge, the consensus appears to be that the problem is not computationally intensive and can be solved on a general purpose computer in real time with existing techniques [Per78].

Edge linking, which is the process of grouping primitive features to describe the outlines of objects, is considered to be both the most difficult and the most important problem [BaB81,Mar75,Per78]. As with most pattern recognition tasks involving the grouping of primitive features, the edge linking problem is inherently complex, and a majority of the current approaches involve an enumerative search of some form. Thus, because of its combinatorial nature, it cannot currently be solved in a reasonable amount of time on existing hardware when taken in the context of most real world applications. Unlike the edge detection operation, there has been no prior attempt to develop formal techniques which could lead to the development of special purpose hardware to solve the edge linking problem.

1.2. The Graph Labeling Approach

A major direction of the current investigation is into the development of hardware based algorithms for the rapid solution of the edge linking problem. The approach taken here is based on a mathematical formulation, referred to as the graph labeling model. Although a heuristic concept of graph labeling and its application to syntactical grouping problems in pattern recognition have been proposed elsewhere [RHZ76,ZHR77], a formal sense of this concept is established for the first time as part of the current work. The result is a definition for the graph labeling problem as a specially structured set partitioning problem [Sai75,Mur76,BaP74]. As such, established techniques for 0-1 integer

programs can be brought to bear on its solution. The theory for the basic components of the algorithms which are proposed here - Lagrange duality, dynamic programming, implicit enumeration search strategies, and discrete relaxation - are well established. However, the use of the graph labeling model will allow the application of these techniques to the edge linking problem in a manner not possible with previous approaches. Furthermore, the special structure of the edge linking application is brought to bear on the development of means by which these techniques can be integrated into an implicit enumeration strategy.

The formal definition of a graph labeling model, and its application to the edge linking problem are the major contributions made within the context of this work. However, an initial investigation has also been made into certain aspects of the the structure of the graph labeling problem itself. There are several factors which have motivated this investigation. With current techniques, set partitioning problems involving 1,000 to 2,000 variables can usually be solved in a reasonable amount of time on a general purpose computer [BaP74,Mar74]. However, the equivalent set partioning problem applied to an edge linking task would involve on the order of 100,000 variables. Thus it is not clear that even special purpose hardware would be sufficient for this application and this then implies the use of heuristic approaches. Since this work emphasizes the relationship between the model and the application, as a contrast to past work in this area, it is desirable that the user have full knowledge of the behavior of a heuristic algorithm. For this reason an attempt has been made to develop a theory for the graph labeling model. It was our intent to include, for example, a description of the facets of the graph labeling polytope, and the linear programming relaxation of the graph labeling problem. Although this work is far from complete, some initial results have been established, and several conjectures, supported by simulation results are presented.

Theoretical work on the graph labeling problem is important for another reason: a great deal of research is currently in progress on the general set partioning problem because of its wide range of applications to problems in operations research [BaP74]. Since the graph labeling problem is a special case of set partioning, results pertaining to the latter would support this effort, in particular since special case studies can sometime lead to insights into the more general problem. Perhaps more important is the fact that, any set partioning problem can be transformed into an equivalent graph labeling problem, as will be shown. In many cases this transformation is not particularly useful. However, some applications, in particular those in which the associated constraint matrices are sparse, are well suited to be represented in this manner. Then the results and algorithms presented here can be applied directly.

1.3. Research Overview

The above discussion can be summarized by stating the major areas which are covered in the following chapters.

(1) The graph labeling problem is formulated as a special case of the general class of set partitioning problems, and some initial theory related to this problem is developed.

(2) The resulting graph labeling model is applied of the problem of edge linking in computer vision.

(3) The use of established techniques for the solution of set partitioning problems to the special case of the graph labeling problem and in particular to the special case of the edge linking application is discussed. The form that these algorithms take when they are applied to the structure of the graph labeling problem is studied with the general objective to exploit the parallelism of the underlying space, to the extent possible, to allow for decentralization of some aspects of these algorithms.

(4) Experimental results pertaining to the proposed approach to the edge linking problem are presented. Solutions to real world problems were not obtained even with the specialized techniques which have been developed here, because of the inherent size of the problems. However, the results of several experiments based on heuristic approaches are presented. The purpose of this is to demonstrate the potential in the proposed approach to the edge linking problem, and to encourage further work.

(5) For theoretical and experimental results which were not obtained but which are considered important for the further development of this topic, suggestions which hopefully will guide further work have been given both within the individual chapters and in the summary and conclusions.

1.4. The Organization

The remainder of this document is organized as follows: Chapter II is a discussion of both the edge linking problem and the graph labeling model as well as the relationship between the two. The origin of this model, which includes a survey of the past work which has lead to the proposition of the graph labeling approach to the edge linking problem, is covered. Finally, the justification for the form of the graph labeling model which is proposed here, is given. Chapter III contains the background and definitions required in the formal statement of the proposed graph labeling problem. The relationship between the graph labeling problem and well established problems in operations research are also given. Chapter IV discusses the theory of the graph labeling problem to the extent which has been developed within the context of this thesis. Chapter V covers the form of the various approaches to combinatorial problems cited above when applied to problems structured in accordance with the graph labeling model. Chapter VI illustrates the use of the graph labeling approach on real world problems. The work is summarized in chapter VII, in which suggestions for further

research are also given. Details concerning the application graph labeling model to the edge linking problem as well as other issues are contained in the Appendix.

CHAPTER II

BACKGROUND AND OVERVIEW

This chapter covers the basic aspects of the of past and present research which has lead to the current topic. It starts with a brief overview of the target application: the edge linking problem. In particular, the basic nature of the problem and means by which it is currently being approached will be discussed. A review of graph labeling approaches to the edge linking problem will be given. Difficulties with a lack of a theory associated with early approaches to graph labeling has lead to work on a means for formalizing this subject. In the context of this discussion, an argument is made for a formal model which is based on the assertion that graph labeling is in fact a classification problem.

2.1. The Edge Linking Problem

The function of a general vision system is assumed here to be one of recognizing the objects in a scene which is represented initially as a matrix of grey level values. In order to discuss the role played by the edge linking process within the context of such a vision system, the machine perception problem must be approached by interpreting the environment through multiple levels of description. The initial level of description is given as a primitive feature map, edge map, or *primal sketch* [Mar76], which results from the application of detectors sensitive to a basic set of primitive features at a local level. The next level of description is achieved by grouping these features into sets of

continuous contours. The function of a contour is to delineate points in the original image which separate an object from its background. The result of this grouping process is referred to as a line drawing or cartoon. Once a line drawing representing the original image has been extracted, objects can usually be easily recognized from the shape of the boundaries which circumscribe them. Even though most actual approaches are more involved than this basic description suggests, it is within the context of this approach that the current work derives its relevance.

The problem of edge linking in computer vision addresses itself to the intermediate level problem of grouping primitives to form smooth contours. Current and past approaches to edge linking belong to a continuum of strategies which range from procedural to formal, i.e., where a mathematical model defining the correct grouping is specified. On another dimension, these strategies may attempt to incorporate global knowledge (top-down, model-based[1], or knowledge-based approaches) or may perform the linking process based strictly on local information. Procedural approaches are characterized by the line tracking algorithms such as, for example, the Binford-Horn line finder [Hor71], as well as other linking strategies including the relaxation labeling processes described below. The knowledge based approaches are represented by the work of Shirai, [Shi75] and includes, for example, the Hough transforms [DuH72], and the generalized Hough transforms [Bal81].

In the discussion presented below, an approach to the problem of edge linking is considered in which a formal model is specified and relationships between primitives in the grouping process are defined on a local basis. This approach is best represented in past and present work by graph searching

---

[1] The reference to the "model" in this case may not necessarily refer to a formal mathematical model, but rather may be some contour based description of what objects can actually exist in the image.

methods [BaB82] wherein the image raster is viewed as a graph in which each pixel is a vertex connected to some subset of the pixels in its immediate neighborhood, as illustrated in figure 2.1. In a state space search applied to this graph, weights are placed on transitions, which are intended to reflect the likelihood of a contour actually existing between two adjacent pixels in the image. These weights are usually derived in conjunction with the output of the primitive feature detection process. A formal definition of the problem results, which is equivalent to finding a maximum weighted path through a graph, often between two predefined points.

Most of the well-established implicit enumeration strategies, appearing in one form or another, have been applied to the underlying graph searching problem.

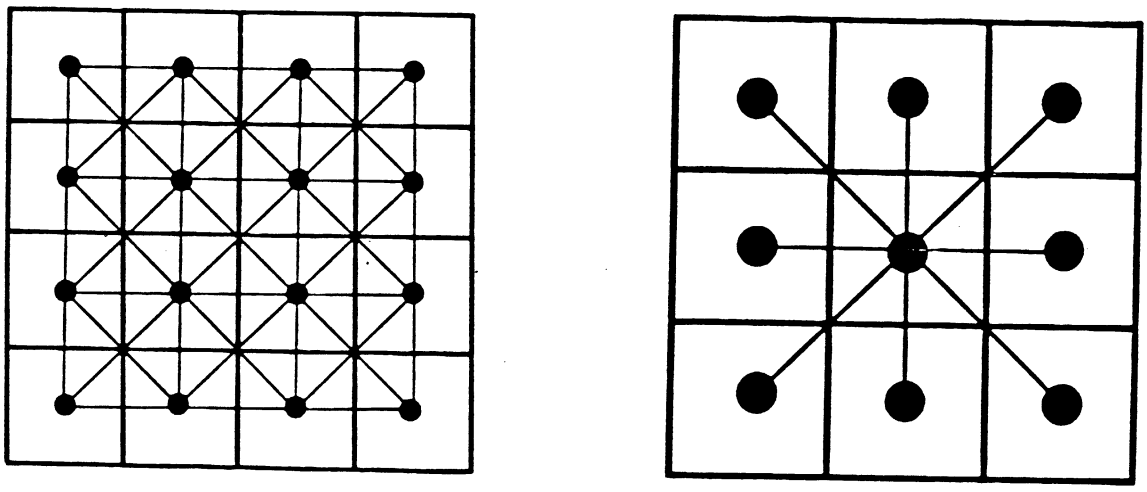


Figure 2.1a (left): The underlying graph in the graph searching approaches and its relationship to the original image raster.
Figure 2.1b (right): The immediate neighborhood of a given vertex (center pixel).

Although, because of the size of the underlying space, heuristics are almost always used to augment the search process. Thus, it has been generally accepted that limitations on the success of the graph searching approaches are not based as much on the relationship between the model and the application (i.e., how these maximally weighted paths result in a useful definition of the contours in the image) as the fact that they must ultimately deal with the combinatorics of the optimization process itself. As noted previously, it is expected that the approach to edge linking based on the graph labeling model will also involve heuristics, however, it is hoped that the resulting approaches will be more robust, and the effect of the heuristics better understood. Comparitive studies and surveys on the various search strategies applied to this problem can be found, for example, in [AsM78,LWW78,Ros81].

2.2. The Graph Labeling Model and the Edge Linking Problem

In the graph labeling approach studied here, the relationship between the model and the application is essentially the same as in the graph searching approaches discussed above. However, the form the particular model is better suited to take advantage of some of the structure which is inherent in the underlying problem. Furthermore, this model allows for the application of certain techniques to the solution of the optimization problem, which cannot be directly applied within the context of the graph searching model.

A graph labeling problem involves the assignment of a single label λ_j from some set, $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$, of possible labels to every vertex of a graph. The result is referred to as a *labeling* of the graph.[2] The assignment process is

---

[2]This use of the term "labeling" is unfortunate because it corresponds to another definition in graph theory. The more common definition of a labeled graph is one in which the vertices are distinguished from each other [Har72], as they are here anyway. However, this word will be used in the sense defined above throughout, in order to be consistent with past and present work in graph labeling problems in the pattern recognition area. It is suggested that in further work a different term be found.

performed in conjunction with initial information as to the correct label to choose for each vertex, and often in conjunction with contextual information as to how labels on adjacent vertices interact. In the graph labeling approaches to the edge linking problem the underlying graph is defined in the same as it was for the search strategies discussed above. The label set corresponds to a set of primitive features, and the initial information is taken to be a strength measure, merit figure, or *cost*, c_{ij}, associated with each feature or label λ_j on each pixel or vertex, v_i, of the graph. As was the case in the graph searching methods, the costs are assumed to have been derived from the output of the primitive feature extraction process. Examples of scene events or primitive features corresponding to possible label sets for the edge linking application are given in the Appendix.

Given the initial costs, one means by which labels can be assigned to the vertices of a graph is simply to choose a label at each vertex such that the corresponding strength measure is maximal. This strategy is often referred to as a "local maxima selection" rule [ZLM81]. A local maxima selection rule is usually not, in and of itself, sufficient to derive a "meaningful" labeling, however. Figure 2.2a shows a typical result of using this strategy on a set of initial costs.[3] The underlying label set in this case corresponds to the scene events of the Appendix. Figure 2.2b is the desired result of the edge linking process. This line drawing results from the correct assignment of labels to every pixel in the image. The basic assumption that is being made here is that the correct label can be assigned to a given pixel by considering the initial labeling values for each label on every pixel in the image. However, this process requires more information than is available strictly among the labeling values for labels at that pixel.

---

[3]The details on the derivation of these costs are covered in Chapter V

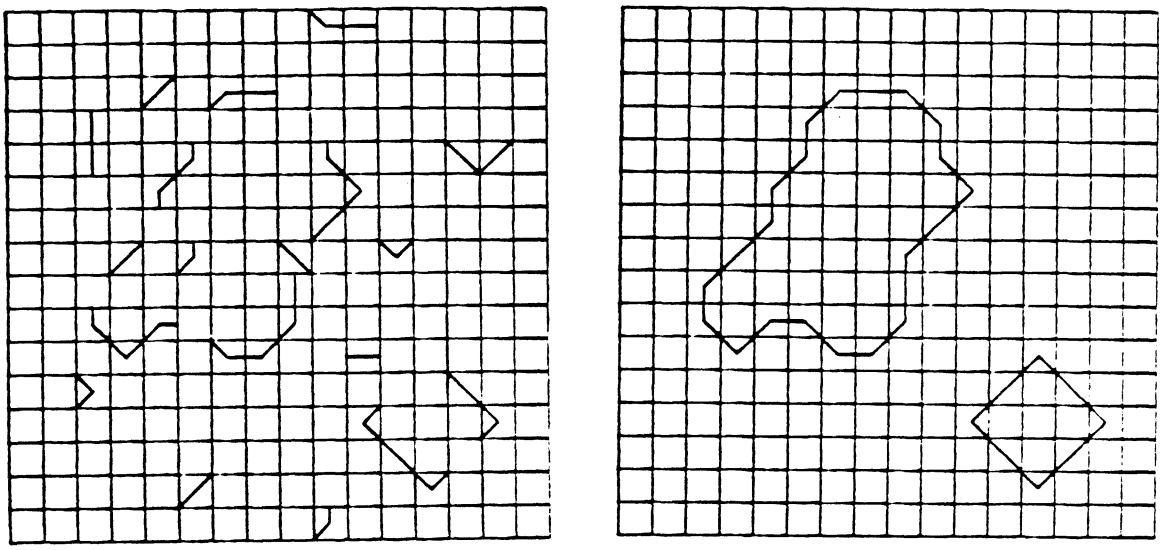


Figure 2.2a (left): Initial labeling derived from a set of hypothetical labeling values based on the label set of Appendix A.

Figure 2.2b (right): The desired labeling, showing a completed contour.

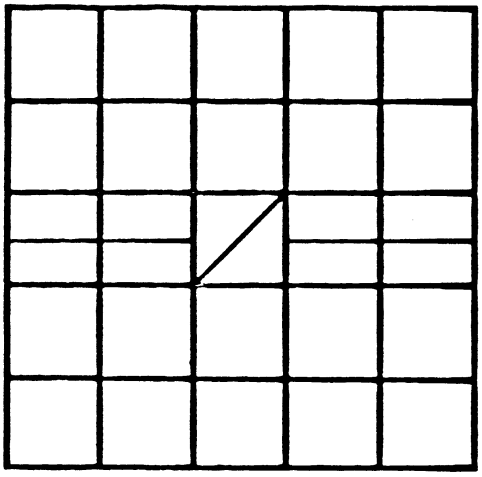
This point is considered further in figure 2.3a which shows a 5 x 5 image. Hypothetical values for the initial costs from which, by a local maxima selection process, the labeling of this figure has been derived are given in figure2.4. If it is assumed that the signal generating the costs is in fact an encoding of a scene involving only smooth contours, then it seems reasonable to conclude that the correct labeling is the one shown in figure 2.3b. This is an assertion that the center pixel was initially mislabeled. There is a great deal of evidence supporting this assertion in the labeling values associated with neighboring pixels. The problem is how this evidence should be integrated into a pixel labeling correction scheme. The central issue of the work described in this thesis is the development of models upon which such schemes can be based and the

development of hardware efficient algorithms to implement them.

2.3. Past and Present Approaches to Graph Labeling

2.3.1. The Relaxation Labeling Processes

The graph labeling approach to edge linking described above was initiated with an investigation into relaxation labeling processes and their application to the problem of curve and line enhancement [Wal75,RHZ76,ZHR77]. The relaxation labeling processes refer to a class of algorithms in which the costs associated with each label on a given vertex of the graph are updated in an iterative parallel manner based on the current labeling values at that vertex and the costs associated with labels on adjacent vertices. Let



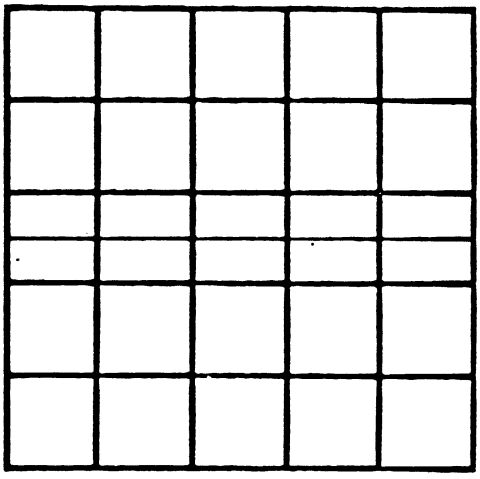


Figure 2.3a (left): A 5 x 5 image labeled from the hypothetical labeling values of figure 2.4 using the labels of the Appendix.

Figure 2.3b (right): The desired labeling.

$c^t = (c^t_{11}, c^t_{12}, \ldots, c^t_{nm})$ be the cost vector at a given iteration, t. Then, for a label λ_j on a vertex v_i, the cost c^t_{ij} is a function, $c^t_{ij} = F_{ij}(c^{t-1})$, where F_{ij} depends only on the components c^{t-1}_{ij} and $c^{t-1}_{i'j'}$ for all $i' \in N(i)$ and all $\lambda_{j'} \in \Lambda$, where $N(i)$ is the set of vertices adjacent in the graph to vertex v_i.

The stated goal of these procedures is to incorporate information about the correct labeling of a particular vertex, which is assumed to be distributed throughout the network, into the labeling values at that vertex. An overview of the apparent strategy is shown in figure 2.5. The global enhancement process is represented as a map $s: R^{nm} \rightarrow R^{nm}$ (note[4]) which is defined as $s = \lim_{n \to \infty} F^n(c^0)$

0.143 0.075 0.782	0.038 0.101 0.861	0.142 0.100 0.758	0.069 0.031 0.900	0.119 0.019 0.862
0.151 0.021 0.828	0.126 0.063 0.811	0.096 0.036 0.868	0.018 0.067 0.915	0.020 0.115 0.865
0.799 0.082 0.119	0.803 0.154 0.043	0.134 0.711 0.155	0.779 0.152 0.069	0.751 0.165 0.084
0.154 0.116 0.730	0.033 0.105 0.862	0.164 0.109 0.727	0.161 0.087 0.752	0.151 0.167 0.682
0.138 0.079 0.783	0.060 0.110 0.830	0.062 0.157 0.781	0.102 0.007 0.891	0.136 0.129 0.735

Figure 2.4: Hypothetical (but typical) labeling values which lead to the mislabeling of the center pixel of figure 2.3. Within each pixel, the labeling values are shown for labels λ_{16}, λ_{19}, and λ_{21} of the label set of the Appendix.

[4]It is assumed that there are n vertices in the underlying graph for the graph labeling labeling problem, and m labels in the label set.

assuming this limit exists, where $F = (F_{11}, F_{12}, \ldots, F_{nm})$. The function m maps from the current labeling values into a labeling according to the local maxima selection rule. The labeling $\bar{\lambda}$ is derived directly from the initial costs whereas the labeling $\bar{\lambda}'$ is derived from the updated costs, assuming the process converges. The hope is that the labeling $\bar{\lambda}'$ will be an improvement, in some sense, over the labeling, $\bar{\lambda}$.

Two forms of labeling processes, discrete and continuous, are distinguished in the literature by the nature of the local information and the representation of the contextual information. In a discrete labeling process the initial costs take on only values of 0 or 1, where $c_{ij} = 1$ if it is possible to assign label λ_j to vertex v_i, and $c_{ij} = 0$ otherwise (based on some external criteria). The contextual information is expressed in terms of binary relations defined on the label set, Λ,

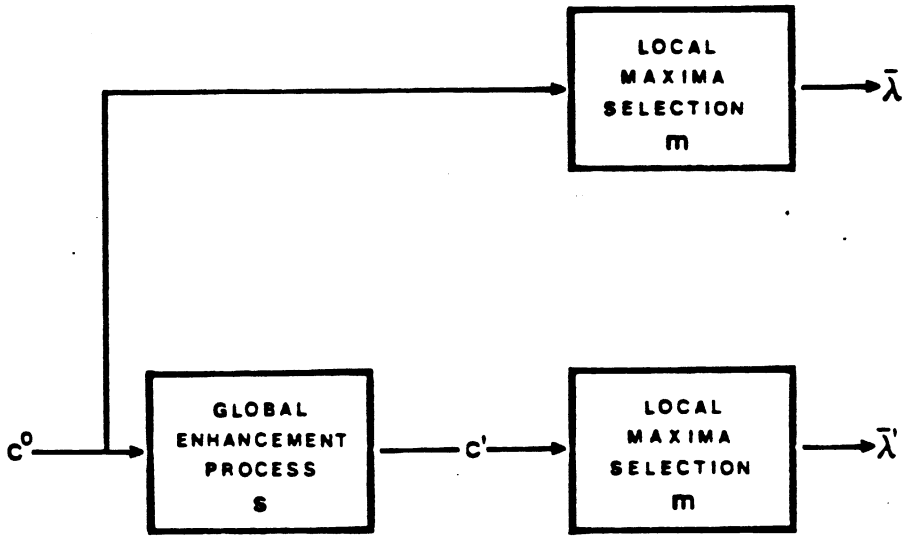


Figure 2.5: An overview of the basic approach incorporated into the relaxation labeling processes.

which make explicit those pairs of labels which can occur simultaneously on adjacent vertices. In a continuous labeling process, the costs are taken to be real numbers, usually $c_{ij} \in [0,1]$ and the contextual information is represented by *compatibility* coefficients, $r_{ii'}(\lambda_j,\lambda_{j'})$, between pairs of labels λ_j and $\lambda_{j'}$ on adjacent vertices v_i and $v_{i'}$, where in most cases either $r_{ii'}(\lambda_j,\lambda_{j'}) \in [0,1]$, or $r_{ii'}(\lambda_j,\lambda_{j'}) \in [-1,1]$. For highly compatible features the corresponding compatibility measure would be 1, and highly incompatible features the compatibility measure would be 0, or -1, depending on the range that these values are allowed to assume. For example, in the curve and line enhancement application, the compatibility measure for the labels corresponding to two horizontal edge segments on horizontally adjacent pixels would be 1, whereas the compatibility measure for an horizontal edge segment and an adjacent vertical edge segment would be, say -1, as shown in figure 2.6.

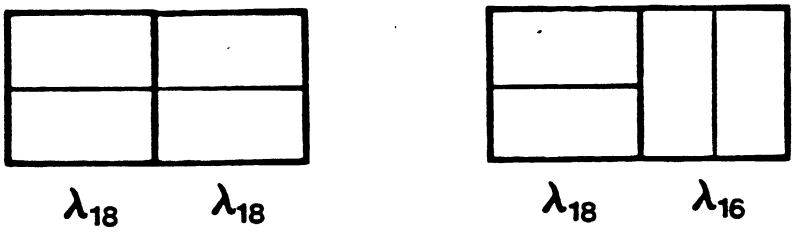


Figure 2.6: Compatibility between two pairs of labels according to the approach of the relaxation labeling processes.

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 process, subsequent work [Mon74,RHZ76] cast Waltz filtering in terms of a set of parallel, iterative equations; that is, as a cooperative process. Most of the theory of constraint networks and discrete relaxation which has been established to date appears in a paper by Montanari [Mon74] and Rosenfeld et al. [RHZ76]. Although some further contributions can be found in the work of Freuder [Fre78], Mackworth [Mac77], 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]. Further applications can also be found in surveys by Davis and Rosenfeld [DaR80], and Haralick and Shapiro [HaS79]. Because of their importance to the work pursued in this thesis, issues related to constraint networks and the discrete relaxation labeling processes are treated further in the next chapter.

The updating rules for the continuous relaxation labeling processes were originally derived as a direct extension of the discrete relaxation labeling process. However, whereas the discrete relaxation labeling processes are well understood the extension to the continuous case has proven to be without basis. An example of such an updating rule is the one which was originally proposed with for the edge enhancement problem given as [RHZ76]:

$$c_{ij}^{t+1} = F_{ij}(c^t) = \frac{\sum_{i' \in N(i)} c_{ij}^t \, [\, 1 + q_{ii'}^t \,]}{\sum_{\lambda_{j'} \in \Lambda} \sum_{i' \in N(i)} [\, c_{ij'}^t \, [\, 1 + q_{ii'}^t \,]\,]} \qquad 2.1$$

where

$$q_{ii'}^t = \sum_{\lambda_k \in \Lambda} r_{ii'}(\lambda_j, \lambda_k) \, c_{i'k}^t \qquad 2.2$$

Equation 2.2 expresses the "support" for a label λ_j on pixel i as a average of the current labeling values on adjacent pixels weighted by the compatibility coefficients. Equation 2.1 is a means for combining these support measures into an updated labeling value. As before, Λ is the set of all possible labels, c_{ij}^t are the costs at iteration t and $r_{ij}(\lambda,\lambda')$ are the compatibility coefficients.

The basis for the design of the continuous relaxation labeling processes is the assumption that a label which is more compatible with the current labeling on its neighborhood will receive a greater share of the support so that its value will increase on the next iteration. However the behavior of the updating rule is dependent, among other things, upon the actual values given to the compatibility coefficients. There is currently no formal means for assigning these values, although ad hoc approaches based on sample correlation coefficients or mutual information [PeR78], as well as others [Yam79] have been used.

Other updating rules have been proposed. For example, in the product rule updating method [Kir80], equation 2.1 has been replaced by

$$c_{ij}^{t+1} = \frac{c_{ij}^t \prod_{i' \in N(i)} q_{ij}^t}{\prod_{\lambda_{j'} \in \Lambda} c_{ij'}^t \prod_{i' \in N(i)} q_{i'j'}^t} \qquad 2.3$$

Another updating rule which has been proposed [Pel80] is given by replacing equation 2.1 with:

$$c_{ij}^{t+1} = \frac{c_{ij}^{t} \sum_{i' \in N(i)} q_{ij}^{t}}{\sum_{\lambda_{j'} \in \Lambda} c_{ij'}^{t} \sum_{i' \in N(i)} q_{i'j'}^{t}} \qquad 2.4$$

Equations 2.3 and 2.4 followed from certain assumptions which were made in attempt to put at least some aspects of the relaxation labeling processes on a formal basis. It is felt however [HuZ80,DiG82,DNG82] that these assumptions are not justifiable and certain basic aspects of the processes are still not well understood. One of the problems in designing a numerical procedure in an ad hoc manner is that there is no guarantee that the labeling values will ever converge to a fixed point. It has been our experience, in fact, that they usually do not. Even if they did, there is still no understanding as to the relevance of the final labeling values to the original application. Clearly, for this to happen, models must be developed which can related the meaning of a given labeling to the related application.

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 hardware implementation seems also to have had some effect on its growth. 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.

A survey of recent applications for continuous relaxation labeling techniques would require an unreasonable amount of space. Reference to only a few of these applications are made here, while surveys by Rosenfeld [Ros79,Ros81] and Davis and Rosenfeld [DaR80] are more complete. One of the earliest applications of relaxation labeling was to curve and line enhancement [ZHR77] as discussed above. Marr et al. [MaP76,MPP77] employs relaxation in the correspondence problem for stereo vision. 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], and the solution to substitution ciphers [PeR79].

2.3.2. Optimization Approaches to the Graph Labeling Probiem

The problem of the numerical behavior of existing labeling algorithms as well as a lack of theory for the continuous relaxation labeling processes was addressed in the development of what will be referred to here as optimization approaches, to the definition of the mapping **s** of figure 2.5. For example, in one such definition [HuZ80], a given initial labeling vector $c = (c_{11}, c_{12}, \ldots, c_{nm})$, is combined with the matrix of compatibility values:

$$R = \begin{bmatrix} r_{11}(\lambda_1,\lambda_1) & r_{11}(\lambda_1,\lambda_2) & & r_{11}(\lambda_m,\lambda_m) \\ r_{12}(\lambda_1,\lambda_1) & r_{12}(\lambda_1,\lambda_2) & & r_{12}(\lambda_m,\lambda_m) \\ & & & \\ & & & \\ r_{nn}(\lambda_1,\lambda_1) & r_{nn}(\lambda_1,\lambda_2) & & r_{nn}(\lambda_m,\lambda_m) \end{bmatrix}$$

to define the quadratic program:

$$s = \max \; c^T R c$$

subject to:

$$\sum_{j=1}^{m} c_{ij} = 1, \quad i=1,\ldots,n.$$

Research into other optimization based means for defining the enhancement map have appeared in several recent publications [FaB81,Fau81]. In most cases, the associated objective functions are designed so as to allow for (at least partial) decomposition of the optimization procedure, in order to maintain the cooperative spirit of the relaxation labeling processes, or the objective function is defined locally at the outset.

2.4. The Foundations of a Formal Model

Although the optimization approaches address the problem of putting the enhancement map on a formal basis, an important fact concerning these methods is still being ignored: the overall process is in actuality a map $m(s(c^0))$ from a continuous subset of R^{nm} into the discrete set of all possible labelings. The argument which is being made here is that in order for the graph labeling model to be in some sense useful, the user should have an understanding as to what that mapping is, and how it relates to a given application. This essential feature is missing from the graph labeling algorithms discussed above. Mappings from a continuous domain into a discrete domain, when defined within the context of problems of pattern recognition are usually treated as classification rules. Thus, in order to put the entire subject on a formal basis, the approach which is taken here is to first specify which labelings are legal and which are illegal in a manner consistent with a given application, so as to define a set of classes. Then the classification rule will be specified. Finally, the problem of implementing the given rule (in a hardware efficient manner, to the extent possible) will be addressed.

The mapping which will be studied within the context of this thesis will be the "max-sum" classification rule. That is, the goal will be to choose a legal

labeling such that the sum of the initial labeling values is maximal. The reason for this choice is that it can be related to several well established classification rules from statistical decision theory. For example, assume the input to the feature detection process is a 0-1 labeling vector observed in the presence of noise, that is, $c_{ij} = 1$ if λ_j is the actual label assigned to vertex v_i and $c_{ij} = 0$ otherwise. Then a simple transformation of the form

$$c'_{ij} = -(1 - c_{ij})^2 - \sum_{j'=1,\ldots,m,\ j' \neq j} c_{ij'}^2 \qquad 2.5$$

converts this process into the well established nearest neighbor classification rule. A transformation of the form

$$c'_{ij} = \log (Pr (c_{ij} | \lambda_j)) \qquad 2.6$$

converts the max-sum decision rule into the well established maximum likelihood classification rule, so long as all the labeling values, given the associated label, are conditionally independent.

