AN ALGORITHM WHICH AUTOMATICALLY CONSTRUCTS DISCRIMINATION GRAPHS IN A VISUAL KNOWLEDGE BASE

by
Jan A. Mulder

Department of Mathematics, Statistics,
and Computing Science
Dalhousie University
Halifax, Nova Scotia
Canada B3H 3J5

Abstract

Model-based vision systems must be capable of dealing with large quantities of hypothetical interpretations. Hypothesise-and-test methods often lead to combinatorial explosions of possible interpretations. Discrimination graphs prevent such explosions by imposing a hierarchical organisation on the domain of interpretation. Such an organisation effectively reduces the number of interpretations that the system has to deal with. This paper presents an algorithm which automatically constructs discrimination graphs.

1 Introduction

If we are to develop a "general-purpose* vision system which can take as input a digitised picture of any natural scene and which produces as output a meaningful description of such a scene, then we must equip the system with a representation which allows for a quick and smooth access to domain-specific knowledge. Most model-based vision systems obtain access to the knowledge base of a particular domain after a segmentation process has scanned the image in search for particular features. The description of these features imposes constraints on the interpretations possible. Features, constraints, and interpretations can be represented as a graph (or hypergraph) with the features (n) as variables, each with an associated domain of interpretations *(d)* and the constraints (e) as relations. The problem of finding globally consistent interpretations for the image then becomes the problem of finding all possible n-tuples such that each n-tuple is an instantiation of all n variables satisfying the relations. This problem is equivalent to the constraint satisfaction problem [5]. The existence of algorithms which can solve this problem in less than exponential time has been questioned. Depth-first backtracking, for instance, is of $0(e<f)$ in its worst case performance [5]. Such an algorithm causes a major problem for a general-purpose vision system in which we can expect domain sixes to be very large, because no advance knowledge exists of the domain of interpretation.

The research for efficient solutions to this "exponentiality" problem has taken several directions. The search for more efficient algorithms, among other things, has led to algorithms that do "intelligent backtracking" [I] and to so-called network consistency algorithms |3|. The latter is a group of polynomial time algorithms which do not necessarily solve the constraint satisfaction problem, but which eliminate all local inconsistencies that cannot participate in a global solution. Network consistency algorithms will generally reduce the overall domain sise. This makes them attractive as pre-processors for algorithms such as depth-first backtracking. Recent research has focused on a more effective organisation of the domain of each variable [7]. A hierarchical organisation can be imposed on the domain by means of discrimination graphs.

2 Discrimination graphs

Discrimination graphs are based on the assumption that we can classify image features in one or more dimensions (e.g. shape, texture), the result of which is a finite number of categories for each dimension. Discrimination graphs (DG's) are based on a (potentially unnatural) categorisation of object classes that belong to a particular image feature category. A DG is a directed, acyclic graph. Every source node of the graph is an abstract object class which intensionally represents all the elementary object classes that belong to a particular image feature category. The leaves of the graph are elementary object classes. Intermediate nodes represent subsets of the set of objects represented by their ancestor(s).

This paper does not provide an in-depth discussion of DG's. The idea of imposing a hierarchical domain organisation by means of DG's was first presented in [7], an in-depth discussion is provided in (8]. Intuitively, DG's are useful because image features can have a wide variety of interpretations for which no hierarchical organisation exists. For instance, a collection of green pixels in an aerial photograph can be farm land, a golf course, or even the astroturf in a stadium. Because no natural categorisation scheme exists for such interpretations we cannot use a specialisation hierarchy as a means of replacing these interpretations by more abstract ones. DG's enable us to impose a hierarchical organisation on an arbitrary set of natural objects.

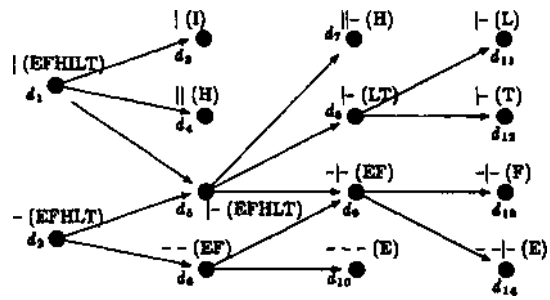

Figure 1: an example of a discrimination graph

Figure 1 shows a DG for a simple character recognition problem. The image consists of one character which is a letter containing horisontal and vertical line segments only. Each letter is classified in two different ways: by the orientation of strokes and by the number of strokes in a particular orientation (table 1). Recognition takes place in steps. The strokes are the image features and they are investigated one at a time. The DG constructed for this situation reflects this process. The source nodes of the graph are on the left and they represent the possible set of interpretations after investigation of one stroke. Their successors represent the (sub)set of interpretations after two strokes have been investigated. The leaves represent the interpretations possible after all strokes have been looked at. All interpretations (abstract or natural) are labeled (e.g. d_1). The constraints associated with each interpretation are shown as well. For instance, interpretation do

represents all letters with at least one horisontal and at least one vertical stroke. The letters E, F, H, L, and T satisfy this constraint.

