THE FIXED POINTS OF IMAGES AND SCENES

Simon Kasif
Azriel Rosenfeld

Computer Vision Laboratory, Computer Science Center
University of Maryland
College Park, MD 20742

Abstract

This paper describes a general method of consistent labelling based on fixed point theory. We show that a large class of labelling problems can be formulated as the problem of finding the fixed point of a function defined on a semilattice. Moreover, in many cases a simple iteration process is guaranteed to give the fixed point within a limited number of steps. The standard "discrete relaxation" process is a special case of this approach, and other cases can also be defined.

1. Introduction

Since the mid-1970's, there has been considerable interest in a class of cooperative processes, known as "relaxation" processes, for solving consistent labeling problems [1-3]. The basic idea of this approach is as follows: let there be given a set of interrelated objects, such as regions in an image; a set of labels that each object can have; and a set of "compatible pairs" of labels that each pair of objects can have. Then by repeatedly discarding incompatible labels (i.e., a label is discarded from an object if, for some other object, there is no label compatible with it), we can often reduce the labelling ambiguity, and in many cases can obtain a unique labeling. This process is known as "Waltz filtering" or "discrete relaxation."

In this paper we show that discrete relaxation is a special case of a general class of processes for finding fixed points of functions defined on a semilattice. The general fixed point problem is formulated in Section 2, and some examples, including the case of discrete relaxation, are presented in Sections 3 and 4. It should be mentioned that our approach can also be generalized to non-discrete situations; this will be the subject of a forthcoming paper.

2. Fixed points of functions on semilattices

The relation $\leq$ on a set S is called a partial order if it is reflexive, antisymmetric, and transitive - i.e., if for all x, y, z in S we have

$x \leq x$

$x \leq y$ and $y \leq x$ imply $x = y$

$x \leq y$ and $y \leq z$ imply $x \leq z$

The pair $(S,\leq)$ is called a partially ordered set, or poset.

If T is any subset of S, $t \in T$ is called a least element of T if $t \leq w$ for all $w \in T$. Evidently t is unique, since if t and t' are both least elements of T we have $t' \leq t$ and $t \leq t'$. The least element of S, if it exists, will be denoted by e.

If T is any subset of S, $x \in S$ is called an upper bound of T if $t \leq x$ for all $t \in T$. Moreover, $y \in S$ is called the least upper bound (LUB) of T if it is an upper bound, and if $y \leq x$ for any upper bound x. (Evidently y is unique, since it is the least element of the set of upper bounds of T.) $(S,\leq)$ is called an upper semilattice if for all u,v in S, the set {u,v} has an LUB.

A chain in S is a sequence of elements $x_0, x_1, \ldots$ such that $x_0 \leq x_1 \leq \ldots$ We will call $(S,\leq)$ complete if any chain has an LUB.

Let $(A,\leq)$ and $(B,\leq)$ be posets (to simplify the notation we use the same symbol for both partial order relations). A function $f:A \to B$ is called monotonic if for all x,y in A we have

$x \leq y$ implies $f(x) \leq f(y)$

Let $(A,\leq)$ and $(B,\leq)$ be complete upper semilattices; then f is called upper semicontinuous if it is monotonic, and for all chains $\{x_0, x_1, \ldots\}$ in A we have

$$\mathrm{LUB}\{f(x_0), f(x_1), \ldots\} = f(\mathrm{LUB}\{x_0, x_1, \ldots\})$$

(Note that since f is monotonic we have $f(x_0) \leq f(x_1) \leq \ldots$, so that the sequence on the left hand side is a chain.)

Let x be a function from a set A into itself; we recall that x is called a fixed point of f if $f(x)=x$. If $(A,\leq)$ is a poset, y is called the least fixed point of f if it is a fixed point, and $y \leq x$ for all fixed points x of f. (Note that y is unique.)

The support of the National Science Foundation under Grant MCS-79-23422 is gratefully acknowledged, as is the help of Janet Salzman in preparing this paper.

Theorem 1. Let $(S,\leq)$ be a complete upper semilattice with least element e, and let $f:S \to S$ be upper semicontinuous; then f has a least fixed point, namely $\mathrm{LUB}\{e, f(e), f^2(e), \ldots\}$.