After the classification rule has been selected, the issue remains as to how to specify the classes. This will in general be problem dependent, however, it is assumed here that the problem domain can be modeled in such a way as to specify an underlying constraint network. As discussed in the next chapter, a constraint network divides the set of all possible labelings into those which are *consistent* (i.e. legal) and those which are inconsistent. Consistency is a global property which is, however, defined on a local basis. In the edge linking application a pair of labels are considered to be locally consistent if the scene events they represent do not have an edge segment broken across a pixel boundary, and inconsistent otherwise. Examples of consistent and inconsistent pairs of labels are given in the Appendix. A labeling is considered to globally consistent so long as every pair of labels in it are locally consistent. The consistent labelings will be then taken to be the classes in the classification

process.

In the graph labeling approaches to the curve and line enhancement and edge linking problems discussed above, the criterea used to judge the effectiveness of a given procedure has been with past approaches basically subjective, e.g. how good the resulting line drawing looks. This is symptomatic of the lack of connection between the application and any formal methodology. With just a little reflection however, one can establish a more formal basis for the edge linking application, since a line drawing looks good if it in some sense represents the objects in the image and if there are no broken line segments in it. The set of all consistent labelings defined here are those which do not contain broken lines. Evidence for the fact that the result of using this model should accurately reflect the objects in the image is taken from the success of the graph searching methods and the relationship between this model and the model used here.

CHAPTER III

THE PROBLEM AND RELATED DEFINITIONS

The focus of this chapter is on the proposed definition of the graph labeling problem and its relationship to certain classes of 0-1 integer programs. In order to present this formulation, a review the basic definitions and ideas related to constraint networks is necessary. We give here a brief discussion of those aspects of the topic which are important to the work described in following chapters. However, it should be noted that constraint networks are in themselves an important topic relevant to problems in almost every major area of artificial intelligence. More detailed treatments of this subject can be found in [Dia82] and in the references given in the survey of chapter II. Finally, it should be noted that the treatment of the topic given here is not necessarily consistent with the treatments found elsewhere. This discussion is structured to the use of constraint networks in the definition of the graph labeling problem itself.

3.1. Constraint Networks

Definition: A constraint network is defined here to be a triple, $C(G,\Lambda,R)$ consisting of:

(1) A graph $G = (V,E)$, with vertex set $V = \{ v_1, v_2, \ldots, v_n \}$, and edge set $E \subseteq V \times V$,

(2) A set, $\Lambda = \{ \lambda_1, \lambda_2, \ldots, \lambda_m \}$ of symbols, called *labels*, and,

(3) A class, $R = \{ R_{ii'} \}$, of binary relations, referred to as *constraint relations*, defined on the label set: $R_{ii'} \subseteq \Lambda \times \Lambda$, for all ii' wherein there is one such relation for every edge $v_i v_i' \in E$ (note[1]).

Although a more general case may be considered, it will be assumed here that the constraint relations are symmetric, that is $R_{ii'} = R_{i'i}^T$. The graph labeling problem involves assigning a unique label, or set of labels to each vertex of the graph. A label, λ_j assigned to vertex v_i is considered to be (pairwise, or locally) *consistent* with a label λ_j' assigned to vertex v_i', if $(\lambda_j, \lambda_j') \in R_{ii'}$ and inconsistent otherwise.

Example 3.1: Consider the graph $G = (V,E)$ with $V = \{ v_1, v_2, v_3, v_4 \}$, and $E = \{ v_1v_2, v_2v_3, v_3v_4 \}$ (i.e., G is a path with four vertices). Let $\Lambda = \{ \lambda_1, \lambda_2, \lambda_3 \}$, and define:

(a) $R_{12} = \{ (\lambda_1,\lambda_1),(\lambda_2,\lambda_2),(\lambda_3,\lambda_3) \}$,

(b) $R_{23} = \{ (\lambda_1,\lambda_3),(\lambda_2,\lambda_2),(\lambda_3,\lambda_1) \}$,

(c) $R_{34} = \{ (\lambda_1,\lambda_1),(\lambda_2,\lambda_2),(\lambda_2,\lambda_3),(\lambda_3,\lambda_3) \}$,

Figure 3.1a illustrates the graph G, and the associated constraint relations. In general, G will be referred to as the *underlying* or *coarse graph* for the constraint network C. The form of the underlying graph defines the network topology and has important implications for algorithms for the graph labeling problem as is discussed in chapter IV. Figure 3.1b is the *direct graph* corresponding to the given constraint network. In this n-partite graph, (here n = 4), there is a vertex, v_{ij}, for every label λ_j on each vertex v_i. An edge exists between a pair of vertices v_{ij} and $v_{i'j'}$ if and only if v_i is adjacent, in G, to v_i', and $(\lambda_j,\lambda_j') \in R_{ii'}$, that is, if and only if (λ_j,λ_j') is a locally consistent pair of labels. Figure 3.1c shows a

---

[1] Note: an edge with end points V_i and V_i' will be denoted as V_iV_i' throughout.

graph, which like the direct graph, has a vertex for every label on every vertex of the underlying graph G. However, in this case, there is an edge between a pair of vertices v_{ij} and $v_{i'j'}$ if and only if v_i is adjacent, in G, to $v_{i'}$, and $(\lambda_j,\lambda_{j'})$ notelof $R_{ii'}$, that is, if and only if $(\lambda_j,\lambda_{j'})$ is a locally inconsistent pair of labels. This graph will be referred to as the *complement* of the product graph, or simply, the complement graph. A convention which will be used throughout is to show edges between invalid pairs of labels as dashed lines. Finally, the graph obtained from the complement graph by adding an edge between every pair of labels on a given vertex will be referred to as the *augmented complement graph.*

Labelings: Associate with each vertex, $v_i \in V$, a subset $\Lambda_i \subseteq \Lambda$, of the label set. An n-tuple $L = (\Lambda_1, \Lambda_2, \ldots, \Lambda_n)$ of such subsets will be called a *labeling* of G. For a given vertex, v_i, each label $\lambda_j \in \Lambda_i$ is referred to as being *assigned* to that vertex. In order to resolve possible ambiguity, a specific label, say, λ_j associated with a specific vertex, say v_i will occasionally be referred to as λ_{ij}.

Consistency: A labeling, L, is considered to be (globally) *consistent*, if the following condition holds: for all i=1,...,n and for all $\lambda_j \in \Lambda_i$ associated with vertex v_i, there exists for every adjacent vertex, $v_{i'}$, at least one label $\lambda_{j'}$ in the associated label subset $\Lambda_{i'}$ such that $(\lambda_j,\lambda_{j'}) \in R_{ii'}$, that is, such that $(\lambda_j,\lambda_{j'})$ is a consistent pair of labels. In other words, every label in each label subset associated with the vertices of G must be consistent with at least one label in the label subsets associated with every adjacent vertex.

Ambiguity: A labeling, L will be called *unambiguous* if $|\Lambda_i| = 1$ for all $\Lambda_i \in \Lambda$, and *ambiguous* if there exists an i such that $|\Lambda_i| > 1$. A labeling such that $|\Lambda_i| = 0$ for all i will be referred to as the empty, or null labeling. An unambiguous labeling L will often be denoted as $\bar{\lambda} = \lambda^1\lambda^2 \cdots \lambda^n$ where λ^i is the unique member of Λ_i. By

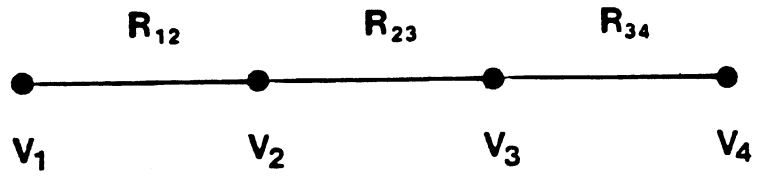


Figure 3.1a: Underlying graph for the constraint network of example 3.1.

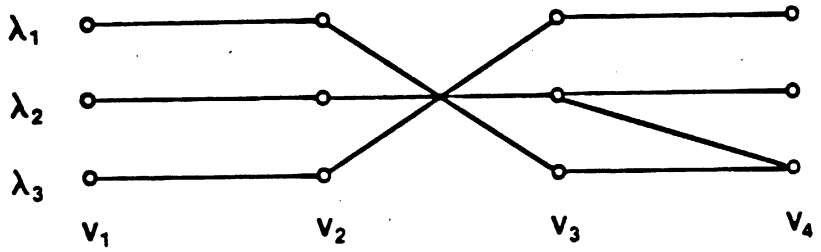


Figure 3.1b: Product graph for the constraint network of example 3.1.

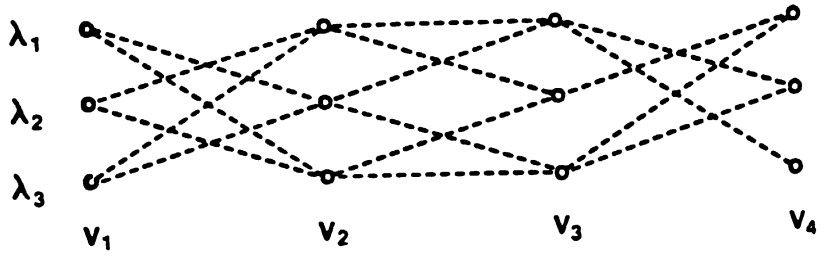


Figure 3.1c: Product graph for the constraint network of example 3.1.

convention, the notation λ^i will be used to denote a specific label assigned to vertex v_i.

Example 3.2: In diagrams involving constraint networks, a given labeling of that constraint network will be illustrated by filling in the circles representing the labels associated with each vertex of the graph, if they are elements of that labeling. Consider the constraint network of example 3.1. Let $L = (\Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4)$ be a labeling with

$$\Lambda_1 = \{\lambda_1, \lambda_3\}, \quad \Lambda_2 = \{\lambda_1, \lambda_3\}, \quad \Lambda_3 = \{\lambda_1, \lambda_2, \lambda_3\}, \quad \Lambda_4 = \{\lambda_1, \lambda_3\}.$$

The representation of this network is shown in figure 3.2a. L is an unambiguous labeling which is, however, inconsistent. In particular, the label λ_2 on vertex v_3 has no support on vertex v_2. That is, there is no label λ_j, $j=1,2,3$, in the label subset Λ_2 such that (λ_j,λ_2) is consistent. The labeling in figure 3.2d is however, a consistent labeling.

Lattice structure of labelings: The set of all possible labelings of a given constraint network defines a lattice. Let $L = (\Lambda_1, \Lambda_2, \ldots, \Lambda_n)$, and $L' = (\Lambda'_1, \Lambda'_2, \ldots, \Lambda'_n)$ be labelings. The lattice operations are given as:

(1) labeling intersection (meet):

$$L \cap L' = (\Lambda_1 \cap \Lambda'_1, \ \Lambda_2 \cap \Lambda'_2, \ldots, \ \Lambda_n \cap \Lambda'_n),$$

(2) labeling union (join):

$$L \cup L' = (\Lambda_1 \cup \Lambda'_1, \ \Lambda_2 \cup \Lambda'_2, \ldots, \ \Lambda_n \cup \Lambda'_n)$$

The partial order is defined by labeling containment: $L \subseteq L'$ if and only if $\Lambda_i \subseteq \Lambda'_i$ for all i. Other set operations will be extended to labelings in the same manner, that is, on a component by component basis. For example

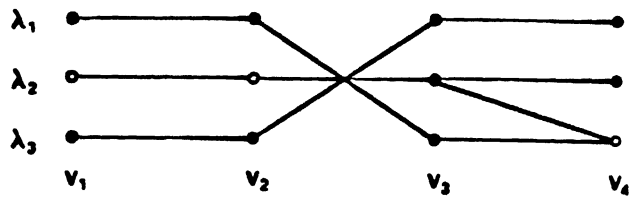


Figure 3.2a: Discrete relaxation: iteration 0.

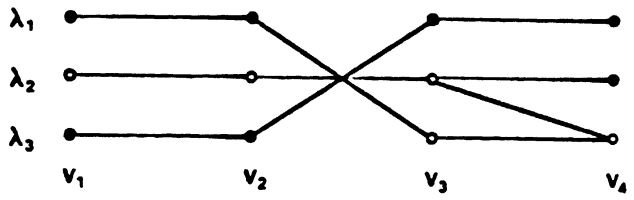


Figure 3.2b: Discrete relaxation: iteration 1.

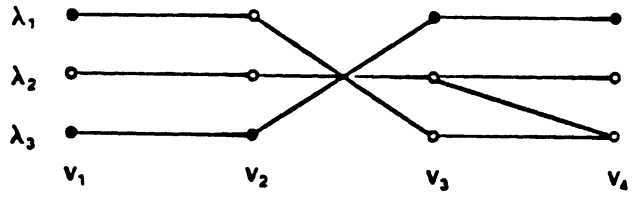


Figure 3.2c: Discrete relaxation: iteration 2.

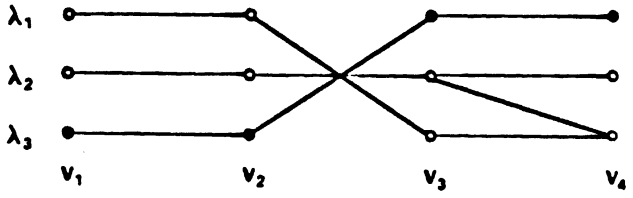


Figure 3.2d: Discrete relaxation: iteration 3.

$$L - L' = (\Lambda_1 - \Lambda'_1,\ \Lambda_2 - \Lambda'_2, \ldots,\ \Lambda_n - \Lambda'_n).$$

The null labeling is $L_\phi = (\phi, \phi, \ldots, \phi)$, the universal labeling is $L_U = (\Lambda, \Lambda, \cdots, \Lambda)$.

Induced graphs, projections, labeling induced subnetworks: Let $G = (V,E)$ be a graph. A subgraph induced by a subset $A \subseteq V$ of the label set is the graph $G_A = (A,E_A)$ such that (v_1,v_2) is in E_A if and only if $(v_1,v_2) \in E$ and $\{v_1,v_2\} \subseteq A$. The projection of the constraint network C onto the subgraph G_A is a constraint network: $C_A = (G_A,\Lambda,R_A)$ with underlying graph G_A and $R_{ii'} \in R_A$ if and only if $(v_i,v_{i'}) \in E_A$. The projection of a labeling $L = (\Lambda_1, \Lambda_2, \ldots, \Lambda_n)$ onto A, is the labeling, L_A of the projected constraint network C_A with $\Lambda_i \in \Lambda_A$ if and only if $\Lambda_i \in \Lambda$ and $v_i \in A$. The subnetwork induced by the labeling Λ is the subgraph of the product graph induced by the vertices corresponding to the labels in L.

Minimal constraint networks: A constraint network, in which every label on every vertex participates in an unambiguous labeling is called *minimal*. A labeling induced subnetwork may not be minimal even considering the labels participating in the labeling itself. It turns out that this fact alone accounts for the difficulty of the graph labeling problem (i.e. the fact that it is NP-complete).

Fundamental cycles, fundamental anticycles: Let $C = (G,\Lambda,R)$ be a constraint network in which the underlying graph G is a cycle. A fundamental cycle, F(p) in C is a cycle without cords in the product graph which "passes through" a given vertex in the underlying graph exactly p times. That is, the labeling of the equivalent labeling induced subnetwork has $|\Lambda_i| = p$ for all i. A fundamental anticycle, F(p) is a cycle without cords in the complement graph which passes though each vertex in the underlying graph exactly p times.

3.2. Discrete Relaxation

Given an initial (possibly inconsistent) labeling $L = (\Lambda_1, \Lambda_2, \ldots, \Lambda_n)$ there exists an iterative, parallel, and decentralized algorithm for finding the largest (possibly ambiguous) consistent labeling $L^C = (\Lambda_1^C, \Lambda_2^C, \cdots, \Lambda_n^C)$ contained in L (note[2]). The basic strategy involved in this algorithm is to iteratively remove those labels from the label set of a given vertex which are not consistent with the labeling on the neighborhood of that vertex. Removing a label from a label set at a given iteration may cause the removal of a label from the label set of an adjacent vertex at the next iteration, so that the effect is seen as propagating through the extent of the network (hence the term "constraint propagation" which has often been applied to algorithms that resolve inconsistency in this way).

The algorithm proceeds in an lock step manner and changes the composition of the label set at each iteration. Denote the current labeling at an iteration t by $L^t = (\Lambda_1^t, \Lambda_2^t, \cdots, \Lambda_n^t)$. Define:

$$c_{ij}^t = \begin{cases} 1 & \text{if } \lambda_j \in \Lambda_i^t \\ 0 & \text{otherwise} \end{cases}$$

to be the predicate, or indicator function for label λ_j being a member of label set Λ_i^t at iteration t. Also, define:

$$r_{ii'}(\lambda_j,\lambda_{j'}) = \begin{cases} 1 & \text{if } (\lambda_j,\lambda_{j'}) \in R_{ii'} \\ 0 & \text{otherwise} \end{cases}$$

to be the predicate, or indicator function for the pair $(\lambda_j,\lambda_{j'}) \in R_{ii'}$. Then the iterative equations which describe the discrete relaxation labeling process are

---

[2]Because the union of two consistent labelings in a consistent labeling, the largest consistent labeling $L_C \subseteq L$ is unique

given as:

$$c_{ij}^{t+1} = c_{ij}^{t} \wedge \{ \bigwedge_{i' \in N(i)} [\bigvee_{j' \in \Lambda} r_{ii'}(\lambda_j, \lambda_{j'}) \wedge c_{i'j'}^{t}] \} \qquad 3.1$$

In words: a label λ_j will be removed from the label set Λ_i^t for vertex v_i at a given iteration t if there exists a neighboring vertex $v_{i'}$ such that, for no $\lambda_{j'} \in \Lambda_{i'}^t$ is the labeling pair $(\lambda_j, \lambda_{j'})$ consistent.

Clearly, $L^{t+1} \subseteq L^t$. Also, if $L^t = L^{t+1}$ for some iteration t then $L^t = L^{t+r}$ for all $r \geq 1$. The smallest $t \geq 0$ at which then $L^t = L^{t+1}$ will be called the *settling time* for the network, for the given input L. Since labels can only be removed from the label set, the algorithm is guaranteed to converge. Obviously, $|\Lambda| \times |V| = mn$ is an upper bound on the settling time of a network for any initial labeling. In fact, it appears that better bounds can be obtained, both in the general case, and in problems with specific structure. This issue will be discussed further in chapter IV.

Example 3.3: Applying the discrete relaxation operator to the constraint network of example 3.1 with the labeling shown in figure 3.2a results in the labeling shown in figure 3.2d. The iterations of the discrete relaxation operations which lead to this labeling are shown in figures 3.2a through 3.2d.

Example 3.4: Consider the constraint network illustrated in figures 3.3a through 3.3d. The discrete relaxation operation applied to the constraint network with the initial labeling shown in figure 3.3b results in the labeling of figure 3.3c. This labeling is consistent but not unambiguous. Furthermore, if a label were to be removed from any label set the result would be the null labeling. The labeling induced subnetwork of figure 3.3c, which is shown in figure 3.3d is a fundamental cycle.

Although the discrete relaxation labeling process does not guarantee an unambiguous consistent labeling, it can however be used as a means to reduce the search space in an implicit enumeration algorithm for the graph labeling problem as is discussed below.

3.3. The Graph Labeling Problem

3.3.1. Definition

Let $C(G,\Lambda,R)$ be a constraint network, where G is a graph with n vertices, and Λ is a set with m labels. Associate with each label, λ_j on each vertex v_i a real cost $c_{ij} \in R$. A *graph labeling problem* (GLP) defined on C is given as:

$$\textbf{maximize:} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij}\, x_{ij}, \tag{3.2}$$

$$\textbf{subject to:} \quad \sum_{j=1}^{m} x_{ij} = 1, \quad i=1,\ldots,n, \tag{3.3}$$

$$x_{ij} + x_{i'j'} \leq 1 \tag{3.4}$$

for every pair of *inconsistent* labels $(\lambda_j,\lambda_{j'})$ on adjacent vertices v_i and $v_{i'}$,

$$x_{ij} \in \{0,1\} \tag{3.5}$$

Constraint 3.3 serves to guarantee that exactly one label is chosen for each vertex in the graph. Constraint 3.4 specifies that a label chosen for a given vertex will be consistent with the labels chosen for adjacent vertices. Finally, constraint 3.5 requires that the x_{ij} be maintained as 0-1 (decision) variables. The resulting problem is, therefore, one of selecting an unambiguous consistent labeling such that the sum of the initial labeling values is maximal.

Let the cost vector be denoted as $c^T = (c_{11}, c_{12}, \ldots, c_{nm})$. $x^T = (x_{11}, x_{12}, \ldots, x_{nm})$ will denote the vector of 0-1 variables. In matrix-vector form the graph labeling problem can be expressed as:

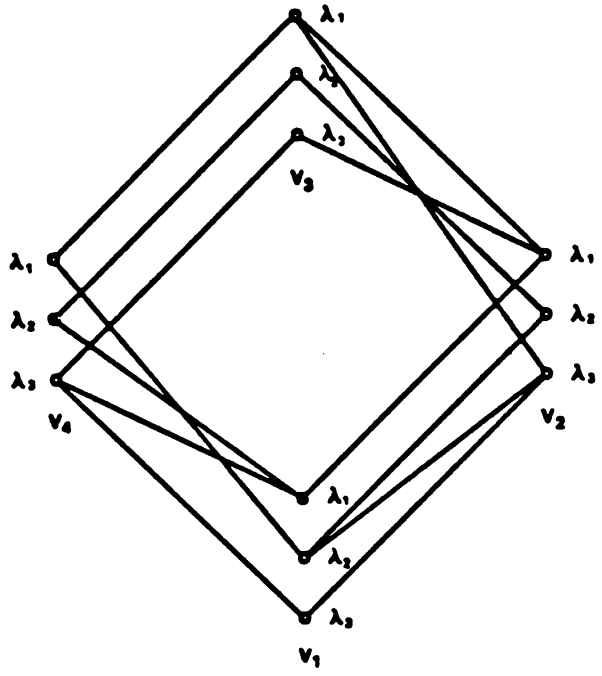


Figure 3.3a: Constraint Network for Example 3.4

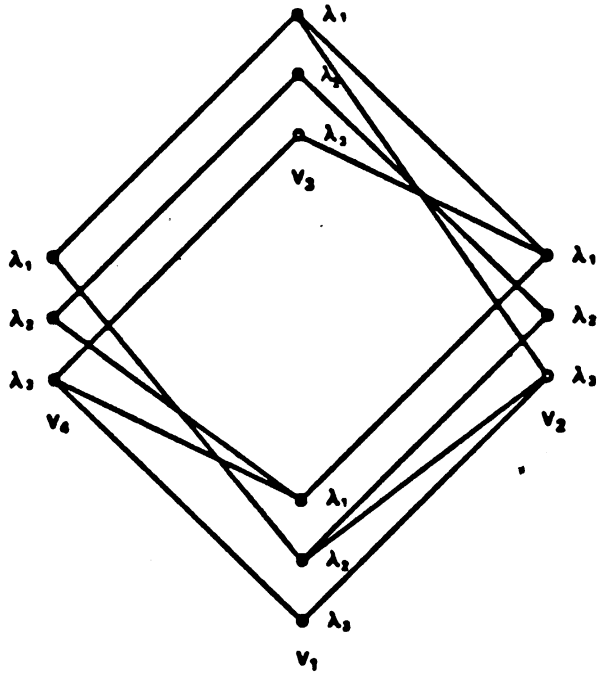


Figure 3.3b: Constraint Network for Example 3.4: Initial Labeling

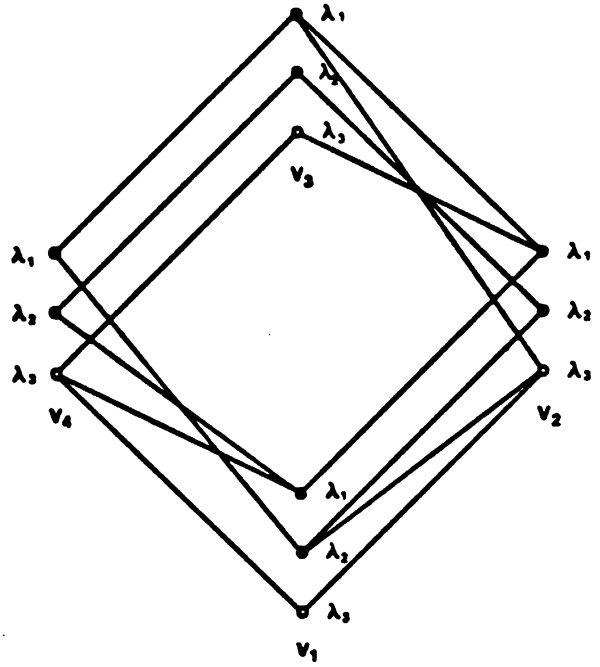


Figure 3.3c: Constraint Network for Example 3.4: Final Labeling

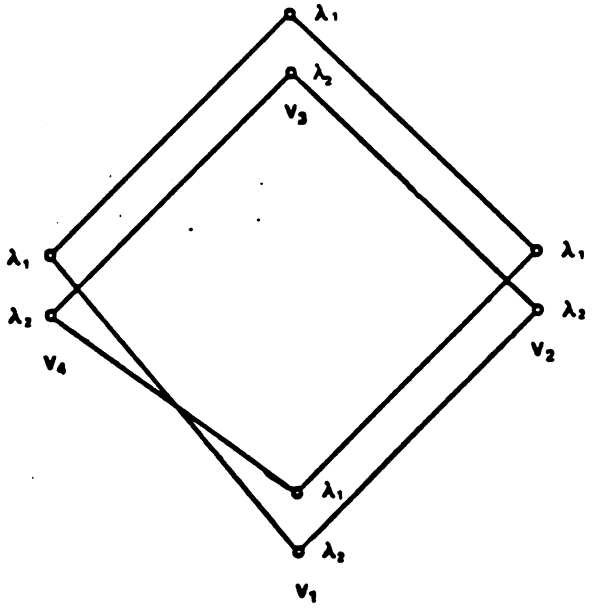


Figure 3.3d: Fundamental Cycle Derived From Final Labeling

maximize: $c^T x$

subject to: $A_1 x = e$

$A_2 x \leq e,$

with $x_{ij} \in \{0,1\}$, and e is a compatibly dimensioned vector of all ones: $e = (1, 1, \ldots, 1)$. The matrix A_1 has the *block diagonal* form:

$$A_1 = \begin{bmatrix} 1\,1\,1 \ldots 1 & & \\ & 1\,1\,1 \ldots 1 & \\ & & 1\,1\,1 \ldots 1 \end{bmatrix} \qquad 3.6$$

where all blank entries are assumed to be equal to zero.

A_2 is also a block structured (usually sparse) matrix with blocks at the intersection of columns corresponding to all the labels for a given vertex and rows corresponding to constraints between a given vertex and one or more of its neighbors. It is the incidence matrix of the complement graph for the underlying constraint network.

3.3.2. Example Problem

An example is given to illustrate the definitions given above. The underlying constraint network for this problem is given in complement graph of figure 3.4. The initial values associated which each label is shown in parentheses next to the respective label. The solution to this problem is to select label λ_2 on vertex v_1 and λ_2 on vertex v_2. The optimal objective value is 17. The primal statement of the problem is:

Maximize: $12x_{11} + 9x_{12} + 3x_{13} + 10x_{21} + 8x_{22} + 2x_{23}$

Subject to: $x_{11} + x_{12} + x_{13} = 1$

$$x_{21} + x_{22} + x_{23} = 1$$

$$x_{11} + x_{21} \leq 1$$

$$x_{11} + x_{22} \leq 1$$

$$x_{12} + x_{21} \leq 1$$

$$x_{ij} \in \{0,1\},\ i=1,2,\ j=1,2,3$$

3.3.3. PMI Constraints

The constraints derived from the constraint network for a given graph labeling problem imply other constraints which cannot be derived from an algebraic combination of, but are nontheless implied by the original constraints. A class of such constraints referred to as *pairwise maximally inconsistent (PMI)* constraints are discussed further in the following chapters. PMI constraints for a given pair

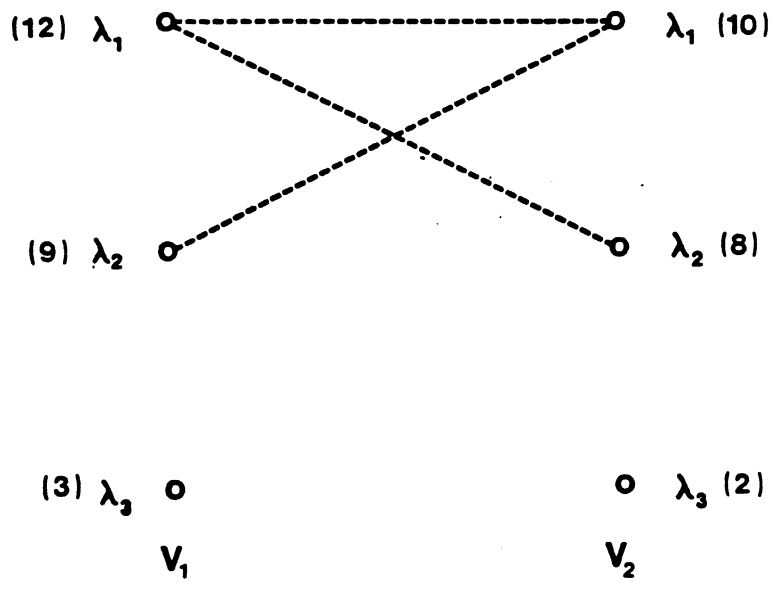


Figure 3.4: Complement graph for example problem 3.3.2.

of vertices are derived from the cliques of the augmented complement graph corresponding to pairs of adjacent vertices. For example, in the problem of section 3.3.2 there are four cliques in the augmented complement graph. One contains the labels λ_{11}, λ_{12}, and λ_{21}. Another contains the labels λ_{11}, λ_{21}, and λ_{22}. The other two correspond to all the labels on a given vertex, that is, λ_{i1}, λ_{i2}, and λ_{i3} for i=1,2. Note that at most one label can be chosen from any clique in a consistent labeling. The problem expressed in terms of PMI constraints is given as:

$$\text{maximize:} \quad 12x_{11} + 9x_{12} + 3x_{13} + 10x_{21} + 8x_{22} + 2x_{23}$$

$$\text{subject to:} \quad x_{11} + x_{12} + x_{13} = 1$$

$$x_{21} + x_{22} + x_{23} = 1$$

$$x_{11} + x_{12} + x_{21} \leq 1$$

$$x_{11} + x_{21} + x_{22} \leq 1$$

$$x_{ij} \in \{0,1\}, \; i=1,2, \; j=1,2,3$$

Certainly, other constraints can be formed in a similar manner for the general graph labeling problem. There appears, however, to be a computational advantage in formulating the problem in terms of PMI constraints. Furthermore, certain characteristics of special case graph labeling problems can be derived when the problem is formulated in this manner as will be discussed in the following chapters.

3.3.4. Related Problems: Covering, Partitioning, and Packing

Three closely related and well established classes of 0-1 linear integer programs are given as follows:

[SC] Set covering [BaP74,Sal75,Bar81,Etc77,Sly82]:

$$\text{minimize:} \quad c^T x$$

subject to: $A x \geq e$, $x_i \in \{0,1\}$.

[SPP] Set partioning [BaP74,Sal75,Mar74,NTN74]:

minimize: $c^T x$

subject to: $A x = e$, $x_i \in \{0,1\}$.

[SP] Set packing [BaP74,Sal75,Pad73]:

maximize: $c^T x$

subject to: $A x \leq e$, $x_i \in \{0,1\}$.

A is a 0-1 matrix (with no other structure assumed), $c^T = (c_1, c_2, \ldots, c_n)$ is a cost vector and, $x^T = (x_1, x_2, \ldots, x_n)$ is a 0-1 vector.[3] The reference to the "set" in these problem definitions is derived by interpreting a column, a_j of the $m \times n$ matrix A as being the indicator vector for a subset S_j of $S = \{ 1, 2, \ldots, m$. If a cost c_j is associated with each S_j, the problems become those of finding an optimal cost covering, partioning, and packing of these subsets. Set packing problems are often expressed as equivalent vertex packing problems [NeT74,NeT75,PiQ77] defined on the derived graph of the matrix A.[4]

By adding the appropriate slack variables to the inequality constraints, GLP is easily seen to be a special case of SPP. In this case the constraint matrix has the form;

---

[3] In discussing the graph labeling problem, double indices (e.g. x_{ij}, c_{ij}) will be used to emphasize the structure of the problem. In other other cases, variables and constants will be singly subscripted.