DG's have been implemented in Mapeee-3, a sketch map inter=-pretation program (8). Sketch maps are a useful domain for studying the theory, design, implementation, and evaluation of different knowledge representation schemes [4,2,8].

letter	vertical strokes (|)	horizontal strokes (-)
E	1	3
F	1	2
H	2	1
I	1	0
L	1	1
T	1	1

Table 1: frequency and orientation of strokes

DG's allow HI to replace a large set of natural interpretations in the domain of each variable by a muck smaller set of abstract interpretations. A network consistency algorithm called *hierarchical arc consistncy* has been designed and implemented which operates in hierarchical domains [6j. This algorithm tests the consistency between the interpretations of adjacent variables. Inconsistent interpretations are replaced by their successors in the graph after which the consistency test is repeated. This test-replacement sequence continues either until a consistent interpretation is found, or until each variable's domain is empty. Only a very small part of the interpretations in the graph need to be considered in this replacement operation. The worst case complexity of hierarchical arc consistency is $0(ed^3)$. However, a few simple constraints improve this to $O(log_d(e + (3n/2)))$ [6]. The combination of hierarchical arc consistency with DG's provides a quick and smooth transition from abstract and domain-independent interpretations near the top of the graph to the more domain-dependent interpretations at the leaves.

3 Constructing discrimination graphs

How do we construct a graph such as the one in figure 1? The graph is based on S premises: an explicit representation for each natural interpretation (the leaves), an explicit representation for each possible combination of constraints, and an explicit representation for. the addition of a single constraint (the arcs). This results in the creation of 14 feature categories $(F_1, \ldots\ldots, F_{14})$ each with a set of possible interpretations. For each one of these sets we create a single abstract interpretation: d_1 corresponding to F_1, d_2 corresponding to F_2 etc. The feature categories are also inter-dependent. For instance, the interpretation* of F_t can only result from the interpretations of F_2. For the construction of figure 1 we therefore need several knowledge sources, among which are an explicit representation of the feature category dependencies and the information represented in table 1. The former tells us which transitions between feature categories are legitimate, the latter informs us which transitions are meaningful. Based on this information we can construct figure 1.

The presence of so many abstract interpretations, among other thinp, requires an algorithm that automatically construct* DG's. Indeed, the value of using DG's would decrease if this could not be done. However, we would like such an algorithm to be domain-independent, that is, not dependent on an intimate knowledge of feature categories and their dependencies. The graph in figure 1 is optimal in the sense that all possible combinations of interpretations are explicitly represented and the addition of a new constraint in the imag leads to a single transition in the graph. Perhaps, it is possible to design a domain-independent algorithm which does not necessarily lead to an optimal graph, but which still constructs a usable one.

For one thing, the graph in figure 1 is redundant in the sense that several feature categories have identical interpretation sets. The graph is constructed such that the relation between offspring and their parents is the subset relation. Subsets represented by different offspring may overlap. The algorithm we propose makes no assumptions about the feature categories. It orders the categories by the setsise of their interpretations and it merges categories with identical interpretations. An abstract interpretation is created for each category. The abstract interpretations with the largest setsise become the source nodes of the DG. The graph created is binary and the subsets represented by the two offspring of each abstract interpretation are disjoint.

The start point for construction is a given feature dimension *FD* consisting of n categories *(F)* with a total of *$* (natural) interpretations (5). Each category $F\ (1 \leq i \leq n)$ has a set /, of possible interpretations (/, 6 5). The sise of /< may vary from category to category and a particular interpretation $Sj\ (1 \leq j \leq s)$ may occur in more than one category.

The construction algorithm operates as follows:

1. Merge the categories with identical interpretation sets. *F* is now of sise $m\ (m \leq n)$.
2. For each *F,* $(1 \leq i \leq m)$ create an abstract interpretation A,. A, intensionally represents the set *U* of possible interpretations for *F,*
3. Order the abstract interpretations by the setsise of their natural interpretations. The interpretation with the largest setsise comes first. Call this ordered list *L.* The abstract interpretations form the elements in this list.
4. Do until *L* is empty:
   Take the first element / from *L* and delete it from *L.*
   If the setsise of / > 1 then do:
   Find the element in *L* that represents the largest subset of /. Call this element I_1.
   If the setsise of I_1 is less than the integer part of half the sise of /
   then execute step i
   else execute step ii.

   i. Split the set represented by / into two disjoint subsets of approximately equal sise, and create two new abstract interpretations *I** and 14 Each of these interpretations represents one of the subsets. Establish a link from 1 to its two siblings and insert the siblings into L at a location that corresponds to their setsise.
   ii. Find the interpretatiox I_2 representing the exclusion of the sets represented by *I* and /\. If this interpretation does not exist then create it and insert it into *L* at the proper location. Establish a link from / to its siblings I_1 and I_2.