Proof: Since e is the least element of S we have $e \leq f(e)$, and since f is monotonic, it follows that $f(e) \leq f^2(e) \leq f(f(e))$, $f^2(e) \leq f^3(e)$, and so on, so that the sequence in braces is a chain, and hence has an LUB, call it E. Since f is upper semicontinuous, we have

$$f(E) = f(\mathrm{LUB}\{e, f(e) f^2(e), \ldots\}) = \mathrm{LUB}\{f(e), f^2(e), f^3(e), \ldots\}$$
$$= \mathrm{LUB}\{e, f(e), f^2(e), \ldots\} = E$$

so that E is a fixed point of f. (Since e is the least element of S, for any $T \subseteq S$ we have $\mathrm{LUB}(T \cup \{e\}) = \mathrm{LUB}(T)$.) Finally, let E' be any fixed point of f. Since $e \leq E'$, we have $f(e) \leq f(E') = E'$, $f^2(e) \leq f^2(E') = E'$, ..., so that E' is an upper bound of $\{e, f(e), f^2(e), \ldots\}$; hence $E \leq E'$, since E is the LUB, which proves that E is the least fixed point of f. //

Note that if a chain is finite, say $x_1 \leq x_2 \leq \ldots \leq x_n = x_{n+1} = \ldots$, its LUB is equal to its greatest element x_n. We say that $(S,\leq)$ has height $\leq k$ if all chains have at most k+1 distinct elements. Evidently, if $(S,\leq)$ has height $\leq k$, and $f:S \to S$ is monotonic, it is also upper semicontinous.

Corollary 2. In Theorem 1, if S has height $\leq k$, the least fixed point of f can be computed in at most k steps.

Proof: Start with e and apply f repeatedly, yielding the chain $e \leq f(e) \leq f^2(e) \leq \ldots$ In any chain, if $x_i = x_j$ for some $i<j$, we must have $x_i = x_{i+1} = \ldots = x_j$ by antisymmetry and transitivity. Since S has height $\leq k$, the elements $e, f(e), f^2(e), \ldots, f^{k+1}(e)$ cannot all be distinct; hence for some $i<j \leq k+1$ we have $f^i(e) = f^j(e)$ (where $f^0(e) \equiv e$), so that by the preceding sentence we have $f^i(e) = f^{i+1}(e) = \ldots = f^j(e)$. Hence $f^{j-1}(e) = f^j(e)$, and by repeatedly applying f to both sides we have $f^j(e) = f^{j+1}(e) = f^{j+2}(e) = \ldots$, so that $f^i(e)$ is the greatest element of the chain. Since $i<k+1$, we see that at worst $f^k(e)$ is the greatest element, making it the LUB of the chain and so the least fixed point of f. //

3. Example 1: Discrete relaxation

In this section we show how the standard discrete relaxation algorithm is a special case of the fixed point computation process of Corollary 1. In order to do this, we first present a simple extension of our results to power sets.

Let $(S,\leq)$ be a poset, let S^n be the nth power of S (i.e., the set whose elements are n-tuples of elements of S), and let $\leq$ be the relation on S^n defined by $f(x_1, \ldots, x_n) \leq (y_1, \ldots, y_n)$ iff $x_i \leq y_i$ for all $1 \leq i \leq n$. Readily, this new $\leq$ is a partial order relation on S^n. Moreover, if S has least element e, then S^n has least element $(e, \ldots, e)$.

Proposition 1. If S has height $\leq k$, then S^n has height $\leq kn$.

Proof: In any chain of distinct elements, at least one term must change from one element to the next. The sequence of values of each term, eliminating repetitions, is a chain of distinct elements in S; hence the number of changes in each term is at most k. Thus the total number of changes is at most kn. //

Let f_i be a function from S^n into S, $1 \leq i \leq n$, and let f be the function from S^n into S^n defined by $f(x_1, \ldots, x_n) = (f_1(x_1, \ldots x_n), \ldots, f_n(x_1, \ldots, x_n))$. Evidently, if each f_i is monotonic, so is f, since we have $(x_1, \ldots, x_n) \leq (y_1, \ldots, y_n)$ implies $f(x_1, \ldots, x_n) = (f_1(x_1, \ldots, x_n), \ldots, f_n(x_1, \ldots x_n)) \leq (f_1(y_1, \ldots, y_n), \ldots, f_n(y_1, \ldots, y_n)) = f(y_1, \ldots, y_n)$.

Proposition 2. Let S be a complete upper semilattice; then if each f_i is upper semicontinuous, so is f.

Proof: Let $\{(x_{11}, \ldots, x_{1n}), (x_{21}, \ldots, x_{2n}), \ldots\}$ be a chain in S^n. By the definition of $\leq$ on S^n, it is easily seen that $\mathrm{LUB}\{(x_{11}, \ldots, x_{1n})(x_{21}, \ldots, x_{2n}) \ldots\} = (\mathrm{LUB}\{x_{11}, x_{21}, \ldots\}, \ldots, \mathrm{LUB}\{x_{1n}, x_{2n}, \ldots\})$. [This follows from the fact that $(y_1, \ldots, y_n)$ is an upper bound of $\{(x_{11}, \ldots, x_{1n}), (x_{21}, \ldots, x_{2n}), \ldots\}$ if and only if y_1 is an upper bound of $\{x_{11}, x_{21}, \ldots\}, \ldots, y_n$ is an upper bound of $\{x_{1n}, x_{2n}, \ldots\}$.] Hence $\mathrm{LUB}\{(f(x_{11}, \ldots, x_{1n}), f(x_{21}, \ldots, x_{2n}), \ldots\} = \mathrm{LUB}\{(f_1(x_{11}, \ldots, x_{1n}), \ldots, f_n(x_{11}, \ldots, x_{1n})), (f_1(x_{21}, \ldots, x_{2n}), \ldots, f_n(x_{21}, \ldots, x_{2n})), \ldots\}$