[4] Given a 0-1 matrix A, the derived graph is the graph G wherein each column, a_j, of A corresponds to a vertex v_j and there is an edge between two vertices v_i and v_j if and only if $a_i \cdot a_j^T > 0$.

$$A_3 = \begin{bmatrix} A_1 & : & 0 \\ \dots & : & \dots \\ A_2 & : & I \end{bmatrix} \qquad 3.7$$

There are also well known transformations from SPP to SC and SP [BaP74]. To transform GLP directly to SP, write GLP as:

[P1] minimize: $c^T x + \vartheta e^T y$

subject to: $A_1 x + y = e$

$A_2 x \leq e$

$x_{ij} \in \{0,1\}$.

where $y^T = (y_1, y_2, \dots, y_n)$, and $y_i \geq 0$. Note that any solution to [P1] in which $y = 0$ is also a solution to the corresponding graph labeling problem. Pricing out y as: $y = e - A_1 x$, results in the problem:

[P2] maximize: $-c^T x + \vartheta e^T (A_1 x - e)$

$= (\vartheta e^T A_1 - c^T) - \vartheta m$

subject to: $A x = \begin{bmatrix} A_1 \\ .... \\ A_2 \end{bmatrix} x \leq e, \; x_{ij} \in \{0,1\}$.

If ϑ is sufficiently large, say $\vartheta \geq \sum c_{ij}$ then $y = 0$ so long as the original problem is feasible, so that a solution to [P2] is guaranteed to be a solution to the original problem. Given the structure of A_1, this problem can be expressed as:

[P3] maximize: $(\bar{\vartheta} + c)^T x$

subject to: $A x \leq e, \; x_{ij} \in \{0,1\}$.

where, $\bar{\vartheta} = (\vartheta, \vartheta, \ldots, \vartheta)^T$.

In order to transform GLP into SC, first it is necessary to add slack variables to the inequality constraints thus converting it to SPP as in equation 3.7. Let A_3 be the resulting constraint matrix. Then, in a transformation similar to that given above [BaP74] GLP is expressed as:

[P4] minimize: $c^T x + \vartheta e^T y$

subject to: $A_3 x - y = e$

where $x_{ij} \in \{0,1\}$ and $y \geq 0$. y is priced out as: $y = A_3 x - e$ resulting in the equivalent problem:

[P5] minimize: $(c^T + \vartheta e^T A_3) x$

subject to: $A_3 x \geq e$

Perhaps the most important relationship between the current work and the established classes of 0-1 integer programs is found in the following transformation from SPP to GLP. Let A be the constraint matrix of a set partitioning problem. The columns and rows of A are permuted to bring it into the *block form* [Mar74]:

$$A_4 = \begin{bmatrix} A_5 \\ .... \\ A_6 \end{bmatrix} \qquad 3.8$$

where A_5 the block form

$$A_5 = \begin{bmatrix} 1\,1\,1 \ldots 1 & & \\ & 1\,1\,1 \ldots 1 & \\ & & 1\,1\,1 \ldots 1 \end{bmatrix}.$$

A *block* is set of columns which correspond to a row of 1's in the matrix A_5. Matrix A_5 now has the form as A_1 in equation 3.6, except that the number of 1's in a given row of matrix A_4 may vary, whereas the number of 1's in a given row of A_1 is equal to m (the number of labels in Λ). Assume that the maximum number of columns in a given block of A_5 is equal to m. Then for each block with $m' < m$ columns, $m - m'$ columns with a 1 only in the appropriate row and the rest of the components set equal to 0, can be added. The cost vector component associated with the added columns are set to an arbitrarily small number. It is assumed that this preconditioning of the constraint matrix has been performed resulting in the constraint matrix

$$A'_4 = \begin{bmatrix} A'_5 \\ \cdots \\ A'_2 \end{bmatrix}$$

and that the resulting problem is expressed as:

[P6] maximize:[5] $c^T x$

subject to: $A'_4 x = e$.

Assume that A'_5 has n rows or blocks. Each block will represent the labels associated with a particular vertex in the equivalent graph labeling problem. The graph derived from the matrix A'_6 is the complement graph for the equivalent graph labeling problem. Hence, there is an edge between two "vertices" (blocks) in the underlying graph, if and only if there is a pair of inconsistent labels on those vertices, as defined by the complement graph. The complement graph then defines a constraint matrix A''_6. Note that A''_6 and A'_6 need not necessarily be the same. The transformed, but equivalent set partitioning problem is given as:

---

[5] Note that the set partioning problem [P6] is expressed here as a maximization problem.

[P7] maximize: $c^T x$

subject to: $A'_5 x = e.$

$A''_6 x = e.$

Assume that A'_5 has n rows or blocks, so that there would be n vertices in the equivalent graph labeling problem. Assume A''_6 has m_6 rows. Problem [P7] would be a graph labeling problem on n vertices except that the second set of constraints are equality constraints. Then using the same approach as in the transformations described above, [P7] can be expressed as:

[P8] maximize: $\hat{c}^T x$

subject to: $A'_5 x = e.$

$A''_6 x \leq e.$

where $\hat{c} = c + \bar{\vartheta} m_6 - A_6$, and $\bar{\vartheta} = (\vartheta, \vartheta, \cdots, \vartheta)$ with $\vartheta > \sum c_{ij}$.

It would be difficult to overstate the implications of these transformations, in particular, the transformations from SPP to GLP and vica-versa. As stated in [PaB74]:

> Among all special structures in integer programming, there are three which have the most wide-spread applications: set partitioning, set covering, and the traveling salesman problem; and if we were to rank the three, set partitioning would probably be number one.

3.4. The Complexity of the Graph Labeling Problem

Consider a variation of the graph labeling problem stated as follows: Given a constraint network, C, assume the costs c_{ij} take on only values of 0 or 1. Then the problem is to find a (any) unambiguous consistent labeling such that the sum of the initial strength measures is equal to n (the number of vertices in the underlying graph). This problem is equivalent to determining if a labeling induced

subnetwork contains a globally consistent unambiguous labeling. As has been stated in the literature several times, this problem, often referred to as the *discrete* graph labeling problem or the consistent labeling problem, is NP-complete [HDR78,HaS79].

It is not difficult to find a polynomial time reduction from the discrete graph labeling problem to nm satisfiablility (n is the number of vertices in the graph, and m is the number of elements in the label set). A more common argument is to reduce it directly to the K-coloring problem, which is known to be NP-complete [GaJ78,Gol80]. Given an arbitrary graph, G, the problem is to determine whether or not that graph can be properly colored with K colors. Equivalently, the underlying graph labeling problem is defined on a constraint network with the underlying graph G. The labels correspond to colors, and each label is inconsistent only with the label representing the same color on every adjacent vertex. Thus, if there exists a globally consistent labeling, the graph can be colored with K or fewer colors.

The extension to the continuous graph labeling problem is direct. It remains to assign initial costs to the labels in such a way as to guarantee that the globally consistent labeling with the maximal labeling value, if it exists, is unique. Otherwise, it could be argued that the difficulty in the discrete graph labeling problem is in the fact that every globally consistent unambiguous labeling has the same value. To do this, assign a unique index, i, from the set {1, 2,..., n} to each vertex of the graph. Furthermore, assign a unique number, c, from the set {1, 2,..., K} to each color. Finally, assign an initial cost of $c\, i^K$ to each color, c, on each vertex v_i. It is clear that with these costs, every unambiguous labeling (both consistent and inconsistent) has a unique total cost. It can be concluded that a polynomial time solution to this graph labeling problem implies therefore, a polynomial time solution to the general K coloring problem.

It should be noted that this argument points to a fundamental difficulty with the solutions to graph labeling, which holds as well for the set partioning problem: determining the existence of *any* feasible solution is NP-complete, in the general case. This is not the case, however, for the set packing and set covering problems although, the problems of finding a maximal packing or minimal covering is still NP-complete [BaP74]. Special cases in which the graph problem is not can be solved efficiently[6] will be discussed in chapter IV.

3.5. The Basic Approach to the Graph Labeling Problem

In the previous chapters and in the previous sections of this chapter the nature of the graph labeling problem was discussed. The origin of the proposed definition, its relationship to the application as well as other problems in combinitorial optimization have been covered. In this section the basic approach to the solution of the graph labeling problem will be outlined.

Integer programming approaches fall into three broad catagories: implicit enumeration (branch and bound), cutting plane, and group theoretic. Group theoretic approaches can not in general be applied to 0-1 integer programs [Bal73] although a variation of the dynamic programming methods often used in conjunction with the group theoretic approaches is discussed in chapter V. Furthermore, the cutting plane techniques do not appear to be suited for the types of hardware implementation which motivates the design of algorithms which are discussed below. Thus the basic strategy involved will be based on some form of implicit enumeration.

An enumeration scheme involves the orderly generation of partial solutions, where a partial solution, or candidate subproblem is derived from the original problem by fixing or restricting the values of some subset of the variables. The

---

[6] That is with the simplex algorithm.

generation is performed in a branching operation which generates subproblems from the candidate problem by further restricting or fixing the values of some of the variables. In most cases, the set of feasible solutions need only be implicitly enumerated, since the derivation of bounds on the best and worst case completions can be used to remove (fathom) subproblems from further consideration. In particular, in any stage of the algorithm, it is assumed that there exists (1) a current best possible solution, with an objective value of z^*, and (2) a set P of candidate subproblems. Then the derivation of these bounds can be used to greatly reduce the generation of partial solution since if it can be determined that the best possible complete solution derived from an element of P is no better than the best known solution, then that subproblem can removed from P.

The branch and bound algorithms which have been the most effective in solving set partitioning problems involve preconditioning the constraint matrix by first ordering the columns into a staircase, or block form in which all columns with the first non-zero entry in a given row are grouped together as discussed above [Pie68,Mar74]. This is done so as to allow for easy seperation of the set of feasible solutions, since exactly one column must be chosen from each block. Note that in the case of GLP the A_1 matrix in addition with a square identity matrix representing the slack variables gives the constraint matrix this form to start with (refer to equation 3.7). However, in this case, no particular advantage is gained because the resulting seperation has given by edict. That is, the problem is designed to in fact allow for those solutions which specify exactly one label per vertex.

For the case of the graph labeling problem, the seperation of the set of feasible solutions will be done in a more direct manner. In this case, each partial solution, or candidate problem will be referred to as a *restriction*. Formally, a restriction, R is a 3-tuple, R = <S0,S1,Sf>, where S0 is a set of variables fixed

at 0, S1 is a set of variables fixed at 1, and Sf is the set remaining variables, for which values have not been specified. The variables for which values have been assigned, in a given restriction R must satisfy the constraints. In particular, for the graph labeling problem, it will be assumed that if a given variable corresponding label, say λ_j on a vertex, say v_i is set to 1 (which is equivalent to selecting that label at that vertex), then the variables corresponding to the rest of the labels at that vertex must be set to zero. Thus in this case, an equivalent formulation of a restriction is that of a (possibly ambiguous) labeling $L = \{\Lambda_1, \Lambda_2, \cdots, \Lambda_n\}$ where $x_{ij} = 0$ or $x_{ij} \in S0$ if λ_j , $x_{ij} = 1$ or $x_{ij} \in S1$ if $\lambda_j \in \Lambda_i$ and $|\Lambda_i| = 1$, and x_{ij} is free, or $x_{ij} \in Sf$ if if $\lambda_j \in \Lambda_i$ and $|\Lambda_i| > 1$.

It will be assumed that the basic model for the implicit enumeration scheme which will be used here has the following components.

[1] A *starting point function*, which selects an initial best objective value, z^*.

[2] A *restriction choice rule*, which chooses a restriction from a set P of current restrictions to examine next.

[3] A *lower bound function*, which determines a feasible completion of a given restriction and the corresponding cost.

[4] An *upper bound function* which finds a upper bound on the cost of completing the current restriction.

[5] A *branching variable choice rule*, which determines how to branch the current restriction (i.e. which variable to fix next).

[6] A feasibility test, which determines whether or not there is a feasible completion of the current restriction.

For the application of the graph labeling model to the edge linking problem, both the starting point function and the lower bound functions will be derived from a "rounding" process discussed further in chapter 5. The upper bound is usually

determined by some form of relaxation of the candidate problem. The most common forms of relaxations are:

[1] Constraint subset relaxation, wherein some of the constraints which affect the elements of Sf are relaxed, thus making it relatively easy to find a completion of the current restriction,

[2] Linear programming relaxation, wherein the integrality constraints of the elements of Sf are relaxed,

[3] LaGrangian relaxation, wherein the constraints affecting the elements of Sf are relaxed via LaGrange multipliers.

Of these three, we will consider here only the constraint subset relaxations and the Lagrangian relaxation. It will turn out that compution of bounds based on these two relaxation strategies can be performed very rapidly based on the assumption that they are implemented in special purpose hardware. The branching rule will be based on heuristics discussed in chapter 6, and the feasibility tests will be implemented with the discrete relaxation operation as discussed in the following chapter.

The basic strategy in the implicit enumeration step which will be considered here is given as:

[1] Use the restriction choice rule to choose a restriction, R, from the current set, P, of restrictions,

[2] Use the feasibility test to determine whether or not a completion of the current restriction exists. If not, remove R from P.

[3] Use the upper bound function to find U(R), If $U(R) \leq z^*$, remove R from P.

[4] Use the lower bound function to find L(R), as well as a feasible solution, x' at which the lower bound is obtained.

[7] Otherwise, use the branching variable choice rule to select a $j \in Sf$. Set $R0 = \langle S0 \cup \{j\}, S1, Sf - \{j\} \rangle$, and $R1 = \langle S0, S1 \cup \{j\}, Sf - \{j\} \rangle$. Add R0 and R1 to P.

The focus of the work in the following chapters is on issues related to the various components of this basic approach, as well as its applicability to the original problem, that of edge linking. Although the enumeration strategy itself cannot be implemented in parallel, many of the fundemental components can, and it is in the hope that by doing so the rapid derivation of line drawings will be possible, that these issues are being pursued.

CHAPTER IV

THEORY

The following chapter is divided into two subsections. In the first, the convergence properties of the discrete relaxation processes are discussed. The second subsection covers issues related to the structure of the graph labeling problem itself.

4.1. Discrete relaxation

One consistent feature of the algorithms which will be examined and developed further in the following chapter is that they will proceed to a point which results in a labeling, L, which is ambiguous. As will be discussed, the performance of these algorithms could be greatly enhanced if it were possible very rapidly either to derive from L the maximal consistent labeling, $L^C \subseteq L$, contained therein, or to determine whether or not there is in fact an unambiguous consistent labeling contained in L. For example, in an implicit enumeration strategy, a restriction, R = <S0,S1,Sf> is equivalent to a labeling, $L = (\Lambda_1, \Lambda_2, \cdots, \Lambda_n)$ in which $\Lambda_i = \{\lambda_j\}$, if the vertex v_i is fixed in R, where λ_j is the unique label at vertex v_i such that $x_{ij} = 1$, and $\Lambda_i = \Lambda$ otherwise (i.e. $x_{ij} \in Sf$ for all j). If there is not an unambiguous consistent labeling within L, then R is infeasible and can be pruned. If there is, we could immediately add to the set S0 the variables corresponding to labels not in L^C.

For an arbitrary labeling L, the derivation of L^C can be accomplished through the use of the discrete relaxation operator alone. Determining whether or not there exists an unambiguous consistent labeling within L can be accomplished by repeatedly using discrete relaxation in conjunction with an enumerative scheme (e.g. with a stack). Even though it can be easily implemented in hardware, the speed of convergence of the discrete relaxation process may become an issue if a large number of such operations are required. Obviously, the settling time bound of $n \times m$ for a problem defined on a constraint network with n vertices and m labels, would be unacceptable. For example, a graph labeling problem defined on a 128×128 raster with $|\Lambda| = 16$ might require (in the worst case) about a quarter million iterations or propagation delays per test. Even if implemented in combinatorial logic, the convergence time, under these conditions could be unacceptable. It appears, however, that the both the worst case, and the expected settling times can be assumed to be much more tightly constrained. This assertion is based both on results generated from simulations and from bounds derived for restricted forms of the graph labeling problem.

It should be noted that in general, one need not wait until the network has settled. In particular, if at some point, there exists a vertex v_i such that the corresponding label set is empty, $|\Lambda_i| = \phi$, then it can immediately be determined that the fixed point of the process will be the null labeling. In the following discussion, the first iteration at which this condition is recognized will be referred to as the null time of the process. If the largest consistent labeling contained within the initial labeling is not null, the null time is infinite. Finally, the detection time of the process is defined to be the minimum of the settling time and the null time. It is clear that it is the detection time which is important in evaluating the performance of the discrete relaxation process.

4.1.1. Supporting sets of labels

The setting, null, and detection times will be referred to collectively as the critical times for the network. Worst case bounds and expected values for the critical times are difficult to derive for general constraint networks. However, an analysis is presented which will hopefully be a contribution to further work. This analysis is based on the concept of a *supporting set* of labels, which is defined as follows: A label λ_j on a given vertex v_i is said to *support* another label λ_j' on a vertex v_i' (v_i and v_i' may be the same vertices) if there exists a consistent labeling $L^{C'}$ and a maximal consistent sublabeling $L^C \subseteq L^{C'}$ such that both λ_j' and λ_j are in $L^{C'} - L^C$. The importance of this definition is in that starting at the consistent labeling L^C and removing the label λ_j from the label set Λ_i at iteration t will eventually "cause" the removal of the label λ_j' from the label set Λ_i' at some later iteration $t' > t$. This follows directly from the fact that the discrete relaxation operator applied to the labeling $L^{C'} - \{\lambda_{ij}\}$ will result in the labeling L^C, which does not contain $\lambda_{i'j'}$.

Since two labels may either support or not support each other, this condition is a relation defined on the set of all possible labels on every vertex of the graph. Obviously, the supporting relation is symmetric and reflexive, but it is not transitive, that is, it is a compatibility relation. It is important to note that the supporting relation specifies a sufficient condition for the situation where the removal of a label at a given vertex will cause the removal of another label at some later time *only under certain conditions*. In particular, the entire initial labeling must be taken into consideration.

Example 4.1: It may appear to be a contradiction that the supporting relation is not transitive, since if the removal of a label λ_j will cause the removal of a label λ_j', and the removal of the label λ_j' will cause the removal of another label $\lambda_{j''}$, then the removal of the label λ_j should cause the removal of the label $\lambda_{j''}$. This

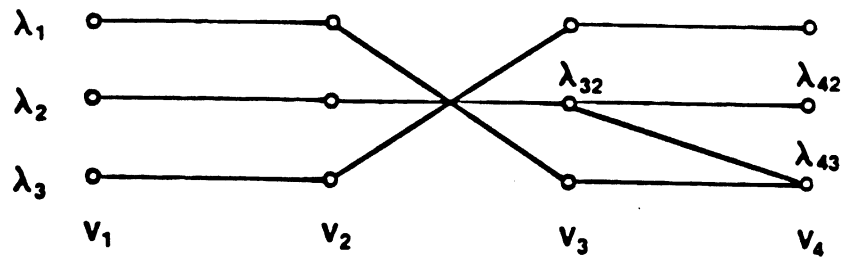


Figure 4.1a: Constraint Network for Example 4.1

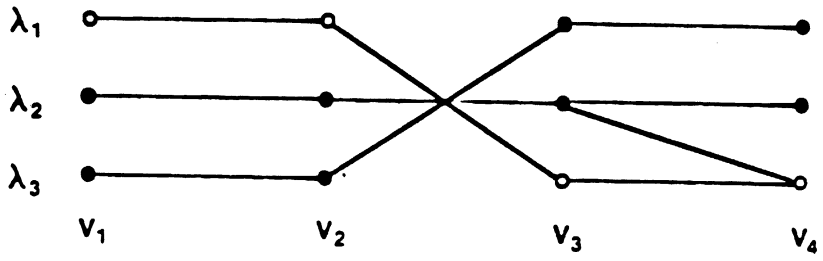


Figure 4.1b: Constraint Network for Example 4.1: Initial Labeling

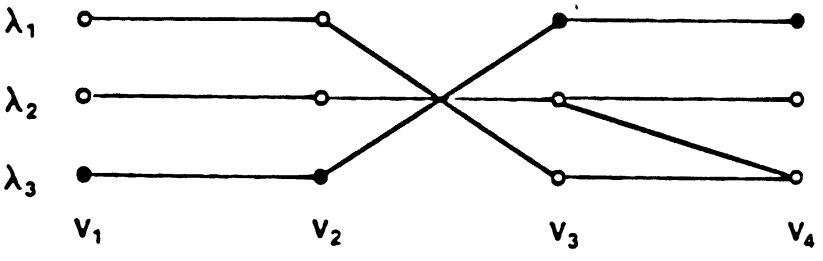


Figure 4.1c: Constraint Network for Example 4.1: Final Labeling

results, however, from the fact that the removal of λ_j is only a sufficient condition for the removal of λ_j', given the proper initial labeling. Consider a network the product graph for which is shown in figure 4.1a. Label λ_{42} supports the label λ_{32} since, given the consistent labeling L^C shown in figure 4.1b, the removal of the label λ_{42} will result in the maximal consistent sublabeling shown in figure 4.1c. Likewise, λ_{43} supports label λ_{32}. However, it is easily verified that there is no difference $L^{C'} - L^C$ between a consistent labeling $L^{C'}$ and a maximal consistent sublabeling L^C which contains both λ_{42} and λ_{43}. Furthermore, removal of the label λ_{42} from any consistent labeling which contains both λ_{43} and λ_{42} will never result in a labeling which does not contain λ_{43}.

4.1.2. Observations for the general case

The discrete relaxation operator maps a given labeling, L into the maximum consistent labeling L^C such that $L^C \subseteq L$. Furthermore, in order for the process to remain active, at least one label must be removed from some label set Λ_i^t at each iteration t. Thus, it appears worthwhile to pursue an analysis of the maximum difference $|L| - |L^C|$ between a given labeling L and the largest consistent labeling L^C contained in L. In particular, we shall consider the differences of the form $|L^{C'}| - |L^C| = |L^{C'} - L^C|$ between any consistent labeling $L^{C'}$ and a maximal consistent labeling L^C properly contained in $L^{C'}$.

One means in which a bound on the worst case settling time could be determined would be to consider every labeling $L^{C'}$ consistent with respect to the given constraint network, and then every maximal sublabeling, L^C contained in $L^{C'}$. Then for every such pair, experimentally determine the worst case settling time by applying the discrete relaxation operator to the labeling $L^{C'} - \{\lambda_{ij}\}$ over all $\lambda_{ij} \in L^{C'} - L^C$. Obviously, this bound would be tight.

Given a particular class of constraint networks one might be able to derive these bounds by considering the special structures involved. Looser, but perhaps easier to determine bounds, might be derived as follows: Let L_d denote the set of all labelings resulting from differences of the form $L^C - L^{C'}$. One obvious bound on the worst case settling time is given as:

$$\max \{L \in L_d : |L| - 1\}$$

Note that this bound may in fact be larger than the one suggested in the previous paragraph.

One further means of determining settling time bounds is given as follows: If $L^{C'}$ is a consistent labeling, and L^C is a maximal consistent labeling contained in $L^{C'}$, then it follows that $L = L^{C'} - L^C$ is either a minimal consistent labeling $L \in L_m$, or it is an inconsistent labeling L. Were this not the case, L could be expressed as the union of a non-empty consistent labeling L', and a non-empty labeling L'' such that $L'' - L' \neq \phi$. Then $L^C \subset L^{C'} \cup L' \subset L^{C'}$ and $L^C \cup L'$ is consistent, so that L^C is not a *maximal* consistent sublabeling. If L is inconsistent, then it is contained in a minimal consistent labeling. Therefore, it follows that the settling time for a given constraint network is always bounded by:

$$\max \{L \in L_m : |L| - 1\}$$

Example 4.2: Consider the case of a constraint network $C(G,\Lambda,R)$ where G is the cycle C_4, and a labeling induced subnetwork which is a fundamental cycle (refer to figure 4.2). This labeling is consistent, and the largest consistent sublabeling contained in it is the null labeling. Hence, every label on this cycle supports every other label. This is true for any fundamental cycle contained in a constraint network. Note that the removal of any label from this labeling will eventually result in the removal of all labels from the labeling, if the discrete relaxation operator is applied. The worst case settling time is easily seen to be $\lfloor \frac{|L|}{2} - 1 \rfloor$,

where $|L|$ is the number of elements in the labeling. Note that this bound is lower than any of the bounds (except those derived from direct experimentation) discussed above.

Example 4.3: The labeling induced subnetwork shown in figure 4.3 is also defined on a network where the underlying graph is the cycle C_4. The subnetwork results from the union of two fundamental cycles, whose intersection is not empty. The worst case settling time, which occurs when the label λ_{13}, is removed is easily seen to be $n-2$, where n is the number of labels in the second fundamental cycle. Although each label on a fundamental cycle supports every other label one that fundamental cycle, a label λ_1, for example, on the fundamental cycle A, does not support a label λ_2 on fundamental cycle B, unless those labels happen to be contained in the intersection of the two cycles.

4.1.3. The case in which the underlying graph does not contain cycles

These two examples show how the effect of removing a label at a given time step can eventually cause the removal of another label, if the first label supports the second. Before continuing to discuss the general case, we consider a class of constraint networks for which strong statements about the critical time bounds can be made.

Lemma 4.1: In the case where the underlying graph is a tree (or forest), the network does not contain any fundamental cycles, and the set L_m of minimal consistent labelings is exactly the set L_{uc} of unambiguous consistent labelings. Furthermore, the set of all consistent labelings is a sup-semilattice, with the join operation being the union of labelings and with the atoms being the elements of L_{uc}.

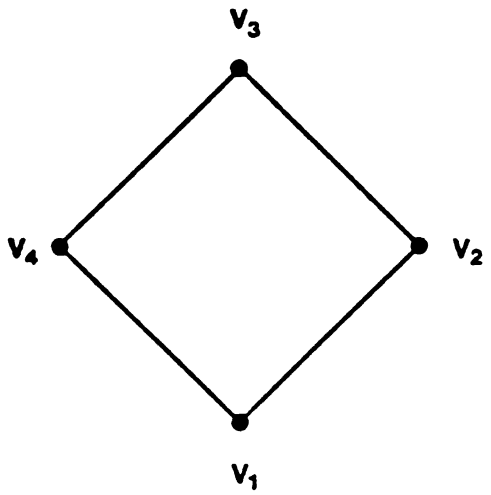


Figure 4.2a: Underlying graph for the constraint network of example 4.2.

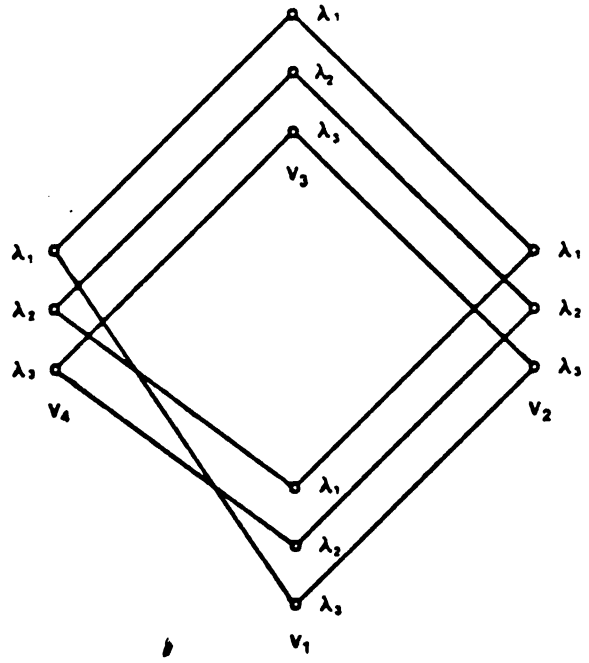


Figure 4.2b: Product graph for the constraint network of example 4.2.

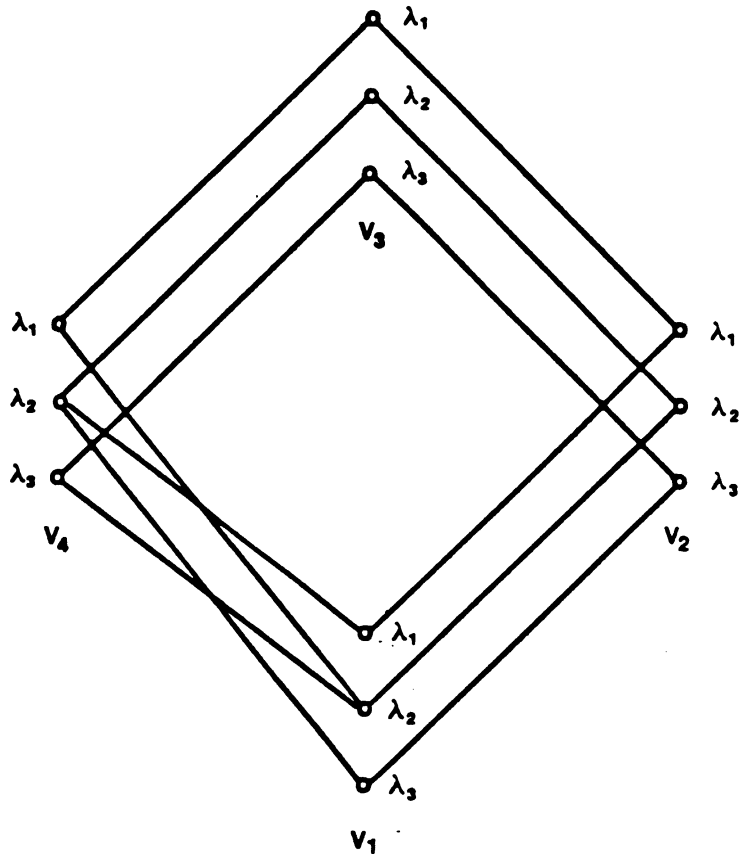


Figure 4.3: The constraint network for example , which is the union of two fundamental cycles, F(2), A and B, where:

$A = \{\lambda_{11}, \lambda_{21}, \lambda_{31}, \lambda_{41}, \lambda_{12}, \lambda_{22}, \lambda_{32}, \lambda_{42}\}$

$B = \{\lambda_{12}, \lambda_{22}, \lambda_{32}, \lambda_{42}, \lambda_{13}, \lambda_{23}, \lambda_{33}, \lambda_{43}\}$

Proof: Obviously, $L_{uc} \subseteq L_m$, and any element of L_m has at least one element in the label set associated with each vertex of the underlying graph. Assume $L = (\Lambda_1, \Lambda_2, \ldots, \Lambda_n)$ is an element of L_m and that L has a component, say Λ_i such that $|\Lambda_i| \geq 2$. Let λ_j be an element of Λ_i. Considering v_i to be the root of the tree, we will define a recursive procedure for finding a globally consistent unambiguous labeling, $L' \subseteq L$ in which the label λ_j on vertex v_i participates. From this it can be concluded that L is not minimal.

By the definition of consistency, for each neighbor v_i' of v_i, there exists an element of λ_j' of Λ_i' such that (λ_j,λ_j') is a consistent pair of labels. We can apply this argument recursively for the neighboring vertices v_i' and the associated labels λ_j'. However, we need never reconsider a vertex which has been previously considered, since the underlying graph contains no cycles, and because the consistency relation is symmetric. Thus, the selection of consistent labels by this process will branch outward from the root node, and the will terminate when all pendant nodes have been reached. The result then, is a globally consistent unambiguous labeling contained in L as desired.

To prove the last part of the lemma, it must be shown that any consistent labeling can be expressed as the union of the unambiguous consistent labelings. Let v_i be any vertex of G, $\Lambda_i \in \Lambda$ the associated label set, and $\lambda_j \in \Lambda_i$ be any label in Λ_i. The recursive procedure defined above can be used to generate an atom $L' \in L_{uc}$ which contains the label λ_j on vertex v_i and which is contained in the original consistent labeling L. The union over the set of all labels in L of the atoms generated by this procedure is obviously equal to L, and the lemma has been proved.