Application of the construction algorithm to the character recognition problem results in the graph illustrated in figure 2. Identical interpretation sets cause a reduction in the number of feature categories. Figure 2 shows the 11 interpretations created with the following correspondences (in parentheses) to figure 1: d_1 $(d_x)_t$ d? (d_3), d_3 (d_2, d_6), d_4, d_6 (d_4, d_7), d_9 (d_8), d_7 (d_9), d_8 (d_{11}), d_9 (d_{12}), <*io (4io><*uJt and *n (<*isj- the interpretation with largest setsise, d_{1t} becomes the source node. d_1 is linked with d_2 and $d_{,}$, both of which already exist. Actually, the algorithm uses existing interpretations almost all the time with the exception of a\ which is newly created. All the splits are caused by step 4ii. The set EFHLT could also have been split into the subsets EFH and LT.

Figure 2 is no longer an optimal graph. All possible combinations of interpretations are still explicitly represented, but the addition of a new constraint now sometimes requires more than one transition in the graph. For example, the transition from d\ to d_4 in figure 1 now requires 3 transitions from d_l through $d3$ and d_4 to $d_{3,}$. The trade-off between figure 1 and 2 is the traditional space-time trade-off. Space-wise figure 2 is more efficient, but accessing the proper interpretation (sometimes) requires more time.

This paper cannot provide an extensive discussion of construction issues. Many of these are addressed in [8]. Suffice it to say here, that the construction algorithm presented above can also be used in knowledge bases which are organised along composition and specialisation hierarchies. In the Mapsee-3 program, for instance, objects are represented at different levels of composition and specialisation. The natural interpretations for each feature category are the leaves of a composition hierarchy. The construction algorithm constructs a DG at the composition leaf level. A projection algorithm then iteratively projects this graph from one level of composition onto the next level up, thereby constructing an abstract composition hierarchy and a DG at each level of composition. This projection algorithm is also discussed in (8|.

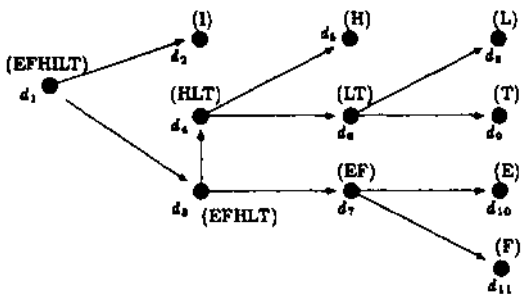

Figure 2: an automatically constructed discrimination graph

4 Conclusion

Discrimination graphs are a useful tool for imposing a hierarchical organisation on domains which cannot be naturally organised in this way. In combination with a hierarchical arc consistency algorithm, discrimination graphs provide a quick and smooth access to domain-specific knowledge. This paper has presented an algorithm that automatically constructs discrimination graphs from a set of naturally interpretable feature categories.

5 Acknowledgements

Comments on a draft of this paper by Jay G licks man and Bill Havens are gratefully acknowledged. This research was supported by NSERC operating grant A0948.

References

[1] R. M. Haralick and G. L. Elliott, "Increasing Tree Search Efficiency for Constraint Satisfaction Problems", *Artificial Intelligence,* vol. 14, pp. 263-313, 1980.

[2] W. S. Havens and A. K. Mackworth, "Representing Knowledge of the Visual World", *IEEE Computer,* vol. 16, no 10, pp. 90-98, 1983.

[3] A. K. Mackworth, "Consistency in Networks of Relations", *Artificial Intelligence,* vol. 8, no 1, pp. 99-118, 1977.

[4] A. K. Mackworth, "Vision Research Strategy: Black Magic, Metaphors, Mechanisms, Miniworlds, and Maps", in *Computer Vision Systems,* eds. A. R. Hanson and E. M. Riseman, Academic Press, New York, pp. 53-60, 1978.

[5] A. K. Mackworth and E. C. Freuder, "The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems", *Artificial Intelligence,* vol. 25, pp. 65-74, 1985.

[6] A. K. Mackworth, J. A. Mulder, and W. S. Havens, "Hierarchical Arc Consistency: Exploiting Structured Domains in Constraint Satisfaction Problems", *Computational Intelligence,* vol. 1, no 3-4, pp. 118-126, 1985.

[7] J. A. Mulder, "Using Discrimination Graphs to Represent Visual Interpretations that are Hypothetical and Ambiguous", *Proc. of the 9th Int. Joint Conf. on Artificial Intelligence,* Los Angeles, pp. 905-907, 1985.

[8] J. A. Mulder, "Using Discrimination Graphs to Represent Visual Knowledge", TR-85-14, Department of Computer Science, University of British Columbia, Vancouver, Canada, 1985.