This lemma can be used directly to derive the following result:

Lemma 4.2: Let $C(G,\Lambda,R)$ be any constraint network such that the underlying graph G is a tree. Let v_i be any vertex of G. Then there is no label associated with v_i which supports any other label on vertex v_i.

Comment: The effect of the removal of a label at a particular vertex v_i at time t is seen as propagating through the extent of the network. What this lemma says, in essence, is that in this class of networks, the effect cannot return to cause the removal of a different label from the label set for vertex v_i at some future time.

Proof: Let $L^{C'}$ be any consistent labeling, and let L^C be any maximal consistent sublabeling, $L^C \subseteq L^{C'}$. Then, from a basic result in lattice theory, $L^{C'} - L^C$ *is an*

atom, say, $L' \subseteq L_m$. Hence, any such difference contains exactly one label on each vertex of the graph. Then no label on any vertex can support any other label on that vertex.

In other words, if a labeling contains the label λ_j' on vertex v_i and if that labeling is consistent (and hence a fixed point of the relaxation operator), the label λ_j' will remain in the label set for all future time. For that matter, so will all the other labels in that labeling, since the labeling is a fixed point. If, on the other hand, there is no consistent labeling which contains the label λ_j' on vertex v_i then λ_j' must necessarily be removed at some future time step. The existence of such labeling, of course, is independent of the state of any of the other labels on vertex v_i.

The two lemmas shown above can be used directly to prove the following result.

Theorem 4.3: Let G be the underlying graph associated with the constraint network C, where G is a tree. Let l_p be the length of the longest open path in G. Then the discrete relaxation operator applied to C must settle in no more than $l_p - 1$ iterations.

Proof: Consider the worst case situation starting with a consistent labeling from which exactly one label, say λ_j on some vertex, say v_i, is removed. Consider v_i to be the root node of the tree, G. In order for the discrete relaxation operator to remain active, at least one label λ_j', on an adjacent vertex v_i' must be removed at the next iteration. This can happen only if λ_j on vertex v_i supports λ' on vertex v_i'. The argument is applied recursively to the neighboring vertex v_i'. However, no vertex from which a label was removed at a some iteration t, can have another label removed at some future iteration $t' > t$ since this implies that two labels on a given vertex support each other, in violation of lemma 4.2.

Let $C(G,\Lambda,R)$ be a constraint network. Let v_i be a vertex of G and let λ_j be a label associated with v_i. The label λ_j on vertex v_i is said to support the *vertex* v_i' if λ_j supports every label $\lambda_j' \in \Lambda_i'$, on vertex v_i'. The distance between two vertices in a connected graph G is defined to be the minimum length of all paths in G having those vertices as endpoints. Given a vertex v_i and a label λ_j associated with that vertex, a *supporting* path defined with respect to the label λ_j on vertex v_i, is a path in G from v_i to another vertex v_i' such that λ_j supports v_i'. The supporting path is called minimal if its only supported vertex v_i' is an endpoint. The radius of convergence of a label λ_j on a vertex v_i is defined to be the maximum length of a minimal supporting path defined with respect to λ_j on vertex v_i.

Example 4.4: Consider the constraint network $C(G,\Lambda,R)$ of figure 4.4. G is a path of length 7. The radius of convergence for label λ_2 on vertex v_4 is 4. The two minimal length convergence paths from λ_2 on this vertex are v_4, v_5, v_6, and v_7 to the right, and v_4, v_3, and v_2 to the left.

With these definitions, the following theorem can be stated and proved:

Theorem 4.4: Let G be the underlying graph of a constraint network $C(G,\Lambda,R)$ where G is a tree. If L is a labeling (possibly inconsistent) which contains at least one minimal labeling $L \in L_m$, then the settling time is bounded by $r_C - 1$, where r_C is the radius of convergence for C.

Proof: Consider a worst case situation which starts at a given consistent labeling $L^{C'}$ from which a given label λ_k on vertex v_i is removed. The result of applying the discrete relaxation operator to $L^{C'} - \{\lambda_{ik}\}$ is the largest consistent labeling $L^C \subseteq L^{C'}$. By assumption, L^C is not the null labeling. Therefore, let L be a minimal consistent labeling $L \subseteq L^C$ (refer to figure 4.5), which contains the label λ_j on vertex v_i. Let v_i' be the first vertex to the left, such that λ_{ij} supports v_i'. Let v_i''

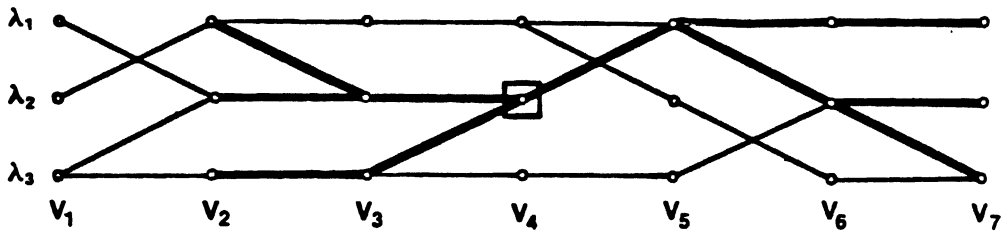


Figure 4.4: The constraint network for example 4.4. The minimal supporting paths for label λ_{42} are (to the left) v_4, v_3, and v_2, and (to the right) v_4, v_5, v_6, and v_7.

be the first vertex to the right such that λ_{ij} supports $v_{i''}$. Now the removal of λ_{ik} cannot cause the removal of any label from either of the vertices v_i' or $v_{i''}$ since there there is a path (through the product graph) from λ_{ij} to every label on v_i' and $v_{i''}$.

Corollary: The bound of $l_p - 1$ of theorem 4.3 is achieved only if the initial labeling Λ is contained in a minimal consistent labeling, or if $r_C = l_p$.

We note that the analysis applied to the class of constraint networks in which the underlying graph does not contain a cycle cannot be extended to the general case. In the first place, if the underlying graph contains a cycle, then there is the possibility that the product graph will contain a fundamental cycle, and the recursive argument used in lemmas 4.1 and 4.2 will not hold. In the second place, the set L_m of minimal labelings of the constraint network does not necessarily generate the set of *all* consistent labelings, as shown in the following example.

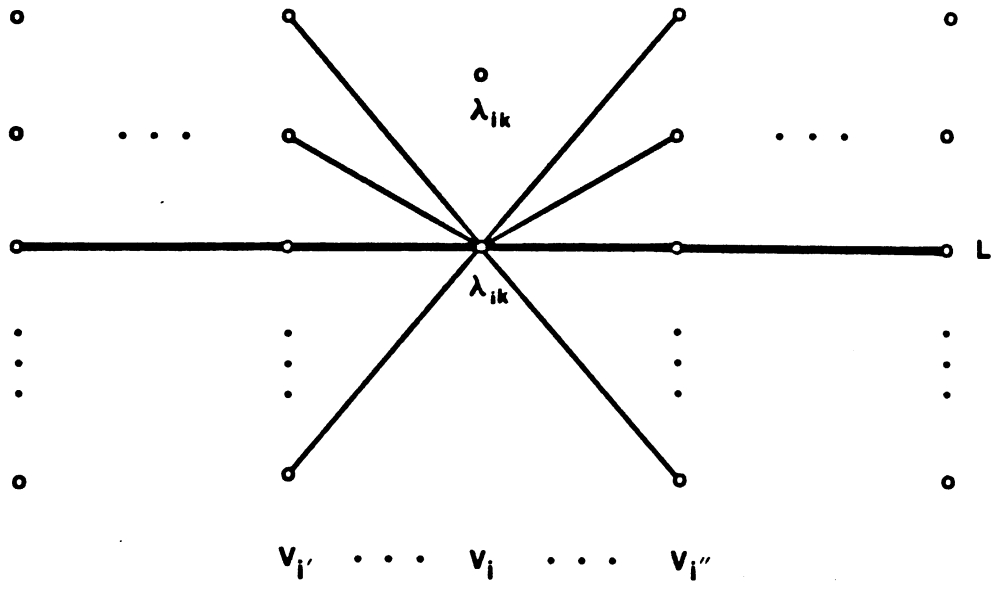


Figure 4.5: Constraint network for theorem 4.4.

Example 4.5: Consider a constraint network $C(G,\Lambda,R)$ with $G = C_4$. A labeling induced subnetwork of C is shown in figure 4.6. The labeling is consistent, and is the union of the fundamental cycles F_1 and F_2 and the path P. Since P is not a subgraph of a fundamental cycle, the given labeling could not be generated by the union of L_m.

4.1.4. Simulation Results

We have not been able to extend the results of the previous subsection to the general case (that is, to the case where the underlying graph contains a cycle). In leu of analytical results, simulation experiments were performed on the constraint networks designed for the edge linking application of the Appendix[1]. The objective was to examine the rate at which the critical times, both average and

[1] Label λ_{21}, the "knot" was ommitted for this experiment.

worst case, increase with the size of the network. Other important questions, such as how the critical times depend on the nature of the network (i.e., the underlying graph and constraint relations), certainly exist. However, an examination into such issues was not attempted.

The simulation was performed on networks corresponding to square rasters of size $i \times i$ for $i = 4, 6, 8, \cdots, 18$. Initial labelings were generated randomly. Experiments were conducted with the probability that a given label on a given vertex was in the initial label set in the range $p = 0.1, 0.2, \cdots, 0.9$. For each value of i and p given above a total of 1024 experiments were conducted. A larger number of experiments for each case would have been more desirable, however the resources needed to conduct these experiments were not available.

The fact that the limitation on the number of experiments greatly affected the outcome is reflected in the worst case settling time results, which are shown in table 4.1. This table contains the worst case overall settling times and the worst case detection times. The worst case critical times for a given $i \times i$ network is taken over the set of all experiments conducted for all labeling densities p. Thus, this table reflects the results of 9216 experiments per network size. Despite this, the results obviously do not reflect in any way the actual values. This is not too surprising, however, when considering the fact that there are 21^{i^2} possible labelings for a square raster with i rows and i columns. If nothing else, it is appears reasonable to conclude from the resulting data that the number of "bad" situations, that is, initial labelings such that the settling times are large, are very limited.

There is further evidence for this in the way in which the settling time are distributed. Table 4.2 gives the histogram for an 8×8 network with $p = 0.5$. The shape of the distribution shown here is typical of the other experiments. Finally, a series of simulations designed specifically to determine the worst case

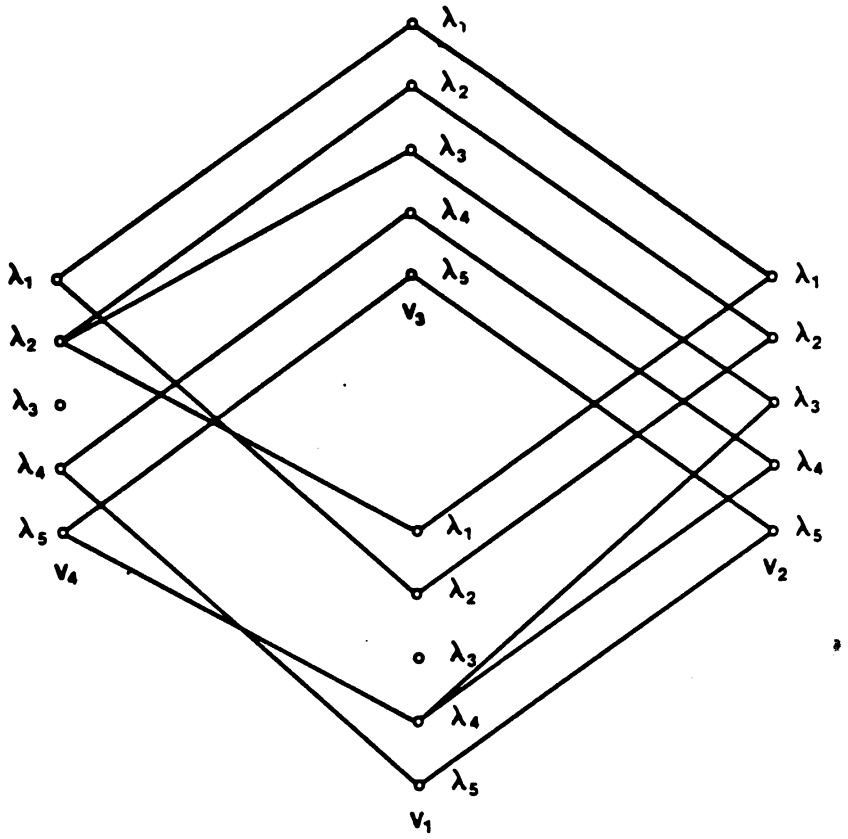


Figure 4.6: Constraint network generated by two fundamental cycles F_1 and F_2, and an inconsistent labeling P, where:

$F_1 = \{\lambda_{11}, \lambda_{21}, \lambda_{31}, \lambda_{41}, \lambda_{12}, \lambda_{22}, \lambda_{32}, \lambda_{42}\}$,

$F_2 = \{\lambda_{14}, \lambda_{24}, \lambda_{34}, \lambda_{44}, \lambda_{15}, \lambda_{25}, \lambda_{35}, \lambda_{45}\}$,

$P = \{\lambda_{14}, \lambda_{23}, \lambda_{33}, \lambda_{42}$.

critical times was performed. These simulations started with the universal labeling from which a randomly chosen label, on a randomly chosen vertex was removed. The discrete relaxation operation was then applied to this labeling until the network settled, resulting in a consistent labeling. A randomly chosen label was then removed from the resulting labeling and the discrete relaxation operation was applied again. This process was repeated until the null labeling resulted. However, the worst case settling times for these experiments turned out in fact to be lower than the ones for the original experiments. This again

points to the apparent difficulty in finding the worst case settling times for a complex constraint network. As before, the conclusion to be drawn is that the relative number of initial labelings resulting in large settling times is small.

The average critical time results appeared to be more reasonable. Table 4.3 shows the average critical time results for the different size networks for various initial labeling densities p. Figure 4.7 is a plot of the critical times vs. network size for p = 0.5, which shows that both the average settling time and the average detection time increases approximately linearly with an increase in the number of rows or columns and linearly with the square root of the overall size. Note that the critical times for are almost constant in the size of the network, for p = 0.8. This result is consistent with other experiments with high labeling densities conducted on networks of this type. Finally, note that the detection

size	settling times	detection times
4 x 4	15	15
6 x 6	27	27
8 x 6	32	32
10 x 10	36	36
12 x 12	33	33
14 x 14	39	35
16 x 16	36	51
18 x 18	41	41

Table 4.1: Observed worst case critical times for the discrete relaxation process.

times for $p = 0.2$ in fact decrease for the networks of size 16×16 and 18×18. This can be explained by the fact that the detection time is the minimum of the settling and null time and at low densities, the probability that a particular vertex is assigned the null labeling to start with is relatively high.

settling time	number of occurrences	settling time	number of occurrences	settling time	number of occurrences
1	0	8	14	15	1
2	4	9	2	16	2
3	336	10	4	17	1
4	418	11	2	18	1
5	156	12	3	19	0
6	49	13	3	20	0
7	27	14	0	21	1

Table 4.2: Distribution of worst case settling times for a 16×16 network with $p = 0.5$.

size	p = 0.2		p = 0.5		p = 0.8	
	settling time	detection time	settling time	detection time	settling time	detection time
4 x 4	4.554	0.535	2.213	2.213	2.000	2.000
6 x 6	4.760	0.794	2.658	2.655	2.000	2.000
8 x 8	4.870	0.650	3.049	3.038	2.000	2.000
10 x 10	4.919	0.510	3.361	3.361	2.002	2.002
12 x 12	4.907	0.349	3.671	3.653	2.000	2.000
14 x 14	4.982	0.225	3.973	3.905	2.001	2.001
16 x 16	5.045	0.175	4.219	4.172	2.004	2.004
18 x 18	5.049	0.110	4.439	4.274	2.001	2.001

Table 4.3: Average critical times for p = 0.2, p = 0.5, and p = 0.8.

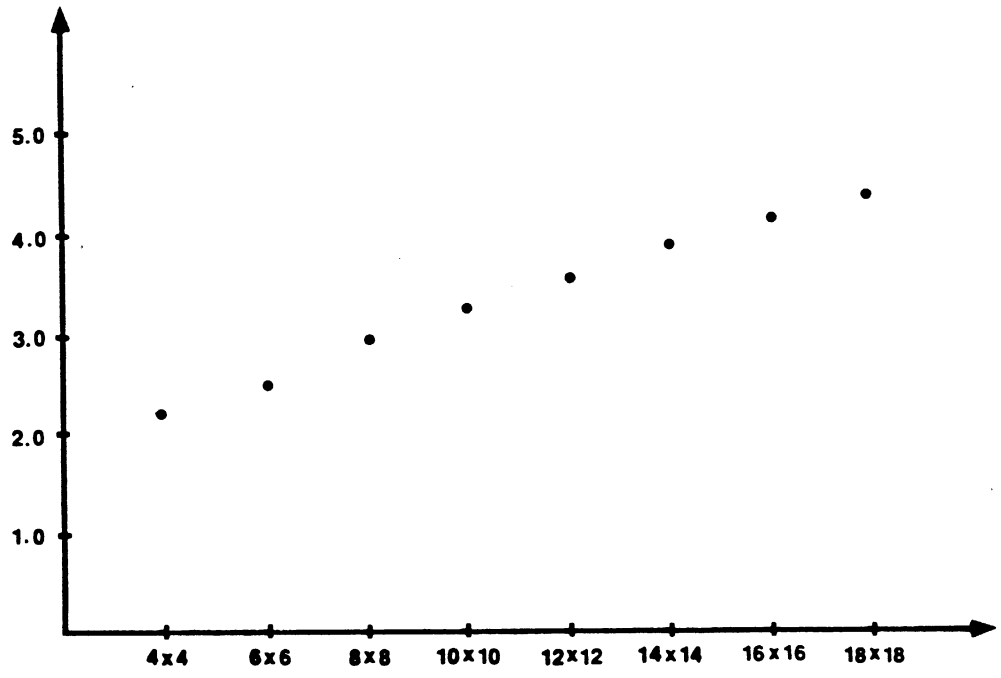


Figure 4.7: Plot of settling times vs. network size for p = 0.5.

4.2. The Structure of the Graph Labeling Problem

A basic description of the extrema and facets of the bounding polytope of a class of 0-1 integer programs is usually sought after in order to increase the understanding of the nature of the problem and to aid in the development of more efficient ways to solve it. This information is useful in the generation of cutting planes for example, and in the derivation of heuristic solutions. Of further interest is the study of the linear programming relaxation of the original problem, that is, the original problem without integrality constraints on the variables. Let:

[P1] maximize: $c^T x$

subject to: $A x = e, \quad x_{ij} \in \{0,1\}$

be a set partitioning problem and $\bar{x}$ be a solution to the linear programming relaxation of [P1]. Let

$$\bar{c} = c - c_B^T B^{-1} A$$

be the reduced costs from an optimal basis B. Then, the problem:

[P2] maximize: $\bar{c}^T x$

subject to: $A x = e, \quad x_{ij} \in \{0,1\}$

is equivalent to [P1], since, $\bar{c}_B^T B^{-1} A x$ is constant for any feasible x. Furthermore it has been reported [BaP74] that the performance of the implicit enumeration techniques applied to this problem is greatly enhanced if [P2] is solved instead of [P1]. This suggests a possible benifit in finding efficient ways for solving (LGLP).

Finally, the behavior of the Lagrange dual approaches discussed in the following section are thought to be related to the polytope of the linear programming relaxation of the graph labeling problem. The intent was to be able

to describe the values of the primal variables in the LP relaxation of the graph labeling problem directly from the results of this algorithm. The analysis of the following section is not complete to the degree that such a tasks can be pursued. It is hoped, however, that the results which are presened will motivate further work.

There are two basic approaches to characterizing the polytopes of a class of 0-1 integer programs. In the first case, these characterizations can be based on the constraint matrix, and in particular, based on the existence (or absence) of submatrices of a particular form. Equivalently, but perhaps more useful because the results are easier to visualize, are the topological characterizations which are based on the derived graph of the constraint matrix. Efforts in this area seem equally divided between these two approachs, and depend for the most part on the underlying application. Obviously, the topological characterizations are the natural approach for the graph labeling problem.

An important element in current work on topological characterizations is the perfect graph, and related theorems [Gol80] which specify necessary and sufficient conditions for the linear programming relaxation to have all integer extrema. Work on characterizing the non-integer extrema related to 0-1 matrices is found in the theory of blocking and anti-blocking polyhedra [Ful71] as well as in studies of special classes of problems such as the vertex and set packing problems [NeT74,NeT75,Pad73].

4.2.1. The LP Relaxation of the Graph Labeling Problem

4.2.1.1. Original Form of the Problem

As a trivial extension to established results concerning the vertex packing polytope, the polytope for (LGLP) can be completely characterized. Let $C(G,\Lambda,R)$ be a constraint network, where G is a graph with n vertices, and Λ is a set with m

labels. Let [P3] be the LP relaxation of a graph labeling problem, with the first constraint (equation 3.3 of chapter 3) replaced with an inequality as:

$$[P3] \quad \text{maximize:} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}, \tag{4.1}$$

$$\text{subject to:} \quad \sum_{j=1}^{m} x_{ij} \leq 1, \quad i=1,\ldots,n, \tag{4.2}$$

$$x_{ij} + x_{i'j'} \leq 1 \tag{4.3}$$

Problem [P3] with the integrality constraints imposed on the variables will be referred to below as the *set packing relaxation* of the associated graph labeling problem. Clearly, any extreme point of the associated graph labeling problem (where equation 4.2 is replaced with an equality) is an extreme point of [P3]. Integer extrema of [P3] in which the inequality 4.2 is strict correspond to solutions in which there exists a vertex to which no label has been assigned (i.e. $x_{ij} = 0$ for all $j=1,\cdots,m$ for a given vertex v_i). Consider the following modification of [P3]:

$$[P4] \quad \text{maximize:} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}, \tag{4.4}$$

$$\text{subject to:} \quad x_{ij} + x_{ij'} \leq 1, \quad j,j'=1,\ldots,m, \quad i=1,\ldots,n, \tag{4.5}$$

$$x_{ij} + x_{i'j'} \leq 1 \tag{4.6}$$

for every pair of *inconsistent* labels $(\lambda_j,\lambda_{j'})$ on adjacent vertices v_i and $v_{i'}$,

$$x_{ij} \in \{0,1\} \tag{4.7}$$

Problem [P4] is similar to [P3] except that equation 4.2 has been replaced by equation 4.5. Replacing equation 4.2 with 4.5 obviously does not change the integer programming polytope. Furthermore, every extrema of [P3] is an extrema of [P4]. In order to show this the following, which result from the work found in

[Pad73] is needed.

Lemma 4.5: The inequality

$$\sum_{j=1}^{m} x_{ij} \leq 1, \quad i=1,\ldots,n$$

is a facet of the integer programming formulation of [P4].

From this, the following lemma can be proved directly:

Lemma 4.6: Every extrema of [P3] is an extrema of [P4].

proof: Let constraints of the form

$$\sum_{j=1}^{m} x_{ij} \leq 1, \quad i=1,\ldots,n \qquad 4.8$$

be added as a cutting plane to the polytope of problem [P4]. From the previous lemma, these hyperplane intersects the polytope of [P4] only at integer points. Furthermore, the polytope described by the hyperplanes of equation 4.8 themselves obviously has only integer extrema. Therefore, since the integer extrema of [P3] and [P4] are the same, the result follows.

Problem [P4] is a vertex packing problem. Therefore, the following result from from [NeT74] can be used to completely characterize the polytope of the graph labeling problem:

Theorem 4.7: Let x_{ij} be an extreme point of the LP relaxation of a graph labeling problem. Then $x_{ij} = 0, \frac{1}{2}$, or 1 for all x_{ij}.

It is not difficult to see that what this theorem means in terms of a graph labeling problem is that the LP solutions will break into connected "regions" of either all integer solutions (corresponding to a choice of a single label at those vertices) and or solutions with two variables, say x_{ij} and $x_{ij'}$ equal to ½. Obviously, internal to any all integer region, each label choice is consistent with

all its neighbors. Consistency is also obviously maintained at the boundaries. That is, at the labels corresponding to the non-zero basic variables are consistent at the boundaries.

4.2.1.2. The Feasible Region Under PMI Constraints

Throughout this section, it will be assumed that the continuous graph labeling problem will be formulated in terms of PMI constraints. We have the following basic results:

Theorem 4.8: The non-zero basic vectors in any optimal solution to the LP relaxation of a graph labeling problem are consistent in the discrete sense. That is, if x_{ij} be a non-zero basic variable in an optimal solution, then for each neighboring vertex $v_{i'}$ there exists a non-zero basic variable $x_{i'j'}$ such that $r_{ii'}(\lambda_j,\lambda_{j'}) = 1$.

Proof: Assume otherwise. Then there exists a pair of adjacent vertices, v_i and $v_{i'}$ and a non-zero basic variable x_{ij} associated with vertex v_i which is inconsistent with every non-zero basic variable on vertex $v_{i'}$. Let $x_{i'j_1}, x_{i'j_2}, \ldots, x_{i'j_s}$ be the non-zero basic vectors on vertex $v_{i'}$. Then by the nature of the PMI constraints, the variable x_{ij} as well the set of non-zero basic vectors on vertex $v_{i'}$ must be contained in at least on PMI set. Therefore,

$$x_{ij} + x_{i'j_1}, x_{i'j_2}, \ldots, x_{i'j_s} \leq 1 \qquad 4.9$$

Furthermore, the sum of all variables at a given vertex must be 1:

$$x_{i'j_1}, x_{i'j_2}, \ldots, x_{i'j_s} = 1 \qquad 4.10$$

Equations 4.4 and 4.2 imply that $x_{ij} = 0$ contradicting the assumption that x_{ij} is a non-zero basic variable.

The following discussion gives a sufficient condition for the polytope of (LGLP) to have all integer extrema. This discussion depends on the results

related to the extrema associated with 0-1 matrices. A brief outline of the necessary definition and results are given here. A more thorough treatment can be found in [Gol80].

Definitions: Let G be an undirected graph.

The *clique number*, $\omega(G)$, of G is the number of vertices in a maximum clique of G.

The *clique cover number*, $\kappa(G)$, of G is the smallest number of cliques (not necessarily maximal) needed to cover the vertices of G.

The *stability number*, $\alpha(G)$, of G is the cardinality of a maximal stable set in G, where a *stable set* (or *independent set)* of a graph is a subset of the vertex set in which no element is adjacent to any other element.

The *chromatic number*, $X(G)$, of G is the smallest number of colors required to properly color G.

Let $G = (V,E)$ be a graph with vertex set V and let $A \subseteq V$. The subgraph induced by A is a graph $G_A = (A,E_A)$ where a pair (v_i,v_j) is in E_A if and only $v_i \in A$, $v_j \in A$, and $(v_i,v_j) \in E$.

A *perfect graph* [Ber73] is one in which

[1] $\omega(G_A) = X(G_A)$ for all $A \subseteq V$,

and

[2] $\alpha(G_A) = \kappa(G_A)$ for all $A \subseteq V$,

The importance of this class of graphs is found in the theorem given below. Let A be the clique matrix of an undirected graph G. Define:

$$P(A) = \{ x \mid Ax \leq 1 \}$$

and

$$P_I(A) = \text{convex hull } \{ x \mid x \in P(A), x \text{ integral} \}.$$

Theorem 4.9 [Chv75]: If A is the clique matrix[2] of an undirected graph G, then G

is perfect if and only if $P_I(A) = P(A)$.

From this, the following result is obtained:

Lemma 4.10: The augmented complement graph of a constraint network with only two vertices is a perfect graph.

Proof: It is straightforward to show that $\alpha(G) = \kappa(G)$. If there exists a consistent pair of labels, say λ_j on vertex v_1 and label $\lambda_{j'}$ on vertex v_2, then clearly $\kappa(G) = 2$, since no maximal clique will cover all the labels on both vertices. Furthermore, $\alpha(G) = 2$. Label λ_j on vertex v_1 and label $\lambda_{j'}$ on vertex v_2 form a stable set. If there does not exist a consistent pair of labels, then a single PMI set covers all the labels on both vertices, so that $\alpha(G) = \kappa(G) = 1$. Since this holds for any two vertex constraint network G, it will also hold for any induced subgraph of G.

Thus, the node-clique matrix, A of the product graph of a given two-vertex constraint network is perfect. Then the polytope defined by

$$x \geq 0, \quad Ax \leq 1 \qquad 4.11$$

contains only (0-1) integer extrema. In the continuous graph labeling problem, the first two rows of $Ax \leq 1$ are constrained to be equalities, that is:

$$x_{11} + x_{12} + \cdots + x_{1m} = 1,$$

$$x_{21} + x_{22} + \cdots + x_{2m} = 1.$$

It is easy to see, however, that the resulting region will be a facet of the polytope described by equation (4.11). By a basic result [Mur76], every extreme point of a facet of a polytope is an extreme point of the polytope, therefore:

Lemma 4.11: For a continuous graph labeling problem based on a constraint network with only two vertices, the feasible region has all 0-1 integer extreme

---

[2] The clique matrix of G has a row for every clique of G and a column for every vertex. The entry for row i and column j is 1 if vertex v_j is in clique i and 0 otherwise.

points.

This result can be extended directly to any graph labeling problem in which the underlying graph is a path:

Lemma 4.12: The feasible region for a graph labeling problem for which the underlying graph is a path has all integer extrema when expressed in terms of PMI constraints.

Proof: Let G be the augmented complement graph of the constraint network. It must be shown that $\alpha(G') = \kappa(G')$ for every induced subgraph G' of G. Let G' be any induced subgraph of G. Assume G' is connected (otherwise each component can be treated separately). We show that there is an independent set of vertices, V', and a clique cover K', as well as a one-to-one correspondence between the elements of V' and K'.

The theorem is proven by induction on the number of vertices in the underlying graph. The case $|V| = 2$ is lemma 4.11. First note that if G' contains a globally consistent labeling $\bar{\lambda} = \lambda^1\lambda^2 \cdots \lambda^n$, then G' is perfect. In this case, the n' vertices of the induced graph G' corresponding to the labels in $\bar{\lambda}$ form an independent set. Therefore G' can by covered by n' cliques, that is, those cliques corresponding to the set of all labels at a given vertex.

Now assume that G' does not contain a globally consistent labeling. Let $\Lambda_i \subseteq \Lambda$ be the set of labels currently associated with vertex $v_i \in V'$. Let $L_1 = \Lambda_1$. Let L_2 be the set of all $\lambda^2 \in \Lambda_2$ which are consistent with at least one $\lambda^1 \in L_1$. Likewise, let L_3 be the set of all Λ_3 which are consistent with at least one $\lambda^2 \in L_2$. In the same manner, define L_4, L_5, and so forth.

Let t be the first vertex such that L_t is empty. Note that such a t must exist, otherwise, we could find a consistent path through the product graph, that is, there would exist a globally consistent labeling in G' to be derived as follows:

pick an element, λ^n of L_n. Pick an element of λ^{n-1} of L_{n-1} which is consistent with label λ^n on vertex v_n (the existence of such an element is guaranteed by construction). Likewise pick a label λ^{n-2} from L_{n-2} which is consistent with label λ^{n-1} on vertex v_{n-1}. Continue in a similar manner until a globally consistent labeling $\bar{\lambda} = \lambda^1\lambda^2 \cdots \lambda^n$ has been constructed.

Now consider the graph G'' induced from G' by considering all vertices in the product graph corresponding to the labels in the sets, $\Lambda_1, \Lambda_2, \ldots, \Lambda_t$. We show that $\alpha(G'') = \kappa(G'') = t - 1$. Since the sets L_1 through L_{t-1} are not empty, there exists a consistent path through the first $t - 1$ vertices of the underlying graph of G'', and it follows easily that $\alpha(G'') \geq t - 1$. Define $\bar{L}_i = \Lambda_i - L_i$, and note that $\bar{L}_i$ and L_i forms a partition of Λ_i. Likewise, $L_1, \bar{L}_1, L_2, \bar{L}_2, \ldots, L_t, \bar{L}_t$ forms a partition of all the vertices in the complement graph of G''. Note finally that every element of $\bar{L}_{i+1}$ is inconsistent with every element of L_i. Thus the elements in the set $L_i \cup \bar{L}_{i+1}$ are contained in some clique, say, K_i. Since $\bar{L}_1 = L_t = \phi$, the cliques K_i, $i=1,\ldots,t-1$ cover the vertices (labels) of G'' (refer to figure 4.8), so $\kappa(G'') = t - 1$.

To complete the proof, let $\bar{G}'' = G' - G''$. It is clear that any set of vertices (labels) in a stable set of $\bar{G}$ is independent of the independent set selected by the process given above. Thus, $\alpha(G) = \kappa(G)$ if and only if $\alpha(\bar{G}) = \kappa(\bar{G})$, which follows from the induction hypothesis.

The above theorem can be generalized to any graph labeling problem where the underlying graph does not contain a cycle. Obviously, only connected components (trees) need be considered.

Theorem 4.13: Let **A** be the constraint matrix under PMI constraints for any graph labeling problem where the underlying graph is a tree. Then $P(A,e) = P_I(A,e)$.

Comment: The strategy used here, will be the same as that used in the above lemma. That is, a subtree G'' will be found in which $\alpha(G'') = \kappa(G'') = t - 1$, where

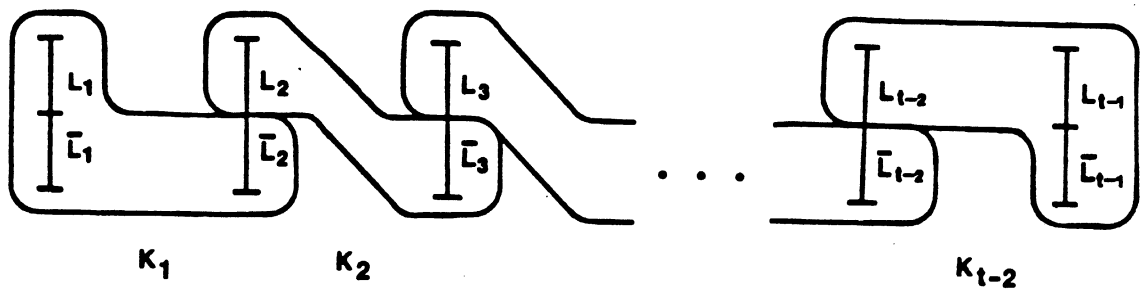


Figure 4.8: Covering Cliques for the Proof of Lemma 4.12.

t is the number of vertices in the graph underlying G''. A maximal independent set will be derived which will be independent of any of the vertices of $\bar{G} = G' - G''$. The result will then follow by induction.

Proof: Assume the theorem holds for any graph labeling problem where the underlying graph is a tree with n−1 or fewer vertices. Let G be the augmented complement graph corresponding to a graph labeling problem where the underlying graph has n vertices. Let G' be an induced subgraph of G. Assume without loss of generality that G' is connected. As above, if there is a globally consistent labeling in G' then the result follows because $\alpha(G') = \kappa(G') = n$. So assume otherwise.

Arbitrarily choose a root node in the underlying graph. Let V_1 be the set of leaf nodes of the graph thus rooted. For each vertex $v_i \in V_1$, let $L_i = \Lambda_i$. Likewise, let V_2 be the set of all predecessors of vertices in V_1. Let $v_2 \in V_2$, let

$v_{i_1}, v_{i_2}, \ldots, v_{i_r}$ be the successors in the underlying graph of v_2, and let $L_{i_1}, L_{i_2}, \ldots, L_{i_r}$ be the associated label sets. Let $L_2 \subseteq \Lambda_2$ be the set of labels λ on vertex v_2 where λ is consistent with at least one label in each of $L_{i_1}, L_{i_2}, \ldots, L_{i_r}$ (refer to figure 4.9). The label sets, L, for the set V_3 of predecessors of the vertices in V_2, are defined in a similar manner. At some point, a level will be reached in which there exists a label set, say L_s, associated with a vertex v_s which is empty. Otherwise, using the same construction as in the lemma above, there would have to exist a globally consistent labeling within L.

Consider the subtree G'' rooted at v_s. For this subtree, $\alpha(G'') = \kappa(G'') = p - 1$, where p is the number of vertices in G'', and a maximal indepedent set exists in which no label is assigned to vertex v_s, which is shown

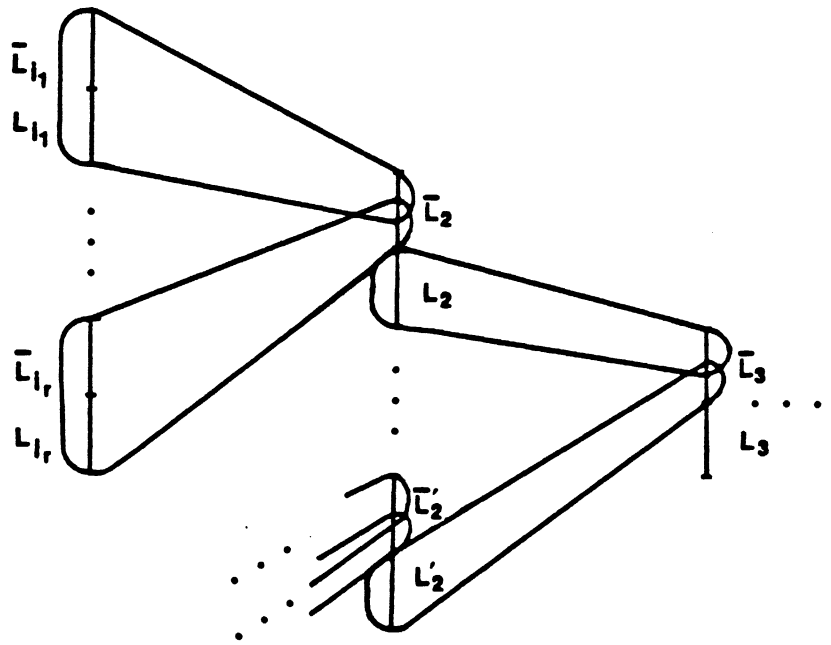


Figure 4.9: Covering Cliques for the Proof of Theorem 4.13.

as follows: Let $\Gamma_i = \Lambda_i - L_i$. With all vertices, v_i in the set V_1 of leaf nodes, associate the clique K_i which contains the elements of L_i and all the labels in the predecessor v_r which are inconsistent with each label in L_i. Evidently,

$$L_2 = \bigcup_{v_i \text{ successor of } v_2} K_i \cap \Lambda_2$$

The cliques associated with the vertices in the set V_2 are defined in a similar manner. Because $L_s = \phi$, the set of cliques for the successors of v_s "cover" Λ_s. As in the lemma above, for each of the subtrees rooted at each of the successors of v_s there exists a globally consistent labeling. The result then follows.

Note that the PMI constraints are all the cliques of the augmented complement graph of a graph labeling problem only when the underlying graph does not contain a clique with more than 2 vertices. When the underlying graph does contain a cycle of length 3, a clique of the augmented complement graph may in fact span all vertices vertices in the clique of the underlying graph. With respect to the more general situation where the graph labeling problem is expressed in terms of the cliques of the augmented complement graph, we pose the following conjecture:

Conjecture 4.14: Let $C(G,\Lambda,R)$ be a constraint network with underlying graph G. If the clique graph of G does not contain any cycles, then the polytope of the associated graph labeling problem expressed in terms of the cliques of the augmented complement graph has all integer extrema.

When cycles are introduced into the underlying graph of a constraint network, the associated polytope expressed in terms of PMI constraints will have non-integer extrema. An attempt was made to characterize the non-integer extrema of a graph labeling problem in this case in a manner similar to the characterization of theorem 4.7. Initial work towards this goal has produced the

following result and the conjectures of the following subsection:

Result 4.15 : Let C(G,Λ,R) be a constraint network, in which the underlying graph, G, is a cycle. If the labels in the fine graph corresponding to the non-zero basic variables form a fundamental cycle, say F(p), then $x_{ij} = \frac{1}{p}$, $x_{ij} \in F(p)$, and $x_{ij} = 0$ is an extreme point of the corresponding polytope. Furthermore, it is the only extreme point in which $x_{ij} > 0$, $x_{ij} \in F(p)$, and $x_{ij} = 0$ otherwise holds.

Proof: First it is shown that this is the only feasible solution for which the stated conditions hold. Assume otherwise: Then there exists an $x_{ij} = \frac{1}{p} - \varepsilon_{ij}$, with $\varepsilon_{ij} > 0$. Assume, without loss of generality, that $x_{11} = \frac{1}{p} - \varepsilon_{11}$ with $\varepsilon_{11} > 0$. Assume furthermore, that λ_{11} is consistent with λ_{21}, and so forth, as shown in figure 4.10. Then, since

$$\sum_{j=2}^{p} x_{1j} = \frac{p-1}{p} - \varepsilon_{11}$$

it can be concluded that $x_{21} = \frac{1}{p} - \varepsilon_{21}$ with $\varepsilon_{21} > 0$, since the labels $\lambda_{12}, \lambda_{13}, \ldots, \lambda_{1p}, \lambda_{21}$ are covered by a clique. Continuing in this manner results in $x_{ij} = \frac{1}{p} - \varepsilon_{ij}$ with $\varepsilon_{ij} > 0$ for all $x_{ij} \in F(p)$. This is impossible, since it implies that:

$$\sum_{j=1}^{p} x_{ij} = 1 - \sum_{j=1}^{p} \varepsilon_{ij} < 1.$$

4.2.2. The Structure of the Graph Labeling Problem: Summary

Initial theoretical results related to some aspects of the graph labeling problem have been presented above. The following chapter covers algorithms for solving it. The discussion of algorithms is based on what is known about the graph labeling problem at this point. It is felt, however, that better algorithms, including good heuristic approaches would be possible if more were known about the basic nature of this problem. What we hoped could be accomplished was a

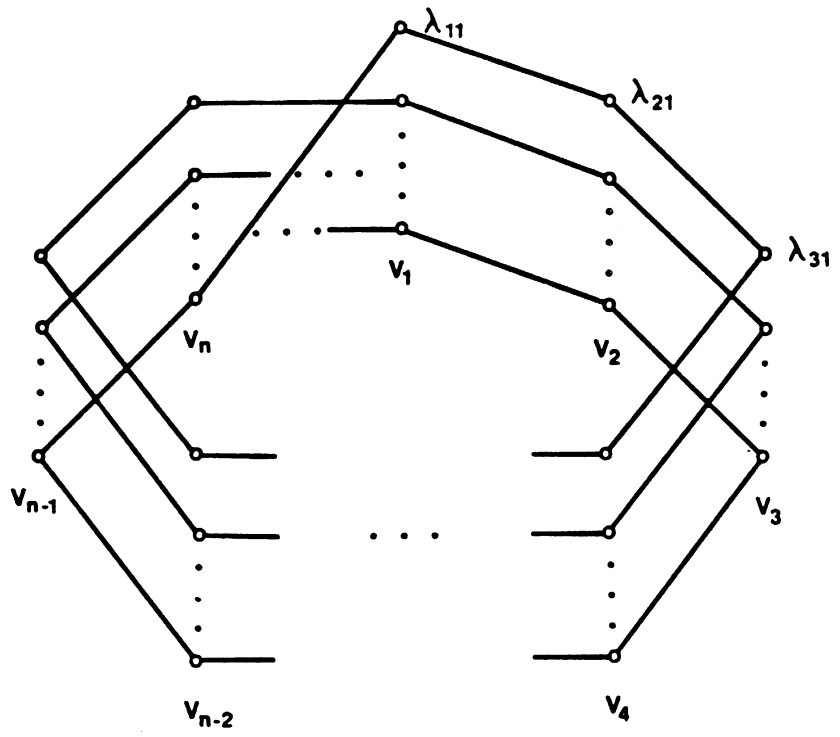


Figure 4.10: Fundamental Cycle for Result 4.15.

complete specification of the polytope for linear programming relaxations of the graph labeling problem as well as for the original (integer) program. In that sense, the work presented here is far from complete. The approach which was to be taken was to specify the extrema first for the graph labeling problem where the underlying graph was a cycle and then extend these results to the more general case. The results which were anticipated from this work are expressed in the following two conjectures:

Conjecture 4.16: For a constraint network under PMI constraints where the underlying graph is a cycle, the *only* fractional extrema correspond to the fundamental cycles, $F(p)$ with $x_{ij} = \frac{1}{p}$ as in lemma 4.15 above, and the fractional extrema of lemma 4.7.

Conjecture 4.17: In a solution to the linear programming relaxation of a graph labeling problem, the vertices of the underlying graph can be separated into connected regions, where the non-zero basic variables associated with the vertices of a given region all take on the same value.

More important to the theory developed here is a characterization of the facets of the polytope of the (integer) graph labeling problem. Note that because the problem is defined in terms of equality as well as inequality constraints, this polytope will be degenerate in most cases. Thus there may be in fact an infinite number of ways to specify the bounding polytope. It would be worthwhile to be able to specify this polytope in a manner such that a sufficient set of constraints can be easily recognized by reference to either the product graph or the (augmented) complement graph of a given graph labeling problem. Most descriptions of the facets of the set partitioning and set (vertex) packing polytopes are given in terms of the forms of certain subgraphs of the derived graph of the constraint matrix. As noted before, one of the aspects of the graph labeling model which makes issues like this easy to approach is the direct relationship between the problem definition and the graphical characterization. As a final note on the work towards a theory for the graph labeling problem, the following conjecture is presented:

Conjecture 4.18: Consider the case of a constraint network $C = (G,\Lambda,R)$ in which the underlying graph is a cycle with n vertices. Let A be the set of fundamental anti-cycles defined on G. Let S be a subset of A such that the union of all the labels in each of the cycles does not contain within it a unambiguous globally consistent labeling, and which is maximal in that sense. Then the set of constraints of the form:

$$\sum_{x_{ij} \in S} x_{ij} \leq n - 1$$

for all subsets $S \subseteq A$ which satisfy this critera, along with the PMI constraints, are sufficient to guarantee only integer extrema of the associated graph labeling polytope.

CHAPTER V

ALGORITHMS

The following chapter presents a discussion of various approaches to the graph labeling problem. As noted previously, an algorithm which guarantees a primal feasible solution must in the general case involve an enumeration scheme. The question is how and to what extent can this enumeration process can be augmented by other techniques. The discussion here focuses on means by which the structure of this particular problem can be used to generate upper bounds for a given candidate subproblem in a basic branch and bound approach.

5.1. Dynamic Programming Approaches

Dynamic programming can be used in the solution of combinatorial optimization problems when the problem constraints are posed in a group theoretic manner. Although group theoretic approaches cannot be used to solve the general 0-1 integer program, a dynamic programming approach, similar to the Vitirbi algorithm, can be used to solve a graph labeling problem when the underlying graph is a path. This algorithm can then be used as a building block in the further development of algorithms for the generation of bounds for the original problem.

5.1.1. The Case Where the Underlying Graph is a Path

In the case where the underlying graph is a path a decentralized process for solving the graph labeling problem does exist. In order to describe the dynamic

programming algorithm used solve the problem under these conditions, assume that a graph $G = (V,E)$ with n vertices $v_1, v_2, \ldots, v_n$ is given. The graph is a path with vertex v_1 adjacent to vertex v_2 and vertex v_2 adjacent to vertex v_3 and so forth. For the sake of discussion, assume the path is oriented from left to right with vertex v_1 on the left and vertex v_n on the right as is shown in figure 5.1. The assumption is that the architecture used to solve this problem specifies a processor for each label on every vertex of the graph, which is performing the updating of the current labeling value for the associated label on that vertex.

In fact, two independent processes are needed. One will transmit labeling information to the right, and the other will transmit information to the left. The current labeling values for the process with data moving to the right will be denoted as R_{ij}^t for the value associated with label λ_j on vertex v_i at iteration t.[1] Likewise the current labeling values for the process with data moving to the left will be denoted as L_{ij}^t.

Figure 5.1: Graph underlying the graph labeling problem for the dynamic programming approach.

[1] The current labeling value R_{ij}^t should not be confused with tha constraint relation $R_{ii'}$. The former will always be shown with the superscript t, while the latter will not.

Initially, these values will be set equal to the labeling value inputs for the original problem:

$$L_{ij}^0 = R_{ij}^0 = c_{ij}^0 \quad \text{for all } i,j. \tag{5.1}$$

In order to describe the updating procedure, consider first the process with data moving to the right, and a given processor performing the updating for label λ_j on vertex v_i. This processor looks to its left and considers the set $\hat{\Lambda}_{i-1,j}$ of labels on vertex v_{i-1} with which the label λ_j on vertex v_i is consistent. That is, $\hat{\Lambda}_{i-1,j} = \{\lambda_{j'} \in \Lambda \mid (\lambda_{j'},\lambda_j) \in R_{i-1,i}\}$. Of those labels, it takes the maximum of the associated labeling values and adds it to the *initial* labeling value, c_{ij}^0. The result is the updated labeling value R_{ij}^{t+1}. In other words the updating rule is given as:

$$R_{ij}^{t+1} = c_{ij}^0 + \max_{\lambda_{j'} \in \hat{\Lambda}} \{ R_{i-1,j'}^t \cdot r_{i-1,i}(\lambda_{j'},\lambda_j) \} \tag{5.2}$$

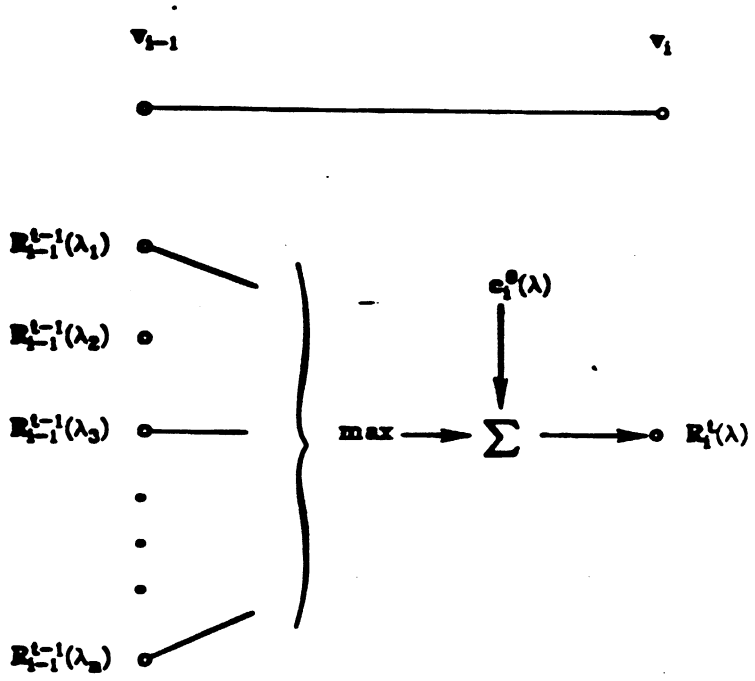
where, as given in section 2 of chapter 3

$$r_{i-1,i}(\lambda_{j'},\lambda_j) = \begin{cases} 1 \text{ if } (\lambda_{j'},\lambda_j) \in R_{i-1,i} \\ 0 \text{ otherwise} \end{cases} \tag{5.3}$$

This updating rule is shown schematically in figure 5.2 below. The strategy for the process with data transmitted to the left is similar except that each processor refers to the set of consistent labels on the vertex to the immediate right. The updating rule is then given as:

$$L_{ij}^{t+1} = c_{ij}^0 + \max_{\lambda_{j'} \in \hat{\Lambda}} \{ L_{i+1,j'}^t \cdot r_{i,i+1}(\lambda_j,\lambda_{j'}) \} \tag{5.4}$$

This is an inherently sequential process. However, there is no reason that it cannot be implemented in terms of a parallel iterative procedure. Assume that there is a unique solution to the problem as defined, that is, there is a unique unambiguous consistent labeling with a maximum sum of the initial labeling values. Then, given the updated labeling values for the process with data flow to the right for any iteration $t \geq n$, we can make a correct decision by choosing that



$$R_i^0(\lambda) = c_i^0(\lambda) \quad \text{for all } i, \lambda$$

Figure 5.2: Illustration of updating rule for the dynamic programming approach for process with data flow to the right.

label λ_j on vertex v_n such that the corresponding labeling value R_{nj}^t is maximum. This statement is implied by the following stronger result, which in fact, is just a restatement of the fundamental principal of dynamic programming applied to this particular case:

Lemma 5.1: Let v_i be a vertex, $1 \leq i \leq n$, and λ_j a label on that vertex. Let $L_i(\lambda_j)$ be the set of all partial labelings which assign labels from Λ to vertices $v_1, v_2, \ldots, v_{i-1}$ and the specific label λ_j to vertex v_i in such a way so that the resulting labeling is consistent. For a given consistent partial labeling $\bar{\lambda}_i = \lambda^1\lambda^2 \cdots \lambda^{i-1}\lambda_j$, associate a value $c(\bar{\lambda}_i)$ which is the sum of the initial labeling values for the labels that participate in that labeling:

$$c(\bar{\lambda}_i) = c_{ij} + \sum_{k=1}^{i-1} c_k(\lambda^k)$$

Then the algorithm given above for information transmitted to the right, assigns to R_{ij}^t the value:

$$R_i^t(\lambda_j) = \max_{\bar{\lambda}_i \in L_i(\lambda_j)} \{ c(\bar{\lambda}_i) \}$$

for all $t \geq i$.

Proof: By induction. The assertion clearly holds for $i = 1$. Assume it also holds for $i-1$. Let $t \geq i-1$. Any consistent partial labeling, $\bar{\lambda}_i \in L_i(\lambda_j)$ with the label λ_j on vertex v_i is made up of a partial consistent labeling $\bar{\lambda}_{i-1} \in L_{i-1}(\lambda_{j'})$ along with the label λ_j on vertex v_i provided that the label $\lambda_{j'}$ assigned to vertex v_{i-1} is consistent with label λ_j on vertex v_i. By the induction hypothesis, for $t \geq i-1$ for every label $\lambda_{j'}$ on vertex v_{i-1} the value of $R_{i-1,j'}$ is given by:

$$R_{i-1,j'}^t = \max_{\bar{\lambda}_{i-1} \in L_{i-1}(\lambda_{j'})} \{ c(\bar{\lambda}_{i-1}) \mid \lambda' \text{ assigned to } v_{i-1} \}$$

So that at $t \geq i$,

$$\begin{aligned} R_t(\lambda) &= c_i^0(\lambda) + \max \{ R_{i-1,j'}^t \mid \lambda_{j'} \in \Lambda, r_{i-1,i}(\lambda',\lambda) = 1 \} \\ &= \max_{\bar{\lambda}_i \in L_i(\lambda_j)} \{ c_{ij}^0 + R_{ij'}^t \mid \lambda_{j'} \in \Lambda, r_{i-1,i}(\lambda',\lambda) = 1 \} \\ &= \max \{ c(\bar{\lambda}_i) \} \end{aligned}$$

In order to complete the algorithm both the process with data transmitted to the right, and the process with data transmitted to the left are run in parallel, as shown in figure 5.3 below. The output of the labeling process for a given label λ_j on a given vertex v_i, at each iteration t is given by the sum of the labeling value for data flow to the left plus the current labeling value for the network with data flow to the right minus the initial labeling value. Formally:

$$c_{ij}^t = L_{ij}^t + R_{ij}^t - c_{ij}^0 \qquad 5.5$$

In the case where the underlying graph is a path the process described by equations 5.2 through 5.5 will solve the problem with data initialized according to

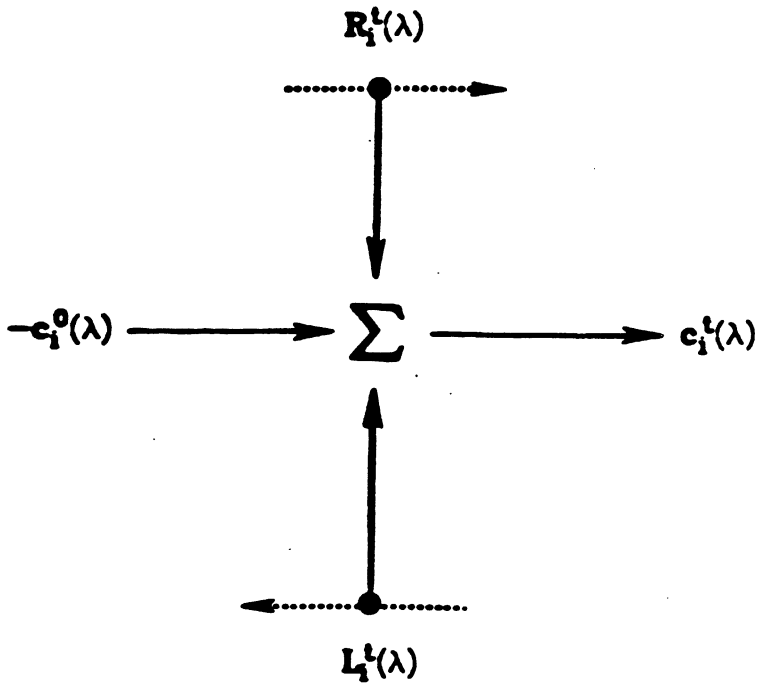


Figure 5.3: Updating process for the parallel iterative scheme derived from dynamic programming.

equations 5.1. That is, for any iteration $t \geq n$, we can choose, by local maxima selection, at each vertex that label such that the corresponding labeling value is maximum. The resulting labeling is (a) consistent, and (b) a solution to the graph labeling problem. Again, this statement follows from a stronger result which is given below:

Theorem 5.2: The process described by equations 5.2 through 5.5 will assign to each labeling value c_{ij}^t for label λ_j on vertex v_i for all v_i, λ_j the sum of the initial labeling values for the best consistent labeling in which the label λ_j on vertex v_i

participates.[2]

Proof: Let $\bar{\lambda} = \lambda^1\lambda^2 \cdots \lambda^n$ be the best consistent labeling in which the label λ^i on vertex v_i participates. Clearly then, the consistent partial labeling for vertices $v_1, v_2, \cdots, v_i$ given by $\lambda^1\lambda^2 \cdots \lambda^i$ is the best consistent partial labeling of these vertices in which the label λ^i on vertex v_i participates. Otherwise, there must exist another consistent partial labeling $\hat{\lambda} = \hat{\lambda}^1\hat{\lambda}^2 \cdots \hat{\lambda}^{i-1}\lambda_j$ such that $c(\hat{\lambda}) > c(\bar{\lambda})$. In this case, if we define

$$\tilde{\lambda} = \hat{\lambda}^1\hat{\lambda}^2 \cdots \hat{\lambda}^{i-1}\lambda^i\hat{\lambda}^{i+1}\hat{\lambda}^{i+2} \cdots \hat{\lambda}^n$$

then $c(\tilde{\lambda}) > c(\bar{\lambda})$. By lemma 5.1 above:

$$c(\bar{\lambda}_i) = R_{ij}^t \quad t \geq i.$$

By a similar argument, the consistent partial labeling of vertices $v_i, v_{i+1}, \ldots, v_n$ given by $\lambda_j\lambda^{i+1}\lambda^{i+2} \cdots \lambda^n$ is the best consistent partial labeling of these vertices,

$$c(\bar{\lambda}_i) = L_i^t(\lambda) \quad t \geq i-1.$$

Thus:

$$c_{ij}^t = R_{ij}^t + L_{ij}^t - c_{ij}^0$$

$$= c(\bar{\lambda})$$

The initial labeling value c_{ij}^0 must be subtracted since it occurs both of the terms R_{ij}^t and L_{ij}^t. The result then follows.

Corollary: In the case that there are no ties for the best consistent labeling the process described by equations 5.2 through 5.5 solves the problem.

---

[2]Note that this holds whether or not λ_j on vertex v_i participates in the solution to the problem or not.

5.2. Dual Approaches to the Graph Labeling Problem

Lagrange duality and price directed decomposition theory serve as a second building block in the enumerative schemes described below. In order to motivate the dual formulation, we consider first the naive approach to the labeling problem which is to simply choose that label at each vertex such that the initial labeling value is maximal. Obviously, if there are no constraints violated in the resulting labeling then the problem has been solved. However, in general, invalid labelings will result. One approach that could be taken in this case would be to penalize the labeling values associated with labels participating in constraint violations, thereby reducing them so that at the next iteration, another overall labeling would result. Hopefully, there would be fewer constraint violations in the new labeling. An iteration of the algorithms which will result from the application of these techniques will have the following form:

[1] Choose a labeling $\bar{\lambda} = \lambda^1\lambda^2 \cdots \lambda^n$ by selecting a label at each vertex such that the corresponding strength measure c_{ij} is maximal.

[2] If the resulting labeling is consistent, then stop. Otherwise,

[3] Reduce the labeling value c_{ij} for every vertex v_i such that the label λ^l participates in an invalid labeling pair $(\lambda^l,\lambda^{l'})$ on adjacent vertices v_i and $v_{i'}$.

[4] Go to 1.

The use of the theory of the Lagrange dual relaxation below is a systematic means by which this strategy can be implemented. The way in which this is done is to associate a non-negative dual variable $u_{iji'j'}$ with every inconsistent pair of labels $(\lambda_j,\lambda_{j'})$ on adjacent vertices v_i and $v_{i'}$. The value of a dual variable can be interpreted as the penalty assigned to the associated invalid pair of labels. In accordance with the theory, penalty values are multiplied by the indicator variables associated with those invalid labels resulting in the definition of an auxiliary function:

$$\varphi(\bar{x},\bar{u}) = \sum_{ij} c_{ij}\, x_{ij} - \sum_{iji'j' \in \mu} u_{iji'j'}\,(x_{ij} + x_{i'j'} - 1) \qquad 5.6$$

where μ is the set of all four-tuples, iji'j' corresponding to inconsistent pairs of labels $(\lambda_j,\lambda_{j'})$ on adjacent vertices v_i and $v_{i'}$. By rearranging terms, $\varphi(\bar{x},\bar{u})$ is represented as:

$$\varphi(\bar{x},\bar{u}) = \sum_{v_i \in V} \sum_{\lambda_j \in \Lambda} r_{ij}\, x_{ij} + \sum_{iji'j' \in \mu(ij)} u_{iji'j'} \qquad 5.7$$

where

$$r_{ij} = c_{ij} - \sum_{i'j' \in \mu(i,j)} u_{iji'j'}$$

is the current cost or payoff, relative to the dual vector $\bar{u}$, or simply, the current *relative cost*. Here $\mu(ij)$ is the set of all 4-tuple iji'j' corresponding to invalid pairs of labels in which the label λ_j on vertex v_i participates. The Lagrange dual form of the original problem [BaS79] is defined as:

$$\text{minimize} \qquad \Theta(\bar{u}), \quad \bar{u} \geq \bar{0} \qquad 5.8$$

$$\text{where:} \qquad \Theta(\bar{u}) = \sup_{\bar{x} \in X} \varphi(\bar{x},\bar{u}) \qquad 5.9$$

$$= \sup_{\bar{x} \in X} \Big\{ \sum_{ij} c_{ij}\, x_{ij} - \sum_{iji'j' \in \mu} u_{iji'j'} \cdot (x_{ij} + x_{i'j'} - 1) \Big\}$$

$$= \sup_{\bar{x} \in X} \Big\{ \sum_{ij} r_{ij}\, x_{ij} + \sum_{iji'j' \in \mu} u_{iji'j'} \Big\}$$

Here $\bar{x}$ is the vector of decision variables:

$$\bar{x} = (x_{11}, x_{12}, \ldots, x_{1m}, x_{21}, x_{22}, \ldots, x_{2m}, \ldots, x_{n1}, x_{n2}, \ldots, x_{nm})$$

and X is defined to be the set of all $\bar{x}$ which conform to constraints (3.3) and (3.5) of the original problem definition.

For any $\bar{u} \geq \bar{0}$, the weak law of duality guarantees that $\Theta(\bar{u}) \geq f(\bar{x})$, for all feasible $\bar{x}$. Thus, if $\Theta(\bar{u}) = f(\bar{x})$, for some pair $(\bar{u},\bar{x})$ then $\bar{u}$ must solve the dual problem and $\bar{x}$ must solve the primal. That is, those elements of $\bar{x}$ equal to 1 correspond to labels to be selected in the solution of the original problem. In this

case, by constraint (1) none of the constraint relations will be violated, and by (2) and the definition of X only 1 label will be selected at each vertex.

5.2.1. Example Problem

Using the example of section 3.3.2 of chapter 3, the Lagrange dual form of this problem is:

$$\text{minimize:} \quad \Theta(\bar{u}), \quad \bar{u} \geq \bar{0} \tag{5.10}$$

$$\text{where:} \quad \Theta(\bar{u}) = \sup_{\bar{x} \in X} \varphi(\bar{x},\bar{u})$$

and

$$\varphi(\bar{x},\bar{u}) = 12x_{11} + 9x_{12} + 3x_{13} + 10x_{21} + 8x_{22} + 2x_{23} -$$
$$u_{1121}(x_{11} + x_{21} - 1) - u_{1122}(x_{11} + x_{22} - 1) -$$
$$u_{1221}(x_{12} + x_{21} - 1)$$

So that

$$\varphi(\bar{x},\bar{u}) = (12 - u_{1121} - u_{1122})x_{11} + (9 - u_{1221})x_{12} + 3x_{11} +$$
$$(10 - u_{1121} - u_{1221})x_{21} + (8 - u_{1221})x_{22} + 2x_{23} +$$
$$u_{1121} + u_{1122} + u_{1221}.$$

As in equation 5.7 above, $\varphi(\bar{x},\bar{u})$ has the form:

$$\varphi(\bar{x},\bar{u}) = \sum_{i=1,2} \sum_{j=1,2,3} r_{ij}x_{ij} + u_{1121} + u_{1122} + u_{1221}$$

where

$$r_{ij} = c_{ij} - \sum_{i'j' \in \mu(i,j)} u_{i'j'}$$

In particular,

$r_{11} = 12 - u_1 - u_2$, for label λ_1 on vertex v_1,

$r_{12} = 9 - u_3$, for label λ_2 on vertex v_1,

$r_{13} = 3$, for label λ_3 on vertex v_1,

$r_{21} = 10 - u_1 - u_3$, for label λ_1 on vertex v_2,

$r_{22} = 8 - u_2$, for label λ_2 on vertex v_2,

$r_{13} = 2$, for label λ_3 on vertex v_2,

Here the set X is defined to be:

$$X = \{(1,0,0,1,0,0), (1,0,0,0,1,0), (1,0,0,0,0,1), (0,1,0,1,0,0), (0,1,0,0,1,0), (0,1,0,0,0,1), (0,0,1,1,0,0), (0,0,1,0,1,0), (0,0,1,0,0,1)\}.$$

For example, $\bar{x} = (0,1,0,0,0,1)$ corresponds to choosing label λ_2 on vertex v_1 and label λ_3 on vertex v_3.

5.2.2. Minimizing the Dual

The goal of the following algorithms are to minimize the dual function $\Theta(\bar{u})$ subject to $\bar{u}$ non-negative. The objective, nominally, is to derive a solution to the original problem by finding the vector $\bar{x} \in X$ corresponding to the point $\bar{u}$ at which the function $\Theta(\bar{u})$ is minimized. However, this strategy will not always work. Because of the discrete (i.e. non-convex) nature of the primal space, a duality gap will exist in the general case, so that the minimum value of the dual function will be strictly greater than the maximum value of the primal. When this occurs, it will be impossible to derive a primal feasible solution from the current point in the dual space. The process of minimizing the dual function is still of interest, however, since the minimum value of the dual can always be used as an upper bound on the best solution of a current candidate problem in a branch and bound approach.

Standard algorithms for minimizing the dual are discussed in [BaS79]. In general, these methods are directed towards finding the steepest descent direction at each iteration, and do not address the issue of the decentralization of the computation. Since the issue of decentralized computation is important to

the goal of implementing as much of the solution as possible in parallel hardware, we do not make specific use of these techniques, but rather concentrate on showing how and when a descent direction can be determined on a local basis. In order to do this, we make some observation about the behavior of $\varphi(\bar{x},\bar{u})$ and hence $\Theta(\bar{u})$ as a function $\bar{u}$.

The value assigned to the function $\Theta(\bar{u}) = \max_{\bar{x}\in X} \varphi(\bar{x},\bar{u})$ for a given $\bar{u}$ results from two factors: one which generates a set, S, of values for the function $\varphi(\bar{x},\bar{u}) = \varphi_{\bar{x}}(\bar{u})$ for every $\bar{x} \in X$ and the other which selects the maximum value of the elements of S. Alternatively, we may, for all $\bar{x}\in X$ consider the set, Φ, of functions $\varphi_{\bar{x}}(\bar{u})$ and examine those closed intervals of R^m on which a particular element of Φ is selected. Because each $\varphi_{\bar{x}}(\bar{u}) \in \Phi$ is a linear function of $\bar{u}$, $\Theta(\bar{u})$ is piecewise linear and from the theory, it is convex.

For the sake of further discussion, we consider $\varphi(\bar{x},\bar{u})$ as the sum of two functions α and β:

$$\varphi(\bar{x},\bar{u}) = \alpha(\bar{x},\bar{u}) + \beta(\bar{u})$$

with

$$\alpha_{\bar{x}}(\bar{u}) = \alpha(\bar{x},\bar{u}) = \sum_{v_i\in V} \sum_{\lambda_j\in\Lambda} r_{ij}\, x_{ij}$$

and

$$\beta(\bar{u}) = \sum_{ijj'i'\in\mu} u_{ijj'i'}.$$

Note that for all $\bar{x}$

$$\frac{\partial\, \beta(\bar{u})}{\partial\, u_{ijj'i'}} = 1$$

for all components $u_{ijj'i'}$ of the dual vector.

Let $X(\bar{u})$ be the set of all $\hat{x} \in X$ such that

$$\varphi(\hat{x},\bar{u}) = \max_{\bar{x}\in X}\varphi(\bar{x},\bar{u})$$

for a given fixed value of $\bar{u}$. Let $\hat{x} \in X(\bar{u})$. Define $g:R^{nm} \rightarrow R^p$ to be the vector valued function constructed from the inequality constraints of the original problem

(p is the total number of constraints). That is, if $x_{i_r j_r} + x_{i'_r j'_r} - 1 \leq 0$ is the r^{th} constraint (according to some arbitrary ordering), then $g_r(x) = x_{i_r j_r} + x_{i'_r j'_r} - 1$. A well-known result states that $\hat{u} = g(\hat{x})$ is a subgradient of Θ at $\bar{u}$. If $| X(\bar{u}) | = 1$ then $\hat{u} = g(\hat{x})$ is the gradient of Θ at $\bar{u}$. Note that $\beta(\bar{u})$ is independent of $\bar{x}$, so that the derivation of $X(\bar{u})$ can be performed completely by inspection of $\alpha(\bar{x},\bar{u})$. In particular, if Λ_i is defined to be the set of all labels λ_j on vertex v_i such that the corresponding reduced value r_{ij} is maximal, then it is not difficult to see that

$$X(\bar{u}) = \Lambda_1 \times \Lambda_2 \times \cdots \Lambda_n.$$

We start by considering the generation of the descent directions for the case where the underlying graph contains only two vertices, say v_i and v_i', and use example problem 5.2.1 for the sake of illustration. Table 5.1 shows $\varphi_{\bar{x}}(\bar{u})$ for $\bar{x} \in X$ as a function of $\bar{u}$ for this example. Assume $\bar{u} = (u_{1121}, u_{1122}, u_{1221}) = (0, 0, 0)$. Here, $\Theta(\bar{u}) = 12 + 10 = 22$. This occurs at

$$X(\bar{u}) = \{ \hat{x} \} = \{ (1,0,0,1,0,0) \}.$$

Note that $\hat{x}$ is not primal feasible (constraint 1a is violated). Because $|X(u)|$ contains a single element, $g(\hat{x}) = (1, 0, 0)$ is the gradient of $\Theta(\bar{u})$ at $\bar{u} = (0, 0, 0)$.

A descent step is taken by increasing u_1 from 0 to 2. At $\bar{u} = (2,0,0)$,

$$X(\bar{u}) = \{\hat{x}_1, \hat{x}_2\} = \{(1,0,0,1,0,0), (1,0,0,0,1,0)\}$$

and $\Theta(\bar{u}) = 20$. The reason that a descent step is generated by increasing u_{1111} is because such an increase causes a decrease in both $\max_{j=1,2,3} \{r_{ij}\}$ and $\max_{j'=1;2,3} \{r_{i'j'}\}$. That is, the relative cost of the best choice on both vertex v_i and v_i' decrease so that

x	$\varphi_x(u)$
(1,0,0,1,0,0)	$22 - u_{1121}$
(1,0,0,0,1,0)	$20 + u_{1121} - u_{1122}$
(1,0,0,0,0,1)	$14 + u_{1221}$
(0,1,0,1,0,0)	$19 - u_{1221} + u_{1122}$
(0,1,0,0,1,0)	$17 + u_{1121}$
(0,1,0,0,0,1)	$11 + u_{1121} + u_{1122}$
(0,0,1,1,0,0)	$13 + u_{1122}$
(0,0,1,0,1,0)	$11 + u_{1121} + u_{1121}$
(0,0,1,0,0,1)	$5 + u_{1121} + u_{1221} + u_{1122}$

Table 5.1: $\varphi_{\bar{x}}(\bar{u})$ for all possible $\bar{x} \in X$ for example problem 5.2.1.

$$\frac{\partial\, \alpha(\bar{x},\bar{u})}{\partial\, u_{iji'j'}} = -2$$

Therefore, even though the value of $\beta(\bar{u})$ increases from 0 to 2, the value of $\alpha(\bar{x},\bar{u})$, for $\bar{x}$ = (1,0,0,1,0,0) decreases from 22 to 18, so the net chance is a decreases of 2.

Increasing u_{1111} beyond 2, say to a value of $2 + \Delta u_{1111}$, for $0 \le \Delta u_{1111} \le 1$ causes a net change of Δu_{1111} in $\beta(\bar{u})$ and $-\Delta u_{1111}$ in $\alpha(\bar{u})$. For $2 < u_{1111} < 3$,

$$\frac{\partial\, \alpha(\bar{x},\bar{u})}{\partial\, u_{1111}} = -1$$

since the maximum relative cost on vertex v_1 which is 10, does not change, while the relative cost on vertex v_2 decreases by a factor of Δu_{1111}. Thus the value of $\Theta(\bar{u})$ is independent of u_{1111} for $\bar{u} = (u_{1111}, 0, 0)$ with $2 \le u_{1111} \le 3$. For $u_{1111} > 3$,

$$\frac{\partial\, \alpha(\bar{x},\bar{u})}{\partial\, u_{1111}} = 0$$

and

$$\frac{\partial\, \beta(\bar{u})}{\partial\, u_{1111}} = 1$$

so that

$$\frac{\partial\, \beta(\bar{u})}{\partial\, u_{1111}} = 1$$

This analysis is summarized in the graph of figure 5.4, which shows $\Theta(\bar{u})$ as a function of u_{1111} for $u_{1122} = u_{1221} = 0$, as well as the functions in the set Φ which are "selected" in each of the regions $0 < u_{11} < 2$, $2 < u_{11} < 3$, and $3 < u_{11}$.

The analysis given above can be easily extended to account for the effect on changes in $\bar{u}$ on $\Theta(\bar{u})$ for other points in the dual space. Let Λ_i be the set of labels λ_j on vertex v_i such that the relative cost $r_{ij} = \max_{k=1,\ldots,m} \{r_{ik}\}$. That is, Λ_i

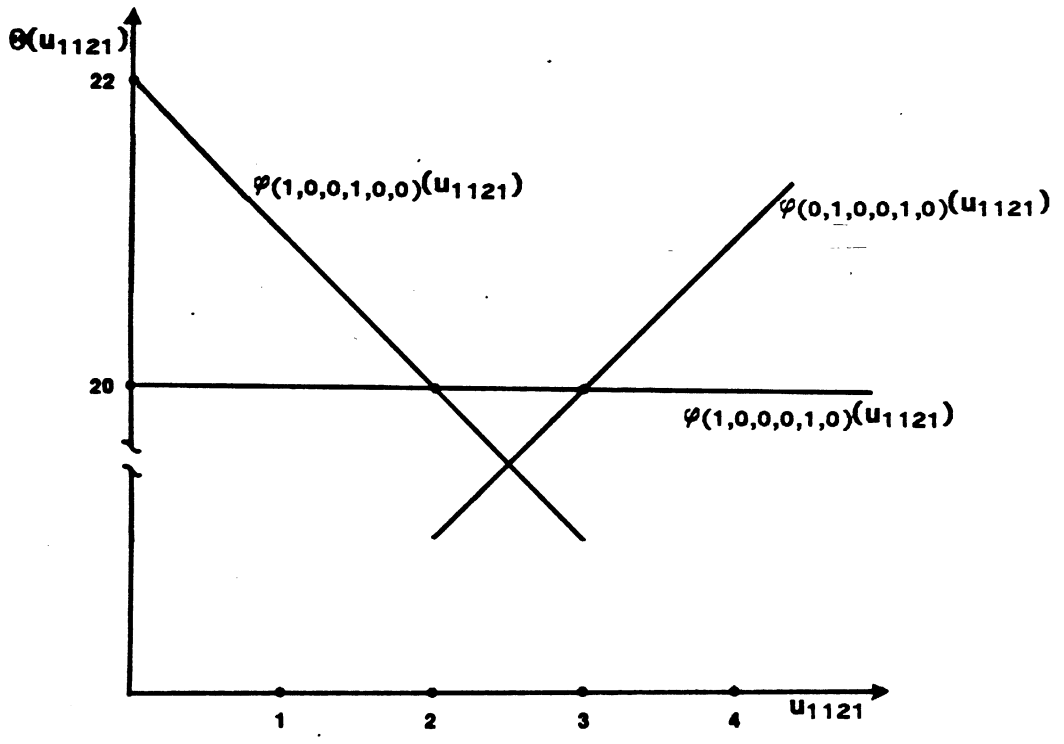


Figure 5.4: $\Theta(\bar{u})$ as a function of u_{1111} for u_{1122} and u_{1221} equal to zero.

is the set of best choices to be made at vertex v_i. Let u_r be a component of the dual vector whose associated constraint is $x_{i_r j_r} + x_{i'_r j'_r} - 1 \leq 0$. Using the same reasoning as above, it can easily be concluded that, for example:

[1] If $\lambda_{i_r j_r} \notin \Lambda_i$ and $\lambda_{i'_r j'_r} \notin \Lambda_{i'}$ then: (1) increasing the value of u_r causes the value of the dual to increase, and (2) decreasing the value of u_r cases the value of the dual to decrease.

[2] If $\lambda_{i_r j_r} \in \Lambda_i$ and $\lambda_{i'_r j'_r} \in \Lambda_{i'}$ and if $|\Lambda_i| > 1$ and $|\Lambda_{i'}| > 1$ then: (1) increasing u_r results in a increase in the value of the dual, and (2) decreasing u_r also causes an increase in the value of the dual.

[3] If $\lambda_{i_r j_r} \in \Lambda_i$ and $\lambda_{i'_r j'_r} \in \Lambda_{i'}$ and if $|\Lambda_i| = 1$ and $|\Lambda_{i'}| > 1$ then: (1) increasing u_r results in no change in the value of the dual, and (2) decreasing u_r causes an increase in the value of the dual.

[4] If $\lambda_{i_r j_r} \notin \Lambda_i$ and $\lambda_{i'_r j'_r} \in \Lambda_{i'}$ and if $|\Lambda_i| = 1$ and $|\Lambda_{i'}| > 1$ then: (1) increasing u_r results in an increase in the value of the dual, and (2) decreasing u_r causes no change in the value of the dual.

The effect of increasing (respectively, decreasing) the value of the component u_r for each of the set of 16 possible situations for the two vertex case are summarized in table 5.2a (respectively 5.2b). This information can then be used in the following descent algorithm for a graph labeling problem defined on an underlying graph with two verices:

Algorithm 5.1

[1] For all vertices, $v_i \in V$, let Λ_i be the set of all labels $\lambda_j \in \Lambda$ such that the associated current relative payoffs r_{ij} are maximal.

[2] If a feasible, or consistent labeling exists among the current labeling $L = (\Lambda_i, \Lambda_{i'})$ then output that labeling as a solution and terminate.

$\lambda_{j_r} \in \Lambda_{i_r}$	$\lvert \Lambda_{i_r} \rvert = 1$	$\lambda_{j'_r} \in \Lambda_{i'_r}$	$\lvert \Lambda_{i'_r} \rvert = 1$	change in $\Theta(u_r)$
no	no	no	no	increase
no	no	no	yes	increase
no	no	yes	no	increase
no	no	yes	yes	no change
no	yes	no	no	increase
no	yes	no	yes	increase
no	yes	yes	no	increase
no	yes	yes	yes	no change
yes	no	no	no	increase
yes	no	no	yes	increase
yes	no	yes	no	increase
yes	no	yes	yes	no change
yes	yes	no	no	no change
yes	yes	no	yes	no change
yes	yes	yes	no	no change
yes	yes	yes	yes	decrease

Table 5.2a: Change in $\Theta(u_r)$ as a function of an **increase** in the dual variable u_r depending on the various conditions of the labeling on adjacent vertices v_{i_r} and $v_{i'_r}$.

$\lambda_{j_r} \in \Lambda_{i_r}$	$\lvert \Lambda_{i_r} \rvert = 1$	$\lambda_{j'_r} \in \Lambda_{i'_r}$	$\lvert \Lambda_{i'_r} \rvert = 1$	change in $\Theta(u_r)$
no	no	no	no	decrease
no	no	no	yes	decrease
no	no	yes	no	no change
no	no	yes	yes	no change
no	yes	no	no	decrease
no	yes	no	yes	decrease
no	yes	yes	no	no change
no	yes	yes	yes	no change
yes	no	no	no	no change
yes	no	no	yes	no change
yes	no	yes	no	increase
yes	no	yes	yes	increase
yes	yes	no	no	no change
yes	yes	no	yes	no change
yes	yes	yes	no	increase
yes	yes	yes	yes	increase

Table 5.2b: Change in $\Theta(u_r)$ as a function of a **decrease** in the dual variable u_r depending on the various conditions of the labeling on adjacent vertices v_{i_r} and $v_{i'_r}$. Note: $u_r > 0$ is assumed.

[3] Use tables 5.2a and 5.2b to determine if a descent direction exists. If not, go to [11].

[4] If a descent direction exists which involves decreasing the value of some u_r go to [7].

[5] Let λ_j be the best choice at vertex v_i and let $\lambda_{j'}$ be the best choice at vertex $v_{i'}$. Let $m_i = r_{ij}$ and let $m_{i'} = r_{i'j'}$.

[6] Let $\bar{m}_i$ be the second best relative cost at vertex v_i and $\bar{m}_{i'}$ be the second best relative cost at vertex $v_{i'}$. Define

$$\delta_i = m_i - \bar{m}_i, \quad \delta_{i'} = m_{i'} - \bar{m}_{i'}$$

and

$$\delta = \min \{\delta_i, \delta_{i'}\}$$

[6] Set $r_{ij} \leftarrow r_{ij} - \delta$ and $r_{i'j'} \leftarrow r_{i'j'} - \delta$, that is, add δ to u_r and go to step [1].

[7] Let λ_{ij} and let $\lambda_{i'j'}$ be the label associated with u_r on vertex v_i and $v_{i'}$ respectively.

[8] Let λ_k be the best choice at vertex v_i and let $\lambda_{k'}$ be the best choice at vertex $v_{i'}$. Let $m_i = r_{ik}$ and $m_{i'} = r_{i'k'}$.

[9] Set

$$\delta_i = m_i - \bar{m}_i, \quad \delta_{i'} = m_{i'} - \bar{m}_{i'}$$

and

$$\delta = \min \{\delta_i, \delta_{i'}\}$$

[10] Set $r_{ij} \leftarrow r_{ij} + \delta$ and $r_{i'j'} \leftarrow r_{i'j'} + \delta$, that is, subtract δ from u_r and go to step [1].

[11] A duality gap exists. Terminate with no solution.

5.2.3. General Algorithm for PMI Constraints

The algorithm given above can easily be extended to minimizing the dual problem formulated in terms of PMI constraints. This extension requires modifying the conditions of tables 5.2a and 5.2b somewhat. In particular, there now may be several labels associated with a given dual variable u_r on each vertex. Thus, for a given set, Λ_i of best choices at a given vertex v_i, several situations are significant and must be distinguished depending on whether or not the clique associated with u_r:

(1) covers all of the labels in Λ_i,

(2) contains at least one but not all of the labels in Λ_i, or

(3) contains no of the labels in Λ_i.

Again, using the analysis of the previous section, a decision chart giving conditions under which a descent (or ascent) direction for the dual exists can be constructed. This is given in tables 5.3a and 5.3b. A descent algorithm for the problem formulated under PMI constraints is easily derived by making the necessary modifications to algorithm 5.1. Note that in general there will be fewer PMI constraints than original constraints between a pair of vertices, so that by bit encoding indicator vectors for the PMI sets a great deal of computational efficiency can be gained in reducing the amount of search required to verify the various conditions found in the decision tables.

There is another descent algorithm based on PMI constraints, which is probably not as robust as the approach described above, but which is useful in generating upper bounds because of its relative simplicity. Given a PMI constraint, such as

$$\sum_{ij \in K} x_{ij} \leq 1 \qquad 5.11$$

where K is some set of vertices in the product graph, one can arbitrarily add

elements of Λ_{i_r} covered by u_r	elements of $\Lambda_{i'_r}$ covered by u_r	change in $\Theta(u_r)$
none	none	increase
none	some[1]	increase
none	all	no change
some	none	increase
some	some	increase
some	all	no change
all	none	no change
all	some	no change
all	all	decrease

Table 5.3a: Change in $\Theta(u_r)$ as a function of an **increase** in the dual variable u_r depending on the various conditions of the labeling on adjacent vertices v_{i_r} and $v_{i'_r}$.

[1] "some" refers the condition where at least one but not all labels are covered.

elements of Λ_{i_r} covered by u_r	elements of $\Lambda_{i'_r}$ covered by u_r	change in $\Theta(u_r)$
none	none	decrease
none	some[1]	no change
none	all	no change
some	none	no change
some	some	increase
some	all	increase
all	none	no change
all	some	increase
all	all	increase

Table 5.3b: Change in $\Theta(u_r)$ as a function of a **decrease** in the dual variable u_r depending on the various conditions of the labeling on adjacent vertices v_{i_r} and $v_{i'_r}$. Note: $u_r > 0$ is assumed.

[1] "some" refers the condition where at least one but not all labels are covered.

constraints of the form

$$\sum_{ij \in S} x_{ij} \leq 1 \qquad 5.12$$

for all subsets $S \subseteq K$. The constraints of equation 5.12 will not change the feasible region for the problem formulated in terms of PMI constraints and are thus valid inequalities. However, if one views the problem conceptually as being formulated in this manner, then the following algorithm, in which the dual variables are not maintained explicitly, can be used:

Algorithm 5.2

[1] For all vertices, $v_i \in V$, let M_i be the set of all labels $\lambda_j \in \Lambda$, such that the associated current relative payoffs r_{ij} are maximal.

[2] If a feasible, or consistent labeling exists among the current labeling $L = (M_i, M_i')$, then output that labeling as a solution and terminate.

[3] If the sets M_i and M_i' cannot be covered by a PMI constraint, then stop.

[4] Let m_i be the value of the current payoffs for elements of M_i and let m_i' be the value of the current payoffs for elements of M_i'.

[5] Let $\bar{m}_i$ be the second best relative cost at vertex v_i and let $\bar{m}_i'$ be the second best relative cost at vertex v_i'. Define

$$\delta_i = m_i - \bar{m}_i, \quad \delta_i' = m_i' - \bar{m}_i'$$

and

$$\delta = \min \{\delta_i, \delta_i'\}$$

[6] Set $r_{ij} \leftarrow r_{ij} - \delta$ for all $\lambda_{ij} \in M_i$ and $r_{i'j'} \leftarrow r_{i'j'} - \delta$, for all $\lambda_{i'j'} \in M_i'$.

[7] Go to [1].

This algorithm is useful in finding local minima starting from the point $u = (0, 0, \cdots, 0)$ only, since the number of dual variables is prohibitively large and would be difficult to maintain. The relative simplicity is in the fact that there

are much fewer conditions to check in determining a local descent direction.

5.2.4. Extension to the General Graph

Algorithm 5.1 and its equivalent form for PMI constraints, as well as algorithm 5.2 can be extended to the case where the underlying graph contains more than two vertices. In this case, an iteration of the algorithm involves a single "pass" over the entire graph, where a pass involves separate consideration of every pair of adjacent vertices (edges) in the graph. In each pass any set of disjoint edges (stable set) in the graph can be considered simultaneously, hence the partial decomposition of the computation. It should be noted that the minima found by this process will not necessary correspond to the minima of the dual function itself. In particular, it may be possible in the case of algorithm 5.1, and its equivalent PMI formulation that a situation is achieved in which no local descent direction exists but yet for which a sequence of local non-ascent, non-descent (degenerate pivots) exists which would eventually lead to a global descent direction.

Consider the constraint network of figure 5.5, for example, which shows some of the cliques corresponding to PMI sets as being circled. Assume that the labels corresponding to these PMI sets all take on the same value which is the maximal value at that vertex. From the rules of table 5.3a and 5.3b no local descent direction exists. All labels at the n vertices which take on the maximum value are covered by the $n-1$ cliques, so that a global descent direction exists if the $n-1$ associated dual variables are increased simultaneously. Furthermore, if one were to perform the $n-1$ degenerate pivots corresponding to first increasing the value of the dual variable associated with the first clique, then increasing the value of the dual variable corresponding to the second clique and so forth, a descent direction would eventually be found.

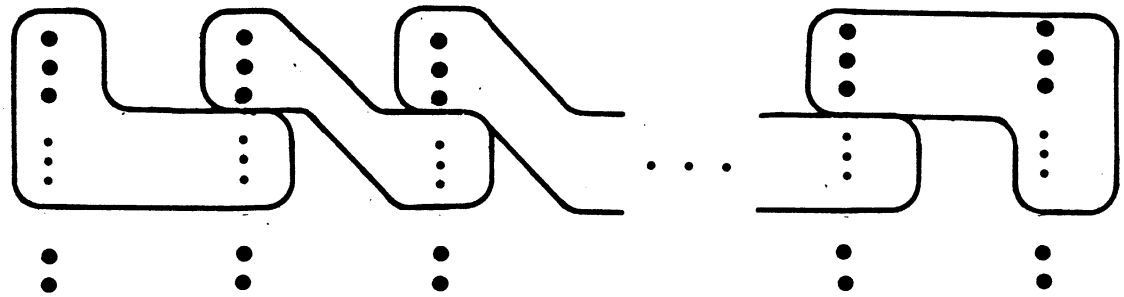


Figure 5.5: Covering Cliques With Global Descent Direction.

5.3. Problems with Special Structure

In the previous section, approaches based on both dynamic and integer programming were introduced. In this section, an approach to generating upper bounds based on a combination of the integer and dynamic programming techniques will be presented. This approach will apply only to the case where the underlying graph presents a structure which gives it a set of significant major directions, for example, a square image with each pixel considered to be adjacent to its four or eight immediate neighbors, or a hexagonal grid.

In order to develop this hybrid approach, we start by making some observations about both the Lagrange dual and dynamic programming approaches described in the previous sections.

With respect to the integer programming approach:

[I1] The minimization algorithm requires knowledge only of those labels with the best and second best relative payoff at each iteration.

[I2] The output gives information only as to which label(s) to choose at each vertex.

With respect to the dynamic programming approach when the underlying graph is a path:

[D1] The dynamic programming algorithm requires full knowledge of every label on each vertex at every iteration.

[D2] The algorithm responds to changes in input without re-initialization.

[D3] The algorithm assigns to every label on each vertex, the sum of the initial labeling values for the maximum consistent labeling that contains that label, this being the case regardless of whether or not that label participates in the maximum consistent labeling.

[D4] If a given label λ_j on a vertex v_i participates in the best consistent labeling along the path, then it is assigned a labeling value c_{ij}, equal to the sum of the initial labeling values of those labels (one for each vertex), which participate in that optimal labeling. Thus, for every label which participates in the optimal consistent labeling, the output labeling value will be the same.

5.3.1. Hybrid Formulation: Columns vs. Columns Case

In order to develop this approach, consider the case of a square raster, with each grid element considered to be connected to its four immediate neighbors. Thus, this graph has two major directions: one corresponding to the rows, and the other corresponding to the columns. Let $v_{r,c}$ denote the vertex corresponding to the pixel which is at the intersection of row r and column c.

By initially ignoring the constraints between labels along the rows, we may start by applying the dynamic programming approach independently along each column, as shown in figure 5.6 below. From the results of section 5.1, the labeling decisions will be consistent from row to row along each column, assuming there are no ties for the best consistent labeling along that column. That is, a label chosen for vertex $v_{r,c}$ will be consistent with the label chosen for the vertex $v_{r-1,c}$ above it, and with the label chosen for the vertex $v_{r+1,c}$, below it. If the labeling decisions are also consistent from column to column along each row, then the problem has been solved for the given input. However, in general, this will not be the case: the label chosen for vertex $v_{r,c}$ will be inconsistent with the label chosen for vertex $v_{r,c-1}$ to the left, or the label chosen for vertex $v_{r,c+1}$ to the right, for at least a few vertices in the graph.

An alternate view towards the problem is, therefore, one of finding a *labeling* for each column, such that each labeling is "consistent" with labelings on "adjacent" columns. The result is the definition of an auxiliary or "reduced" constraint network in which each vertex is identified with a column in the original (or *basic* problem as it will be referred to here) and each "macro" label corresponds to a consistent labeling of that column in the original constraint network (refer to figure 5.7). Note that, in this case, the reduced constraint network is a path, so that one might be tempted to try to solve the reduced problem using the dynamic programming approach. Unfortunately, by observation D2, this would require explicit knowledge of the labeling values associated with *all* macro labels, at each vertex, and this information is not available within the context of the particular problem. However, what is available at each vertex in the reduced graph, and by observation D3, at every vertex in the original graph, is the labeling value associated with the maximal macro label. That macro label is simply that consistent *labeling* such that the sum of the initial labeling value is maximal. This then suggests that enough information may be available to apply

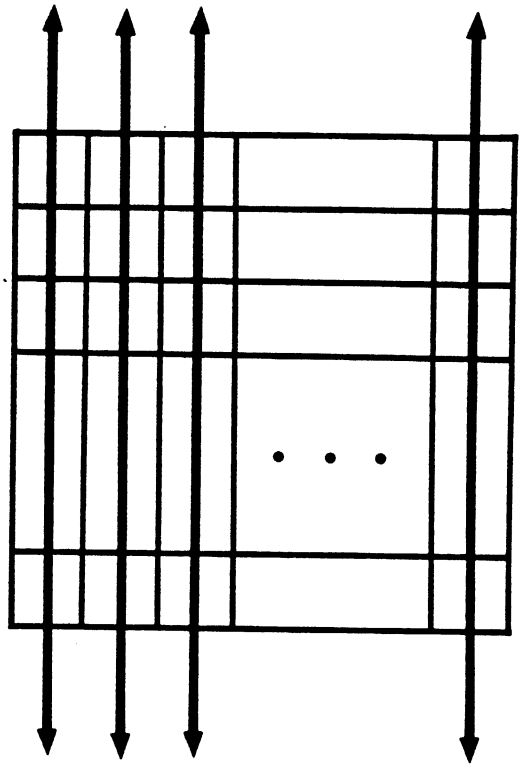


Figure 5.6: Major directions in the raster for the hybrid approach.

the integer programming approach to this problem.

Let Λ_c be the set of macro labels associated with each vertex in the reduced constraint network. That is, Λ_c is the set of consistent labelings for each column in the basic problem), which for simplicity are assumed here to be the same for all vertices. Denote the k^{th} macro label on vertex (column) j by $\sigma_{jk} = \lambda^1\lambda^2 \cdots \lambda_n$ where λ^i is the label corresponding to row i for vertex (column) j, and let x_{jk} be the associated indicator variable. Let $C_{jk} = \sum_{i=1}^{n} c_j(\lambda^i)$ be the initial cost associated with macro label σ_{jk}. The integer programming formulation of the reduced problem, which is equivalent to the original problem statement is given as:

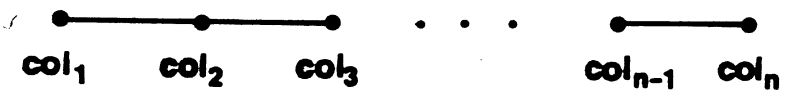


Figure 5.7 : Column reduced constraint network for the hybrid approach.

$$\text{maximize:} \quad \sum_{j=1}^{n} \sum_{\sigma_k \in \Lambda_C} x_{jk} C_{jk} \qquad 5.13$$

$$\text{subject to:} \quad x_{jk} + x_{j'k'} \leq 1, \qquad 5.14$$

for every infeasible pair of macro labels $(\sigma_{jk}, \sigma_{j'k'})$ on adjacent columns col_j and $col_{j'}$,

$$\sum_{\sigma_k \in \Lambda_C} x_{jk} = 1, \quad j=1, \cdots, n, \qquad 5.15$$

$$x_{jk} \in \{0,1\}. \qquad 5.16$$

As before (see section 5.1), constraint (5.14) serves to guarantee that a pair of inconsistent macro labels σ_k and σ_k' are not simultaneously assigned to adjacent columns and constraint (5.15) serves to guarantee that only one macro label is assigned to each column.

Consider a macro label $\sigma_k = \lambda^1\lambda^2 \cdots \lambda^n$ assigned to column j, and a macro label $\sigma_{k'} = \lambda^{1'}\lambda^{2'} \cdots \lambda^{n'}$ assigned to the adjacent column j+1. Assume that σ_{jk} is inconsistent with $\sigma_{j+1,k'}$. Then there must exist at least one row in the raster of the original problem, say i, such that λ^i is inconsistent with $\lambda^{i'}$ in the basic problem. That is, the local inconsistencies in the basic problem correspond to inconsistencies in the reduced problem. Now let $S_{ij}(\lambda^i)$ be the set of all macro labels $\sigma_{jk} \in \Lambda_C$ associated with column j in which the basic label λ^i on vertex $v_{i,j}$ participates (vertex $v_{i,j}$ occurs at the intersection of row i and column j) and let $S_{i,j+1}(\lambda^{i'})$ be the set of all macro labels $\sigma_{j+1,k'} \in \Lambda_C$ associated with column j+1 in which the basic label $\lambda^{i'}$ on vertex $v_{i,j+1}$ participates. Since no macro label $\sigma_{jk} \in S_{ij}(\lambda^i)$ is consistent with a macro label $\sigma_{j+1,k'} \in S_{ij'}(\lambda^{i'})$ we have

$$\sum_{\sigma_{jk} \in S_{ij}} x_{jk} + \sum_{\sigma_{jk'} \in S_{ij'}} x_{jk} \leq 1 \qquad 5.17$$

as a valid inequality (refer to figure 5.8). Formulating the constraints in this manner results in a one-to-one correspondence between the constraints in the reduced problem and the constraints between adjacent vertices from column to column along a given row in the original problem. Furthermore, the descent step taken by increasing the value of a dual variable associated with a constraint of the form of equation 5.17 is equivalent to reducing the *original* values associated with the inconsistent pair of labels $(\lambda^i,\lambda^{i'})$. This then motivates the following hybrid approach for the two column case.

Algorithm 5.3

[1] Apply the dynamic programming approach to the original labeling values independently along each column.

[2] Using the updated labeling values, choose the best label at each vertex along each column.

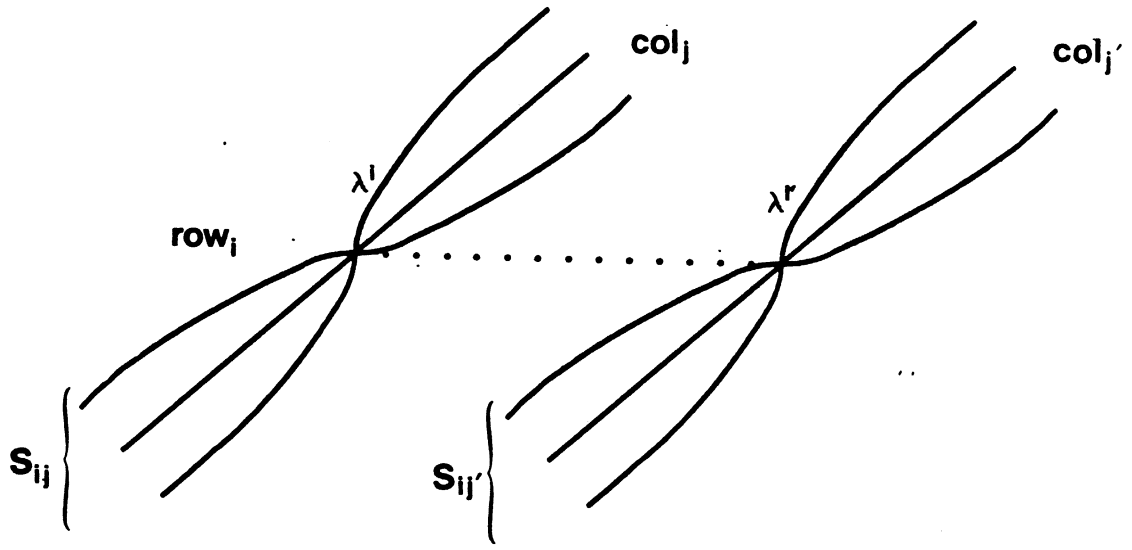


Figure 5.8 : Constraint generated by the set of macro labels on adjacent columns which have labels inconsistent across an intersecting row. Label λ^i on vertex $v_{i,j}$ is locally inconsistent with label $\lambda^{i'}$ on vertex $v_{i,j+1}$.

[3] If each label thus chosen is consistent with the label chosen on the other column for each row, then exit with a solution, or at least an upper bound (refer to the further discussion below).

[4] Consider differences between the best and second best of the updated labeling values along those rows in which the individual labeling choices are inconsistent. Let i be the row in which the minimum of these values occurs. Let this difference be equal to δ.

[5] Subtract δ from the *original* labeling values (that is, add δ to the dual variable associated with that constraint) for the vertices on that row, for both columns.

Go to [1].

The extension of this approach both to the multi-column case and to the case where PMI constraints are used (on a local basis, that is along individual rows) is straightforward. In the case of a graph labeling problem where the underlying graph is a path, it is always possible to chose an unambiguous consistent labeling from a consistent (ambiguous) labeling. It is important to note, however, that in the reduced constraint network approach described above, local consistency (in the original problem) of an unambiguous labeling *does not* guarantee the existence of a globally consistent macro labeling. In order for this to be the case, the macro labels would have to be explicitly enumerated. It is for this reason that algorithm 5.3 will only guarantee an upper bound on the best consistent labeling in the reduced problem.

5.3.2. Hybrid Formulation, Rows vs Columns Case

As an extension of the ideas presented in the previous section, an alternate means by which the dynamic and integer programming approaches can be combined, which involves the application of the dynamic programming approach independently along rows *and* columns, can be developed. As before, a reduced constraint network will result. In the reduced constraint network, there is a vertex for every row and every column of the original problem. The constraints from row to row and from column to column as well as the procedure for generating descent steps are straightforward extensions of those for the case discussed above. In addition, at every vertex in the original problem, there will be an associated constraint between the row and column that intersect at that vertex. This occurs because the dynamic programming algorithm applied to a row is required to generate the same label selection at the vertex as the dynamic programming algorithm applied to the intersecting column. The reduced constraint network for this case is depicted in figure 5.9.

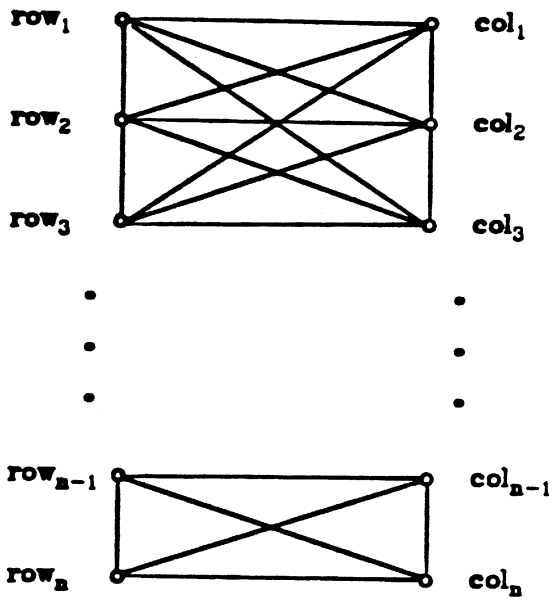


Figure 5.9 : Reduced constraint network for the row vs. column case

Let Λ_C denote the set of all macro labels associated with each column and Λ_R denote the set of macro labels associated with each row. The costs for the macro labels on each row and column will be defined in a manner analogous to that given in the previous section. Let $SC_{ij}(\lambda^l)$ be the set of macro-labels for row i in which the label λ^l on vertex $v_{i,j}$ participates. Let $SR_{ij}(\lambda^l)$ be the set of macro-labels for column j in which the label λ^l on vertex $v_{i,j}$ participates. Associate with the k^{th} macro-label ρ_{ik} on row i a zero-one decision variable $x_\rho(i,k)$. Likewise, associate with the k^{th} macro-label χ_{jk} on column j a zero-one decision variable $x_\chi(j,k)$. The equivalent integer programming statement of the graph labeling problem is then given as:

$$\text{maximize: } \frac{1}{2}\left\{ \sum_{r \in rows} \sum_{\rho_{rk} \in \Lambda_R} x_\rho(r,k)\, C_{rk} + \sum_{c \in cols} \sum_{\chi_{ck} \in \Lambda_C} x_\chi(c,k)\, C_{ck} \right\}$$

associate with the k^{th} macro-label χ_{jk} on column j a zero-one decision variable $x_\chi(j,k)$. The equivalent integer programming statement of the graph labeling problem is then given as:

$$\text{maximize: } \frac{1}{2}\left\{ \sum_{r\in rows} \sum_{\rho_{rk}\in\Lambda_R} x_\rho(r,k)\, C_{rk} + \sum_{c\in cols} \sum_{\chi_{ck}\in\Lambda_C} x_\chi(c,k)\, C_{ck} \right\}$$

$$\text{subject to: } \sum_{\rho_{rk}\in SR_{rc}(\lambda^i)} x_\rho(r,i) = \sum_{\chi_{ck'}\in SC_{rc}(\lambda^i)} x_\chi(c,k') \qquad 5.18$$

whenever macro-label ρ_{rk} on row r uses the same primitive label, λ^i, as macro label $\chi_{c,k'}$ on column c for the vertex at the intersection of row r and column c.

$$x_\rho(i,k) + x_\rho(i',k') \le 1,$$

for every infeasible pair of macro labels $(\rho_{ik},\rho_{i'k'})$ on adjacent rows row_i and row_i',

$$x_\chi(j,k) + x_\chi(j',k') \le 1, \qquad 5.20$$

for every infeasible pair of macro labels $(\chi_{jk},\chi_{j'k'})$ on adjacent columns col_j and col_j',

$$\sum_{\rho_{ik}\in\Lambda_R} x_\rho(i,k) = 1 \qquad \text{for all rows,} \qquad 5.21$$

$$\sum_{\chi_{ik}\in\Lambda_R} x_\chi(i,k) = 1 \qquad \text{for all columns,} \qquad 5.22$$

As before, descent steps can be taken for the constraints between adjacent columns and/or rows simultaneously, so long as the edges in the underlying graph (the reduced constraint network of figure 5.12) are

a simple means for generating descent steps, consider the case of a graph with a single row and single column. Let the vertex at the intersection of that row and column be denoted as $v_{r,c}$. We start by showing that in this case, an optimal feasible descent step is by obtained averaging the two values developed for each label on the intersecting vertex. Consider the integer programming problem given above with only constraints 5.18, 5.21, and 5.22 taken into account:

$$\text{maximize:} \quad \frac{1}{2} \{ \sum_{\rho_{rk} \in \Lambda_R} x_\rho(r,k)\, C_{rk} + \sum_{\chi_{rk} \in \Lambda_R} x_\chi(c,k)\, C_{ck} \} \qquad 5.23$$

$$\text{subject to:} \sum_{\rho_{rk} \in SR_{rc}(\lambda^i)} x_\rho(r,i) = \sum_{\chi_{ck'} \in SC_{rc}(\lambda^i)} x_\chi(c,k') \text{ for all } \lambda^i \in \Lambda. \qquad 5.24$$

$$\sum_{\rho_{ik} \in \Lambda_R} x_\rho(i,k) = 1 \qquad \text{for all rows,} \qquad 5.25$$

$$\sum_{\chi_{ik} \in \Lambda_R} x_\chi(i,k) = 1 \qquad \text{for all columns,} \qquad 5.26$$

Let C_ρ^i be the maximum value of the initial costs for all macro labels assigned to row r in which the label λ^i on vertex $v_{r,c}$ participates. Likewise, let C_χ^i be the maximum value of the initial costs for all macro labels assigned to column c in which the label λ^i on vertex $v_{r,c}$ participates. Note that the dynamic programming approach will automatically generate these values. Then the objective function, equation 5.23 can be expressed as:

$$\text{maximize:} \quad \frac{1}{2} \sum_{\lambda^i \in \Lambda} \{ C_\rho^i\, x_\rho^i + C_\chi^i\, x_\chi^i \}$$

$$\text{where} \qquad x_\rho^i = \sum_{\rho_{rk} \in SR_{rc}(\lambda^i)} x_\rho(r,k).$$

In other words, x_ρ^i is the indicator variable for all macro labels for row r which assign label λ^i to vertex $v_{r,c}$, and x_χ^i is the indicator variable for all macro labels for column c which assign label λ^i to vertex $v_{r,c}$. Note that by equations 5.25 and 5.26, the problem is constrained so that exactly one macro label can be chosen

for each row and each column. Thus, even though the the use of the consolidated decision variables x_ρ^i and x_χ^i by themselves result in an ambiguous choice of labels, the combination of these decision variables along with constraints 5.25 and 5.26 result in a unique choice of labels for the given row and column.

The auxiliary function for the dual form is given by multiplying equation 5.24 by the dual variable v_i and subtracting the right hand side of the resulting equation from the left and adding equation 5.27. By regrouping terms in v_i the dual can be seen to have the form:

$$\varphi(x,v) = \sum_{\lambda^i \in \Lambda} \{ (C_\rho^i + v_i)\, x_\rho^i + (C_\chi^i - v_i)\, x_\chi^i \} \qquad 5.28$$

The Lagrange dual statement of the problem is:

$$\text{minimize:} \qquad \Theta(\bar{v}) = \max_{\bar{x} \in X} \{\varphi(\bar{x},\bar{v})\}$$

where the set X consists of all primal vectors $\bar{x}$ which correspond to assigning exactly one macro-label to row r and exactly one macro label to column c.

Because v is unrestricted, we can make the substitution $v_i = C_\chi^i - \nu_i$ in equation 5.28. Thus

$$\varphi(\bar{x},\bar{\nu}) = \sum_{\lambda^i \in \Lambda} \{ (C_\rho^i + C_\chi^i - \nu_i)\, x_\rho^i + \nu_i\, x_\chi^i \} \qquad 5.29$$

If we set

$$\nu_i = \frac{(C_\rho^i + C_\chi^i)}{2} \qquad 5.30$$

then we have assigned a relative cost to each macro label, equal to the average of the corresponding macro labels with the given label λ^i for the row and column. We have the following result:

Theorem 5.3: The minimum value of

$$\Theta(\bar{\nu}) = \max_{\bar{x}\in X} \{ \varphi(\bar{x},\bar{\nu}) \} \qquad 5.31$$

$$= \max_{\bar{x}\in X} \{ \sum_{\lambda^i\in\Lambda} [(C_\rho^i + C_\chi^i - \nu_i) x_\rho^i + \nu_i x_\chi^i]$$

occurs at

$$\bar{\nu} = (\frac{C_\rho^1+C_\chi^1)}{2}, \frac{C_\rho^i+C_\chi^i)}{2}, \ldots, \frac{C_\rho^i+C_\chi^i)}{2})$$

Proof: For the sake of notational convenience define $f_{1i} = C_\rho^i + C_\chi^i - \nu_i$ and $f_{2i} = \nu_i$. Then because of the definition of the set X,

$$\min_{\bar{\nu}} \Theta(\bar{\nu}) = \min_{\bar{\nu}} \{ \max \{f_{11}, f_{12}, \ldots, f_{1m}\} + \max \{f_{21}, f_{22}, \ldots, f_{2m}\} \qquad 5.32$$

$$= \min_{\bar{\nu}} \{ | (f_{11}, f_{12}, \ldots, f_{1m} |_0 + | (f_{21}, f_{22}, \ldots, f_{2m} |_0$$

where $| \bar{f}_1 |_0$ is the "sup" norm of the vector $\bar{f}_1$. By the Cauchy-Schwartz inequality:

$$| (f_{11}, f_{12}, \ldots, f_{1m}) |_0 + | (f_{21}, f_{22}, \ldots, f_{2m}) |_0 \qquad 5.33$$

$$\geq | (f_{11} + f_{21}, f_{12} + f_{22}, \ldots, f_{1m} + f_{2m}) |_0$$

$$= | (C_\rho^1 + C_\chi^1, C_\rho^2 + C_\chi^2, \ldots, C_\rho^m + C_\chi^m) |_0$$

$$= 2 | (\frac{C_\rho^1 + C_\chi^1}{2}, \frac{C_\rho^2 + C_\chi^2}{2}, \ldots, \frac{C_\rho^m + C_\chi^m}{2}) |_0$$

which is just a restatement of the first part of the theorem.

The algorithm which incorporates the additional inequality constraint is the straightforward extension of algorithm 5.3 in which after generating the values for the macro labels, the *initial* labeling values for at the intersection of a given row and column are changed (increased for rows, decreased for columns) by a factor of the average values of the corresponding macro labels.

5.4. Summary

The discussions of the previous sections of this chapter have covered various means by which upper bounds on a given candidate subproblem can be generated. These techniques have been selected because they appear to be adapted to generated these bounds very rapidly if special purpose hardware were to be constructed for this purpose. In the following chapter, experimental experience with these techniques, along with certain heuristics will be described. The other fundamental components of the branch and bound approach, the test for feasibility and generation of lower bounds can similarly be generated very efficiently. The test for feasibility involves the discrete relaxation operation and the generation of lower bounds involves the use of the bi-direction dynamic programming algorithm both of which will be described further in the following chapter.

CHAPTER VI

EXPERIMENTS

6.1. The Full Enumerative Scheme

As part of the current work, an implicit enumeration approach was attempted using the basic strategy described in section 3.5. For these experiments, the constraint network of the Appendix was used. It is easily verifed that for this constraint network any consistent labeling of a given row allows at least one consistent labeling of an adjacent row. Thus, a "greedy" approach to generating lower bounds on a given restriction and on an initial incumbent solution can be generated by solving the problem using the dynamic programming approach along the first row and then, consistent with the best solution for the first row, solve the problem for the second row, and so forth. As might be expected, the solutions generated in this manner are not reasonable approximations of an expected line drawing. What happens, in fact, is that the choice of spurious edge segments (such as λ_{16}) tends to propagate down successive rows. A better approximation is probably achieved by using the (possibly inconsistent) labelings derived from the upper bound function, which in this case, was implemented by applying the dynamic programming approach along rows and the Lagrange dual formulation of the rows as macro-labels.

In order to be of practical value, the technique must be able to solve problems on at least a 64×64 image. It was determined that, running on a VAX11/780 under UNIX, a problem defined on an 8×8 image took approximately

one and one half hours of computer time for a test image that was only slightly perturbed from an the original integer solution, despite the fact that the programs were very tightly coded. Experiments on larger images were not even attempted. Such experiments would be left to larger machines or special purpose hardware which might make this approach more feasible. Nonetheless, labelings derived from the fixed points of the dual descent algorithms defined on the reduced constraint networks often seemed to be much "cleaner" (i.e. there were fewer inconsistencies than in the original image).

6.2. Heuristic Approaches

As a conclusion, we make note of several apparently "effective" heuristic approaches which were discovered during the course of this project. These approaches are based on the use of a "relaxation-like" updating rule which was discovered as a result of experimentation with ad hoc extensions of the dynamic programming approach to the case where the underlying graph corresponds to an image raster. This approach, referred to here as the "average-max" updating rule (Diamond, 1983) updates the current labeling value c_{ij}^t of a label λ_j on vertex v_i by:

$$c_{ij}^{t+1} = \frac{1}{N+1} \left[\left(\sum_{v_j \in N(i)} \max_{\lambda_k \in \Lambda} \{ r_{ij}(\lambda,\lambda') c_{jk}^t \} \right) + c_{ij}^t \right].$$

Where $N(i)$ is the set of vertices adjacent to vertex v_i, $r_{ii'}(\lambda j,\lambda_{j'})$ is one if label λ_j on vertex v_i is consistent with label $\lambda_{j'}$ on vertex $v_{i'}$ and zero otherwise, and $N = | N(i) |$. Thus, a processor performing the updating for label λ_j on vertex v_i would generate N values to be averaged (along with the current labeling value c_{ij}^t), one such value corresponding to each vertex in the neighborhood, by taking the maximum of the current labeling values associated with labels consistent with λ_j on vertex v_i.

If $\bar{\lambda}^t$ is the current labeling chosen by the local maxima selection process and $c(\bar{\lambda}^t)$ is the sum of the associated current labeling values, then the average-

max updating rule can be easily be seen to have the following properties:

[1] If it is not a consistent labeling, then the sum of the current labeling values with $\bar{\lambda}^t$ will have decreased after the next iteration, that is $c(\bar{\lambda}^{t+1}) < c(\bar{\lambda}^t)$.

[2] If $\bar{\lambda}^t$ is consistent then $c(\bar{\lambda}^{t+1}) = c(\bar{\lambda}^t)$ and the labeling selected at each iteration $t' \geq t$ will be the same.

[3] From [1] and [2] the value $c(\bar{\lambda}^t)$ is a non-increasing function of t. Furthermore, it can be shown that the value associated with a consistent labeling will increase at each iteration, if that labeling is not the one selected by the local maxima selection process.

From these observations, it can be concluded that if the labeling values selected by the local maxima selection process is not consistent, then the difference between the current labeling value associated with a given consistent labeling and the labeling chosen by the local maxima selection process will strictly decrease at each iteration. This does not mean, however, that the application of this process will necessarily result in a consistent labeling. In fact what may happen is that the sum of the current costs of the chosen (inconsistent) labeling and a given inconsistent labeling will converge to the same value.

Despite this, it has been observed that the "average-max" updating rule will almost always derive a consistent labeling in experiments performed on a 16 x 16 image starting with an initial (artifically created) line drawing which is perturbed by adding noise to the initial labeling values. Figure 6.1 is an initial consistent labeling representing a contour. The initial labeling values which represent this figure assigns a value of 1.0 to each label shown and a value of 0.0 to the other labels at the given pixel. Figure 6.2a is the result of adding values sampled from a uniform distribution with lower bound of 0.0 and a maximum possible value of 2.2 to the initial labeling values, and then selecting that label at each pixel such that the current labeling value is maximum. Iterations of this process are shown in

figure 6.2b through 6.2e.

Another ad hoc updating rule which was tried involves alternating a step of the average-max updating rule with a step of the descent procedure of algorithm 5.1. Again no explanation for the behavior of this algorithm can be given, however, in practically all cases, a consistent labeling resulted. The application of this last technique to a real world problem is shown in figure 6.3a though 6.3e.

There may be some promise in the graph labeling approach to the edge linking problem discussed in the preceding sections of this report. Much difficulty has resulted, however, in the application of the ad hoc techniques to real world images. We cite the following problems:

(1) The ad hoc techniques applied to this problem are extremely sensitive to the way in which the initial labeling values are derived. It is expected that even if the problem itself could in fact be solved for a real world example that this would still be the case.

(2) The original experiments were based on the assumption that the initial image could be represented as a complete contour (consistent labeling) to which noise has been added. Given this basis, the ad hoc approaches described above performed extremely well in rederiving the original line drawings in an image. In the case of the initial line drawing derived from a real world images, however, "good" contours were derived only when the initial labelings were derived in conjunction with a a great deal of initial processing. In particular, in many cases the output of the thinning operation applied to the results of the threholded Sobel operator, were in fact better that the results of the linking operation for most images used.

In summary, one of the goals of this project was to give meaning to the apparent strategy of the relaxation labeling processes. The conclusion that can be drawn from our experience with the applications of such techniques to real

world images is however, that even if a formal basis is given to the problem, and even if the problem could be solved as defined, the results would not be very relevant to the ultimate goal of recognizing objects from line drawings, in most situations. It is felt that the type of information which is integrated into the visual recognition process is much more global than is afforded by the local processing inherent in these techniques by themselves. Nonetheless, the optimization approach to the problem of edge linking proposed here may be useful if integrated into a model based approach to computer vision, such as found in the generalized Hough transforms for example. Furthermore there may be classes of images for which the local edge linking approach described here may be useful. For example, it is believed that the contour extraction problem defined on some classes of medical images may be good candidates for these approaches.

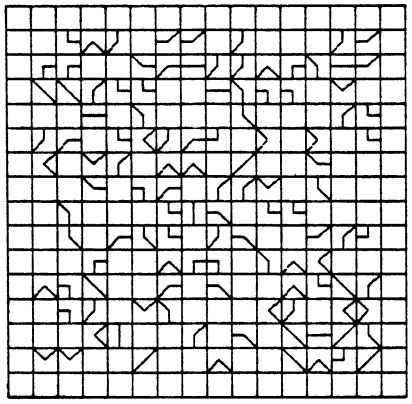


Fig. 6.2a: Initial labeling.

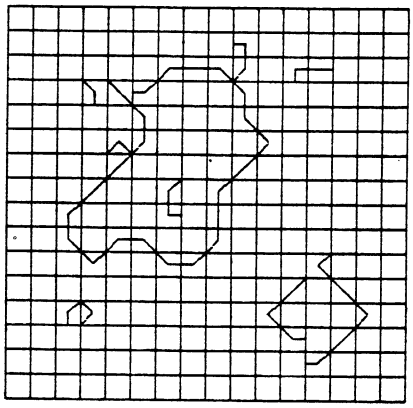


Fig. 6.2b: Iteration 2.

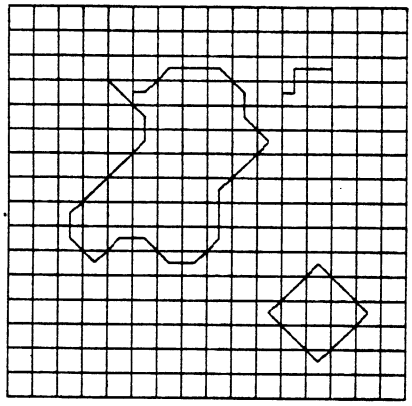


Fig. 6.2c: Iteration 4.

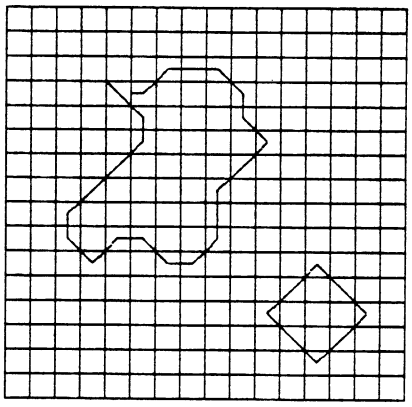


Fig. 6.2d: Iteration 8.

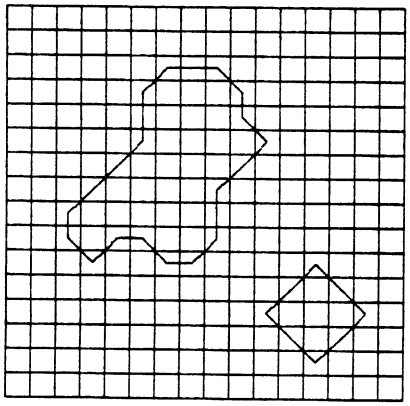


Fig. 6.2e: Iteration 12.

Figure 6.2: Application of the "average-max" updating procedure to the contour of figure 6.1.

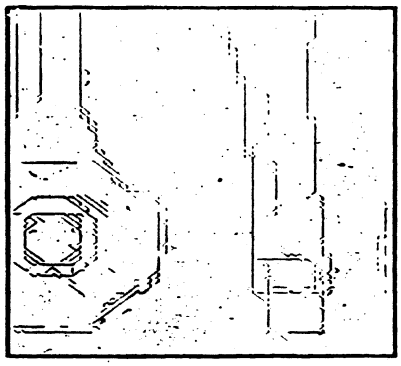


Fig. 6.3a: Initial labeling.

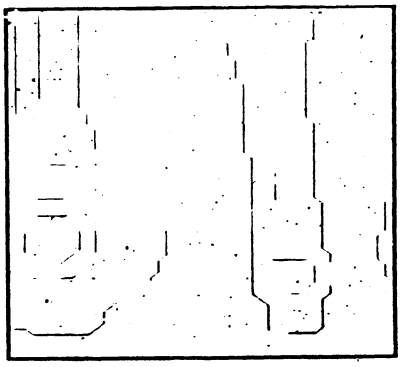


Fig. 6.3b: Iteration 2.

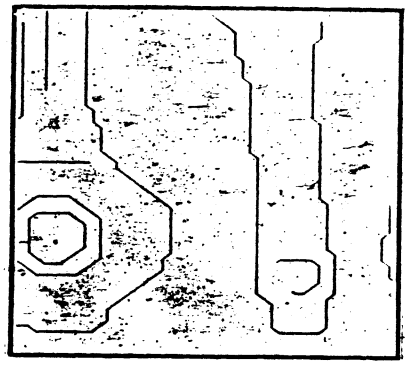


Fig. 6.3c: Iteration 8.

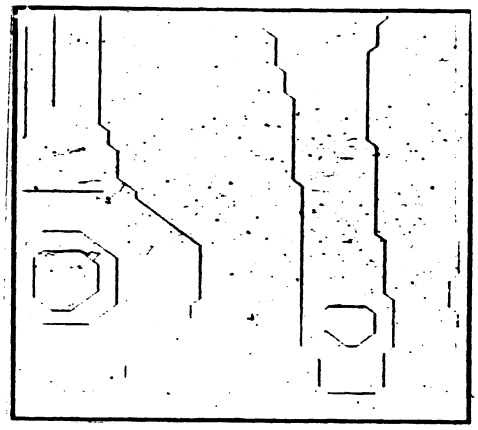


Fig. 6.3d: Iteration 16.

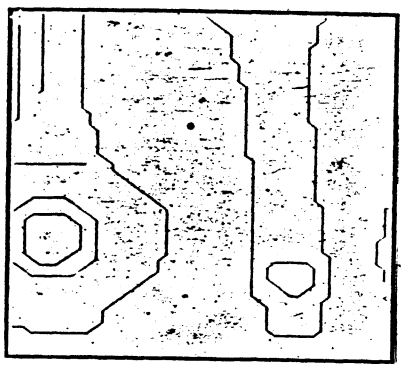


Fig. 6.3e: Iteration 20.

Figure 6.3: Application of the combination "average-max" and Lagrange dual updating procedure to a scene from the GM "bin-of-parts" database.

CHAPTER VII

SUMMARY AND CONCLUSIONS

7.1. Summary and Conclusions

The work that has been presented in the previous six chapters represents the introduction of an approach to the problem of edge linking when it is formulated in terms of the graph labeling model. This work also contains the proposal for the definition of the graph labeling problem on which the algorithms and theory presented here are based. In contrast to prior views towards graph labeling schemes, which are for the most part ad hoc, the approach here has been to specify the meaning of the graph labeling model and its relation to a given application in a formal manner. The goal has been to develop means by which a solution to the edge linking problem can be implemented in hardware so as to provide a real time line drawing extraction scheme. The motivation for this work is obvious when considering the importance of segmentation in general to problems of machine perception, and the importance of the problem of line drawing extraction to the problem of computer vision in particular. We have attempted to explore the feasibility of this approach in solving the edge linking problem. However, in order to do so, it is felt that hardware sufficient to solve the problem which was not available to us would have to be used.

Besides the introduction of a new approach to the application itself, this work further contains initial results related to the graph labeling model. In

particular, time bounds on the convergence of the discrete relaxation operator, under restricted conditions have been derived. These restrictions pertain to the situation where the underlying graph does not contain cycles. Under these conditions, results related to the polytope of the graph labeling problem were also derived. In particular, it has been demonstrated that the feasible region for the linear programming relaxation of the graph labeling polytope has all integer (0-1) extrema. Although it appears that most graph labeling problems of practical value will be based on a graph which contains cycles, these restricted case results may be important in constraint relaxation approaches to generating upper bounds in a branch and bound approach to the more general case problem.

The application of the theory of dynamic programming and Lagrange duality to the graph labeling problem has been presented. Although the basic building blocks of these approaches are well-established, the means by which the components are put together for the particular application is a contribution which has been made here. The algorithms which have resulted from this combination of approaches could be used in the generation of upper bounds in an implicit enumeration scheme. The specific motivation behind their design is that they be adaptable to implementation in a highly parallel manner.

In each of the issues related to the graph labeling problem presented above, the work is clearly far from complete. As such, this thesis represents an exploration of some of the basic issues involved, and hence is more broad than deep. This work represents an initial investigation into a new class of problems and its application to the problem of edge linking. It is felt that the graph labeling model is also relevant to problems in other areas of pattern recognition and artificial intelligence. Beyond the definition of this class of problems and the related analytical results which have been presented in the previous chapter, the real contribution which we hope to have made is in motivating others to pursue this topic futher.

7.2. Suggestions for further work

In most cases, those areas requiring extensions to the current work were noted in the appropriate sections of the preceding chapters. The following lists, by way of summary, areas which it is felt should be pursued further.

Theoretical: Perhaps most important, it is felt that work is needed to further extend the theory of the structure of the graph labeling problem itself. In particular, the nature of the linear programming relaxation of the graph labeling problem should be further explored, as well as a specification of the related facets. Furthermore, the relationship between the fixed points of the dual descent procedure and the LP relaxation should be fully characterized. As noted in the chapter on experiments (chapter 6) there exists a means, in the case where the underlying graph has specific major directions, to "round" to an integer solution. Understanding the nature of this rounding process, as well as devising other ways to find nearby integer solutions can be pursued only once this work is complete. Conjectures which might lead to results are stated at the end of chapter 4.

Of the algorithms discussed in chapters 5 and 6, it is felt that it might be good if a greater understanding of the "average-max" updating procedure could be obtained. It is somewhat difficult, perhaps, to justify this suggestion since it was in answer to the relaxation labeling processes that this entire work was started. However, it is felt that the apparent success of this updating procedure justifies this.

Experimental: The section covering the experimental aspects of this project was necessarily brief. Should the resources become available, it would be interesting to see if full (or at least closer) solutions to the graph labeling problem could be obtained as a means for determining whether or not this entire approach to the

original application, the extraction of line drawings from grey level images, should be pursued further. As noted in the previous chapter, these experiments would be better pursued in conjunction with an array processor. With respect to the hybrid approaches discussed in chapter V, it is felt that several question must still be answered, most important is whether or not the bounds generated from these strategies are tighter than the bounds generated, for example, by the original form of the Lagrange dual minimization procedure.

Finally, there are several crucial questions to be addressed from an applications perspective, which were not even considered here. These issues include, for example, (1) how the initial labeling values should be derived for a given application, (2) how the constraint network should be defined for a given application, and (3) for what classes of images are the techniques derived here relevant, and for what classes of images are they not relevant.

APPENDIX

APPENDIX

CONSTRAINT NETWORK FOR THE

EDGE LINKING APPLICATION

There are three basic components to the constraint network for the edge linking application. They are (1) the underlying graph, (2) the label set, and (3) the constraint relations. The underlying graph for this constraint network is given in figure 3.1. In this case each label corresponds to a scene event, including corners, elbows, and straight lines at various orientations as shown in figure A.1. There are 22 labels in all. Label λ_{20} (not shown) is a "knot", The knot is consistent with any incident line segment and is meant to allow for the intersection of several contours in the scene. Label λ_{21} corresponds to a blank pixel, that is, a pixel with no contour running through it.

The constraint relations for this constraint network are shown in table A.1. This table gives for each label, the consistent labels in each of the four neighboring vertices, (1) immediately to the right, (2) upper right hand corner, (3) immediately above, and (4) upper left hand corner. This table completely specifies the constraint network, since the constraint relations are symmetric. Thus, the constraint relation between the a vertex and the vertex immediately below it, is the transpose of the constraint relation for a vertex and the vertex immediately above it. Because the objective is to implicitly define the set of all possible unbroken contours, a pair of labels on adjacent vertices will be considered to be inconsistent if the scene events they represent have line

segments broken across a pixel boundary. Examples of consistent pairs of labels are shown in figure A.2. Examples of inconsistent pairs of labels are shown in figure A.3.

Note that even by this criterion, there is still some ambiguity in the definition of the constraint relations. For example, it is possible to allow the label λ_5 on a given vertex to be either consistent or inconsistent with the label λ_{16} on the vertex to the immediate right. The constraint relations given above were selected because they define the constraint network in such a way so that given a consistent labeling on a given row of the raster, there exists at least one consistent labeling of the row immediately below and above it. This allows for the use of "rounding" to generate lower bounds in a basic branch and bound approach.

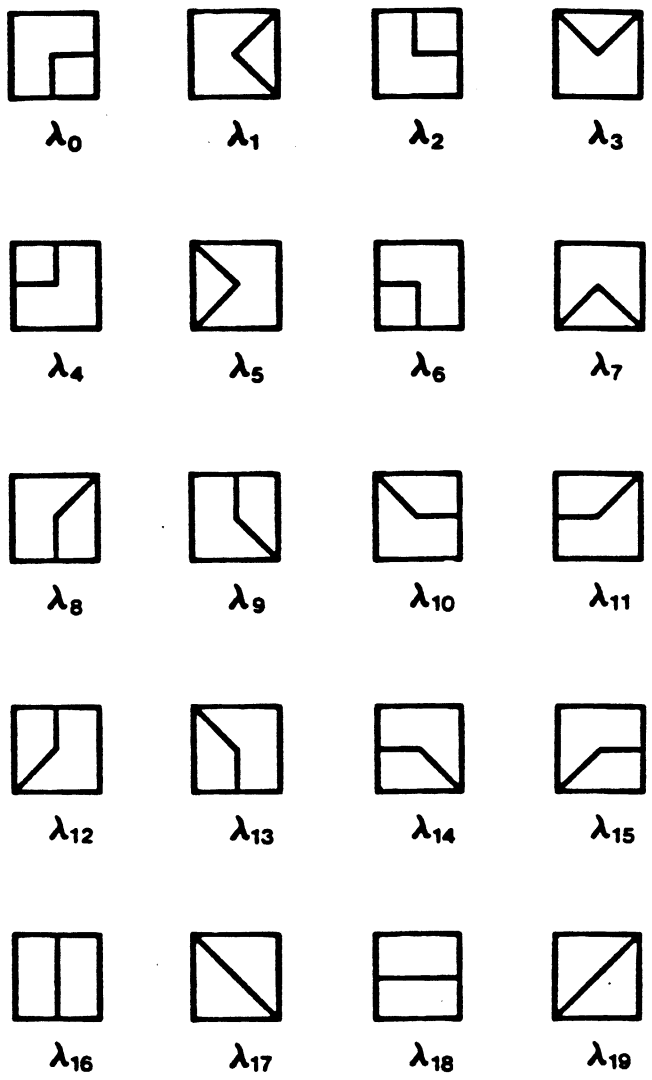


Figure A.1: Scene events and the associated labels for the edge linking application.

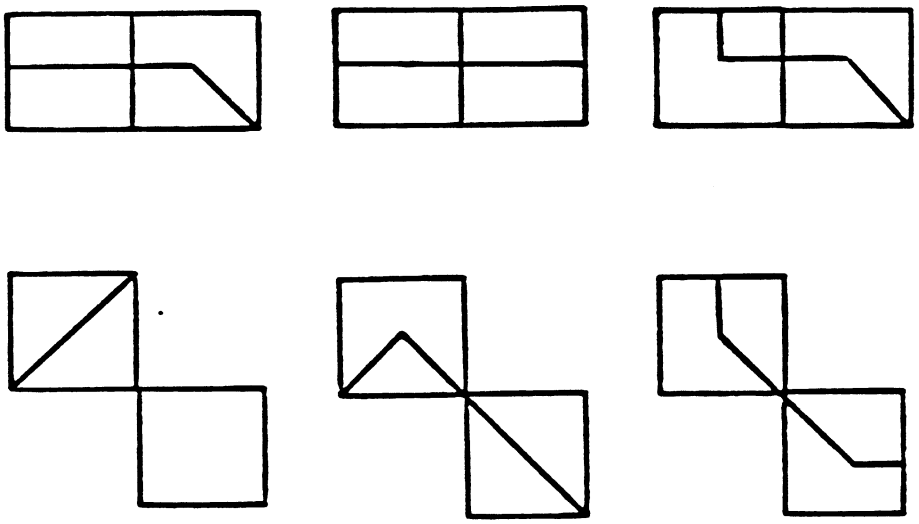


Figure A.2: Examples of locally consistent pairs of labels.

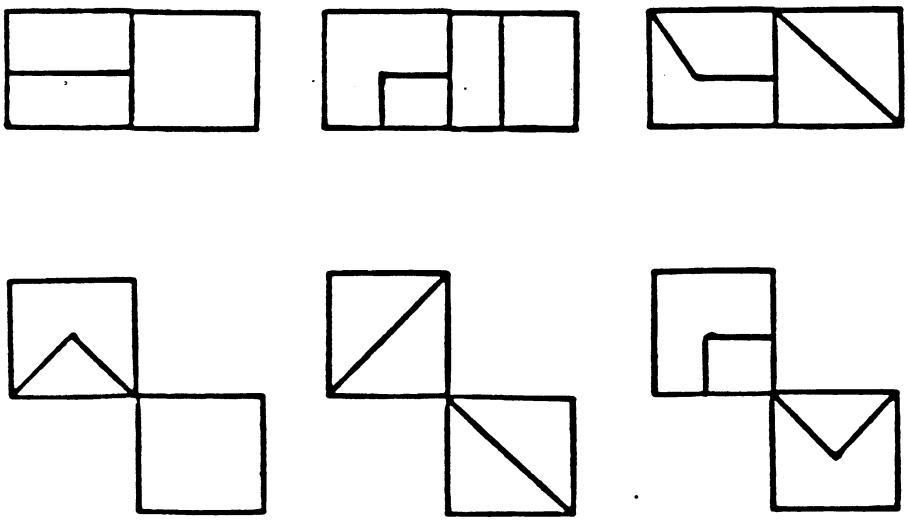


Figure A.3: Examples of locally inconsistent pairs of labels.

label	set of consistent labels on the vertex to the immediate right
λ_0	λ_4, λ_{11}, λ_{14}, λ_{18}
λ_1	λ_{21}
λ_2	λ_6, λ_{11}, λ_{14}, λ_{18}
λ_3	λ_{21}
λ_4	λ_{21}
λ_5	λ_{21}
λ_6	λ_{21}
λ_7	λ_{21}
λ_8	λ_{21}
λ_9	λ_{21}
λ_{10}	λ_4, λ_6, λ_{11}, λ_{14}, λ_{18}, λ_{20}
λ_{11}	λ_{21}
λ_{12}	λ_{21}
λ_{13}	λ_{21}
λ_{14}	λ_{21}
λ_{15}	λ_4, λ_6, λ_{11}, λ_{14}, λ_{18}, λ_{20}
λ_{16}	λ_{21}
λ_{17}	λ_{21}
λ_{18}	λ_4, λ_6, λ_{11}, λ_{14}, λ_{18}, λ_{20}
λ_{19}	λ_{21}
λ_{20}	λ_{11}, λ_{14}, λ_{18}, λ_{21}
λ_{21}	λ_0, λ_1, λ_2, λ_3, λ_5, λ_7, λ_8, λ_9, λ_{10}, λ_{12}, λ_{13}, λ_{15}, λ_{16}, λ_{17}, λ_{19}, λ_{20}, λ_{21}

Table A.1: Definition of constraint relation for a vertex and its neighbor to the immediate right.

label	set of consistent labels on the vertex to the upper right
λ_0	λ_0, λ_1, λ_2, λ_3, λ_4, λ_6, λ_8, λ_9, λ_{10}, λ_{11}, λ_{13}, λ_{14}, λ_{16}, λ_{17}, λ_{18}, λ_{21}
λ_1	λ_5, λ_{12}, λ_{15}, λ_{19}
λ_2	λ_0, λ_1, λ_2, λ_3, λ_4, λ_6, λ_8, λ_9, λ_{10}, λ_{11}, λ_{13}, λ_{14}, λ_{16}, λ_{17}, λ_{18}, λ_{21}
λ_3	λ_7, λ_{12}, λ_{15}, λ_{19}
λ_4	λ_0, λ_1, λ_2, λ_3, λ_4, λ_6, λ_8, λ_9, λ_{10}, λ_{11}, λ_{13}, λ_{14}, λ_{16}, λ_{17}, λ_{18}, λ_{21}
λ_5	λ_0, λ_1, λ_2, λ_3, λ_4, λ_6, λ_8, λ_9, λ_{10}, λ_{11}, λ_{13}, λ_{14}, λ_{16}, λ_{17}, λ_{18}, λ_{21}
λ_6	λ_0, λ_1, λ_2, λ_3, λ_4, λ_6, λ_8, λ_9, λ_{10}, λ_{11}, λ_{13}, λ_{14}, λ_{16}, λ_{17}, λ_{18}, λ_{21}
λ_7	λ_0, λ_1, λ_2, λ_3, λ_4, λ_6, λ_8, λ_9, λ_{10}, λ_{11}, λ_{13}, λ_{14}, λ_{16}, λ_{17}, λ_{18}, λ_{21}
λ_8	λ_5, λ_7, λ_{12}, λ_{15}, λ_{19}, λ_{20}
λ_9	λ_0, λ_1, λ_2, λ_3, λ_4, λ_6, λ_8, λ_9, λ_{10}, λ_{11}, λ_{13}, λ_{14}, λ_{16}, λ_{17}, λ_{18}, λ_{21}
λ_{10}	λ_0, λ_1, λ_2, λ_3, λ_4, λ_6, λ_8, λ_9, λ_{10}, λ_{11}, λ_{13}, λ_{14}, λ_{16}, λ_{17}, λ_{18}, λ_{21}
λ_{11}	λ_5, λ_7, λ_{12}, λ_{15}, λ_{19}, λ_{20}
λ_{12}	λ_0, λ_1, λ_2, λ_3, λ_4, λ_6, λ_8, λ_9, λ_{10}, λ_{11}, λ_{13}, λ_{14}, λ_{16}, λ_{17}, λ_{18}, λ_{21}
λ_{13}	λ_0, λ_1, λ_2, λ_3, λ_4, λ_6, λ_8, λ_9, λ_{10}, λ_{11}, λ_{13}, λ_{14}, λ_{16}, λ_{17}, λ_{18}, λ_{21}
λ_{14}	λ_0, λ_1, λ_2, λ_3, λ_4, λ_6, λ_8, λ_9, λ_{10}, λ_{11}, λ_{13}, λ_{14}, λ_{16}, λ_{17}, λ_{18}, λ_{21}
λ_{15}	λ_0, λ_1, λ_2, λ_3, λ_4, λ_6, λ_8, λ_9, λ_{10}, λ_{11}, λ_{13}, λ_{14}, λ_{16}, λ_{17}, λ_{18}, λ_{21}
λ_{16}	λ_0, λ_1, λ_2, λ_3, λ_4, λ_6, λ_8, λ_9, λ_{10}, λ_{11}, λ_{13}, λ_{14}, λ_{16}, λ_{17}, λ_{18}, λ_{21}
λ_{17}	λ_0, λ_1, λ_2, λ_3, λ_4, λ_6, λ_8, λ_9, λ_{10}, λ_{11}, λ_{13}, λ_{14}, λ_{16}, λ_{17}, λ_{18}, λ_{21}
λ_{18}	λ_0, λ_1, λ_2, λ_3, λ_4, λ_6, λ_8, λ_9, λ_{10}, λ_{11}, λ_{13}, λ_{14}, λ_{16}, λ_{17}, λ_{18}, λ_{21}
λ_{19}	λ_5, λ_7, λ_{12}, λ_{15}, λ_{19}, λ_{20}
λ_{20}	λ_{12}, λ_{15}, λ_{19}, λ_{21}
λ_{21}	λ_0, λ_1, λ_2, λ_3, λ_4, λ_6, λ_8, λ_9, λ_{10}, λ_{11}, λ_{13}, λ_{14}, λ_{16}, λ_{17}, λ_{18}, λ_{20}, λ_{21}

Table A.2: Definition of constraint relation for a vertex and its neighbor to the upper right.

label	set of consistent labels on the vertex immediately above
λ_0	λ_{21}
λ_1	λ_{21}
λ_2	λ_6, λ_8, λ_{13}, λ_{16}
λ_3	λ_{21}
λ_4	λ_0, λ_8, λ_{13}, λ_{16}
λ_5	λ_{21}
λ_6	λ_{21}
λ_7	λ_{21}
λ_8	λ_{21}
λ_9	λ_0, λ_6, λ_8, λ_{13}, λ_{16}, λ_{20}
λ_{10}	λ_{21}
λ_{11}	λ_{21}
λ_{12}	λ_0, λ_6, λ_8, λ_{13}, λ_{16}, λ_{20}
λ_{13}	λ_{21}
λ_{14}	λ_{21}
λ_{15}	λ_{21}
λ_{16}	λ_0, λ_6, λ_8, λ_{13}, λ_{16}, λ_{20}
λ_{17}	λ_{21}
λ_{18}	λ_{21}
λ_{19}	λ_{21}
λ_{20}	λ_8, λ_{13}, λ_{16}, λ_{21}
λ_{21}	λ_1, λ_2, λ_3, λ_4, λ_5, λ_7, λ_9, λ_{10}, λ_{11}, λ_{12}, λ_{14}, λ_{15}, λ_{17}, λ_{18}, λ_{19}, λ_{20}, λ_{21}

Table A.3: Definition of constraint relation for a vertex and its neighbor immediately above.

BIBLIOGRAPHY

BIBLIOGRAPHY

[AsM78] G.P. Ashkar, and J.W. Modestino, "The contour extraction problem with biomedical applications," *Computer Graphics and Image Processing*, 7, 1978, 331-355.

[BaB82] D.H. Ballard, and C.M. Brown, *Computer Vision*, Englewood Cliffs, New Jersey: Prentice-Hall, 1982.

[BaP74] E. Balas, and M.W. Padberg, "Set partitioning," in *Combinatorial Programming: Methods and Applications*, B. Roy (ed.), Dordrecht-Holland, D. Reidel, 1974.

[BaS79] M.S. Bazarra and C.M. Shetty, *Nonlinear Programming: Theory and Algorithms*, New York, John Wiley and Sons, 1979.

[BaT76] H. Barrow and J. M. Tennenbaum, "MSYS: a system for reasoning about scenes," *Stanford Research Institute AI Center Tech. Note 121*, 1976.

[Bal81] Ballard, D.H., "Generalized Hough transform to detect arbitrary shapes," *Pattern Recognition*, 12, 2, pp. 111-122, 1981.

[Bar80] J.J. Bartholdi III, "A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering," *Operations Research* vol. 29, pp. 501-510, 1981.

[Ber73] C. Berge, *Graphs and Hypergraphs*, Amsterdam, North-Holland, 1973.

[COM83] *Computer*, January 1983, Special issue on "Computer Architectures for Image Processing."

[Chu79] K. W. Church, "Co-ordinate squares: a solution to many chess-pawn endgames,"

[Chv75] V. Chvatal, "On certain polytopes associated with graphs," *J. Combin. Theory* 18, 138-154.

[Clo71] M. B. Clowes, "On seeing things," *Artificial Intelligence*, vol. 2, pp. 79-116, 1971.

[DiG82] M.D. Diamond, and S. Ganapathy, "Cooperative solutions to the continuous graph labeling problem: review and reformulation," *Proc. 1982 IEEE Conf. Pattern Recognition and Image Processing*, Las Vegas, pp. 64-71, July 1982.

[DNG82] M.D. Diamond, N. Narasimhamurthi, and S. Ganapathy, "A systematic approach to continuous graph labeling with application to computer vision" *Proc. 1982 AAAI Nat. Conf. on Art. Intell.*, Pittsburgh, pp. 50-54, August 1982.

[Dia82] M.D. Diamond, "The graph labeling problem and the relaxation labeling processes," *RSD-TR-13-82*, Center for Robotics and Integrated Manufacturing, Robot System Division, University of Michigan, Ann Arbor, October, 1982.

[DaR77] L. S. Davis and A. Rosenfeld, "An application of relaxation labeling to spring-loaded template matching," *Proc. Third Intl. Joint Conf. on Pattern Recognition*, pp. 591-597, November, 1977.

[DaR78] L. S. Davis and A. Rosenfeld, "Hierarchical relaxation for waveform parsing," in *Computer Vision Systems*, A. R. Hanson and E. M. Riseman, Eds. New York: Academic, 1978, pp. 101-109.

[DaR80] L. S. Davis, and A. Rosenfeld, "Cooperating processes for low-level vision: a survey," *TR-123*, Dept. of Computer Science, University of Texas, Austin, 1980.

[DoR79] G.G. Dodd, and L. Rossol (eds.), *Symposium on Computer Vision and Sensor-Based Robots*, General Motors Research Laboratories, Warren, Michigan, New York: Plenum Press, 1979.

[DuH72] R.O. Duda, and P.E. Hart, "Unse of the Hough transformation to detect lines and curves in pictures," *Commun. ACM*, 15, 1, pp. 11-15, 1972.

[Dav79] L. S. Davis, "Shape matching using relaxation techniques," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, pp. 60-72, 1979.

[EYR80] J. O. Eklundh, H. Yamamoto, and A. Rosenfeld, "A relaxation method for multispectral pixel classification," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 72-75, 1980.

[Etc77] J. Etcheberry, "The set-covering problem: a new implicit enumeration algorithm," *Operations Research* vol. 25, pp. 760-772, 1977.

[FaB81] O. D. Faugeras, and M. Berthod, "Improving consistency and reducing ambiguity in stochastic labeling: an optimization approach," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-3, pp. 412-424, 1981.

[Fau81] O.D. Faugeras, "Decomposition and Decentralization Techniques in Relaxation labeling," *Comput. Graphics Image Processing*, vol. 13, pp. 341-355, 1981.

[Fre78] E. C. Freuder, "Synthesizing constraint expressions," *Comm. ACM, vol. 21, pp.958-966, 1978.*

[Ful71] D. R. Fulkerson, "Blocking and anti-blocking pairs of polyhedra," *Mathematical Programming* vol. 3, pp. 168-194, 1971.

[GaJ78] M.R. Garey, and D.S. Johnson *Computers and Intractability: A guade to the Theory of NP-completeness* San Francisco; Freeman, 1978.

[Gas74] J. A. Gashnig, "Constraint satisfaction method for inference making," *Proc. 12th Allerton Conf. Circ. Syst. Theory*, Urbana-Champaign, IL, 1974.

[Gol80] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, New York; Academic Press, 1980.

[Gro76] R. M. Grossman, "Some database applications of constraint expressions," *T. R. 158*, Laboratory for Computer Science, M.I.T., Cambridge, MA, 1976.

[HDR78] R. M. Haralick, L. S. Davis, A. Rosenfeld, and D. L. Milgram, "Reduction operations for constraint satisfaction," *Information Sciences, vol. 14, pp. 199-219, 1978.*

[HaE79] R. M. Haralick, and G. L. Elliot, "Increasing tree search efficiency for constraint satisfaction problems," *Proc. 6th Int. Joint Conf. Artificial Intell.*, pp. 356-364, Tokyo, Japan, 1979.

[HaR78a] *Computer Vision Systems*, New York: Academic, 1978.

[HaR78b] A. R. Hanson and E. M. Riseman, "Segmentation of natural scenes," in *Computer Vision Systems*, A. R. Hanson and E. M. Riseman, Eds. New

York: Academic, 1978, pp. 129-163.

[HaS79] R. M. Haralick and L. G. Shapiro, The consistent labeling problem: part I," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, pp. 173-184, 1979.

[HaS80] R. M. Haralick and L. G. Shapiro, The consistent labeling problem: part II," *IEEE Trans. Pattern Anal. Machine Intell., IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 193-203, 1980.

[Har72] F. Harary, *Graph Theory*, Reading, Massachusetts; Addison-Wesley, 1972.

[Hay79] K. C. Hayes, "Reading handwritten words using hierarchical relaxation," *TR-783*, Computer Science Center, University of Maryland, College Park, MD, 1979.

[Hor71] B.K.P. Horn, "The Binford-Horn line finder", *AIM-285*, Artificial Intelligence Laboratory, MIT, 1971.

[HuZ80] R. A. Hummel, and S. W. Zucker, "On the foundations of relaxation labeling processes," *Tr-80-7*, Dept. of Elect. Eng., McGill University, Montreal, Quebec, Canada.

[Huf71] D. A. Huffman, "Impossible objects and nonsense sentences," *Machine Intelligence*, vol. 6, pp. 295-323, 1971.

[Kir80] R. L. Kirby, "A product rule relaxation method," *Comput. Graphics Image Processing*, vol. 12, pp. 158-189, 1980.

[LWW78] J.M. Lester, H.A. Williams, B.A. Wientraub, J.F. Brenner, "Two graph searching techniques for boundary finding in white blood cells," *Computers in Biology and Medicine*, 8, 1978, 293-308.

[MPP77] D. Marr, G. Palm, and T. Poggio, "Analysis of a cooperative stereo algorithm," *A. I. Memo No. 446, A. I. Lab.*, Massachusetts Institute of Technology, 1977.

[MaP76] D. Marr and T. Poggio, "Cooperative computation of stereo disparity," *Science*, vol. 194, pp. 283-287, 1976.

[Mac77] A. K. Mackworth, "Consistency in networks of relations," *Artificial Intelligence*, vol. 8, pp. 99-118, 1977.

[Mar74] R.E. Marsten, "An algorithm for large set partitioning problems," *Management Science*, 20, 5, 774-787, 1974.

[Mar75] D. Marr, "Analyzing natural images; a computational theory of texture vision," *TR 334*, AI Lab, MIT, June 1975. D. Marr, "Early processing of visual information," *Philosphical Transactions of the Royal society of London*, vol. 275, no. 942, October 19, 1976.

[Mar80] P. Marks, "Low-level vision using an array processor," CG 80 281 292

[MoW79] F. S. Montalvo, and N. Weisstein, "An empirical method that provides a basis for the organization of relaxation labeling processes for vision," *Proc. 6th Int. Joint Conf. Artificial Intell.*, pp. 595-597, Tokyo, Japan, 1979.

[Mon74] U. Montanari, "Networks of constraints: fundamental properties and application to picture processing," *Information Sciences*, vol. 7, pp. 95-132, 1974.

[Mur76] K. Murty, *Linear and Combinatorial Programming*, New York: Wiley and Sons, 1976.

[NTN74] G.L. Nenhauser, L.E. Trotter, and R.M. Nauss, "Set partitioning and chain decomposition," *Mathematical Programming* vol. 6, pp. 1413-1423, 1974.

[NeT74] G.L. Nemhauser, and L.E. Trotter, "Properties of vertex packing and independence system polyhedra," *Mathematical Programming* vol. 6, pp. 48-61, 1974.

[NeT75] G.L. Nemhauser, and L.E. Trotter, "Vertex packings: structural properties and algorithms," *Mathematical Programming* vol. 7, pp. 232-248, 1975.

[Pad73] M.W. Padberg, "On the facial structure of set packing polyhedra," *Mathematical Programming* vol. 5, pp. 199-215, 1973.

[Pad74] M.W. Padberg, "Characterisations of totally unimodular, balanced and perfect matrices," in *Combinatorial Programming: Methods and Applications*, B. Roy (ed.), Dordrecht-Holland, D. Reidel, 1974.

[PeR78] S. Peleg and A. Rosenfeld, "Determining compatibility coefficients for curve enhancement relaxation processes," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, pp. 548-555, 1978.

[PeR79] S. Peleg, and A. Rosenfeld, "Breaking substitution ciphers using a relaxation algorithm," *TR-721*, Computer Science Center, University of Maryland, College Park, MD, 1979.

[Pel80] S. Peleg, "A new probabilistic relaxation scheme," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 362-369, 1980.

[Per78] W.A. Perkins, "A model-based vision system for industrial parts," *IEEE Trans. Comput.*, vol. C-27, pp. 126-143, 1978.

[PiQ77] J.C. Picard, and M. Queyranne, "On the integer-valued variables in the linear vertex packing problem," *Mathematical Programming* vol. 9, pp. 97-101, 1977.

[Pie68] J.F. Pierce, "Application of combinatorial programming to a class of all zero-one integer programs," *Management Science*, 15, 191-209, 1968.

[RHZ76] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene labeling by relaxation operations," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, pp. 420-433.

[Ros79] A. Rosenfeld, "Survey; Picture processing: 1978", *Comput. Graphics Image Processing*, vol. 11, pp. 354-393, 1979.

[Ros81] A. Rosenfeld, "Survey: picture processing: 1980," *Computer Graphics and Image Processing, Comput. Graphics Image Processing*, vol. 13, pp. 52-89, 1981.

[Sac79] E. D. Sacerdoti, "Problem solving tactics," *Proc. 6th Int. Joint Conf. Artificial Intell.*, pp. 1077-1085, Tokyo, Japan, 1979.

[Sal75] H.M. Salkin, *Integer Programming*, Reading, Mass.: Addison-Wesley, 1975.

[Shi75] Y. Sharai, "Analyzing intensity arrays using knowledge about scenes," in (P.H. Winston, ed.) *The Psychology of Computer Vision*, New York; McGraw-Hill, 1975

[Sha79] *J.F. Shapiro*, Mathematical Programming: Structures and Algorithms, New York, John Wiley and Sons, 1979.

[Sly82] R.V. Slyke, "Redundant set covering in telecommunications networks," *Proc. IEEE 1982 Intern. Large Scale Systems Symposium*, Virginia Beach,

Virginia, October, 1982.

[StR71] A. Stefanelli, and A. Rosenfeld, "Some parallel thinning algorithms for digital pictures," *Journ. of ACM*, Vol. 18, No.2, April 1971, pp 255-264.

[TBB79] J.M. Tenenbaum, H.E. Barrow, R.C. Bolles, "Prospects for industrial vision", in (G.G. Dodd and L. Rossol, eds.) *Symposium on Computer Vision and Sensor-Based Robots*, General Motors Research Laboratories, Warren, Michigan, New York: Plenum Press, 1979.

[Ull79] S. Ullman, "Relaxation and constrained optimization by local processes," *Comput. Graphics Image Processing*, vol. 11, pp. 115-125, 1979.

[Wal72] D. L. Waltz, "Generating semantic descriptions from drawings of scenes with shadows," Technical Report AI271, M.I.T., 1972.

[Wal75] D. L. Waltz, "Understanding line drawings of scenes with shadows," in *The Psychology of Computer Vision* (P. H. Winston, ed.), McGraw-Hill, New York, 1975.

[WeM78] N. Weisstein, and W. Maguire, "Computing the next step: psychophysical measures of representation and interpretation," in *Computer Vision Systems*, A. R. Hanson and E. M. Riseman, Eds. New York: Academic, 1978, pp. 243-260.

[Wil78] T. J. Willett, "Hardware Implementation of Image Processing Overlays: Relaxation", *Proc. DARPA Image Understanding Workshop*, (L. S. Baumann, ed.), Science Applications, Inc., Arlington, Virginia, 1978.

[Yam79] H. Yamamoto, "A method of deriving compatibility coefficients for relaxation operators," *Comput. Graphics Image Processing*, vol. 10, pp. 256-271, 1978.

[ZHR77] S. W. Zucker, R. A. Hummel, and A. Rosenfeld, "An application of relaxation labeling to line and curve enhancement," *IEEE Trans. Comput.*, vol. C-26, pp. 394-403, 1977.

[ZLM81] S.W. Zucker, Y.G. Leclerc, and J.L. Mohammed, "Continuous relaxation and local maxima selection: conditions for equivalence," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-3, 2, pp. 117-128, 1981.