$$= (\mathrm{LUB}\{f_1(x_{11}, \ldots, x_{1n}), f_1(x_{21}, \ldots, x_{2n}), \ldots\}, \ldots,$$
$$\mathrm{LUB}\{f_n(x_{11}, \ldots, x_{1n}), f_n(x_{21}, \ldots, x_{2n}), \ldots\})$$
$$= (f_1(\mathrm{LUB}\{(x_{11}, \ldots, x_{1n}), (x_{21}, \ldots, x_{2n}), \ldots\}, \ldots,$$
$$f_n(\mathrm{LUB}\{(x_{11}, \ldots, x_{1n}), (x_{21}, \ldots, x_{2n}), \ldots\})$$
$$= f(\mathrm{LUB}\{(x_{11}, \ldots, x_{1n}), (x_{21}, \ldots, x_{2n}), \ldots\}). \quad //$$

We now apply these ideas to the discrete relaxation process. Let $O_1, \ldots, O_n$ be a set of objects and let $L = \{\ell_1, \ldots, \ell_m\}$ be a set of labels. Let S be 2^L (the set of subsets of L), and let $\leq$ be $\supseteq$, i.e., if A,B are subsets of S we write $A \leq B$ iff $A \supseteq B$. Evidently, S is an upper semilattice under this $\leq$; L is its least element; and it has height $\leq m$ (no chain of subsets under $\supseteq$ can be longer than m+1), so that in particular it is complete.

Suppose we are given, for each pair of objects O_i, O_j, a set $C_{hk}(i,j)$ of allowable pairs of labels (L_h, L_k). The interpretation of $C_{hk}(i,j)$ is that O_i cannot have label ℓ_h unless each O_j has some

label ℓ_k such that $(\ell_h,\ell_k)\in C_{hk}(i,j)$. Let f_i be the function from S^n into S defined as follows: For any n-tuple of label sets $(L_1,\ldots,L_n)$ we have $\ell_h\in f_i(L_1,\ldots,L_n)$ iff for each j there exists an $\ell_k\in L_j$ such that $(\ell_h,\ell_k)\in C_{hk}(i,j)$. Evidently, f_i is monotonic. By Proposition 1, S^n has height $\leq mn$, and thus f_i is also upper semicontinuous. By Proposition 2, it follows that $f:S^n\to S^n$, defined by $f(L_1,\ldots,L_n)=(f_1(L_1,\ldots,L_n),\ldots,f_n(L_1,\ldots,L_n))$, is upper semicontinuous. Thus f has a least fixed point, namely LUB{(L,...,L),f(L,...,L),f^2(L,...,L), ...} (Theorem 1), and it can be computed by applying f at most mn times to (L,...,L) (Corollary 1). Note that this is exactly the standard discrete relaxation algorithm, in which we start with (L, ...,L) and repeatedly discard non-allowable labels, "in parallel," until no further change occurs, i.e., until a fixed point is reached.

4. Example 2: Conditional functions

We now give another general example involving a rather different class of constraints.

Let $(S,\leq)$ be a poset. By a proposition on S we mean a function $P:S\to\{e, true, false\}$, where we define a partial order relation on the range of P by specifying that $e\leq true$ and $e\leq false$, but true and false are not related. A conditional function $F:S\to S$ is defined inductively as follows:

a) Any function $f: S\to S$ is a conditional function

b) If P is a proposition and F_1,F_2 are conditional functions, then F defined by

$$F(x)=e \text{ if } P(x)=e$$

$$F(x)=F_1(x) \text{ if } P(x)=true$$

$$F(x)=F_2(x) \text{ if } P(x)=false$$

is a conditional function.

Proposition 3. If P,F_1, and F_2 are monotonic, F (defined as in (b) above) is also monotonic.

Proof: Let $x\leq y$. If $P(x)=e$, we have $F(x)=e$, and since e is the least element of {e, true, false}, we automatically have $F(x)\leq F(y)$. If $P(x)=true$, we have $F(x)=F_1(x)\leq F_1(y)$ since F_1 is monotonic. But since P is monotonic we must have $P(y)=true$, so that $F(y)=F_1(y)$, proving $F(x)\leq F(y)$; and analogously if $P(x)=false$. //

Proposition 4. If $(S,\leq)$ is a complete upper semilattice, and $P,F_1$and F_2 are upper semicontinuous, so is F.

Proof: Let$\{x_1,x_2,\ldots\}$ be a chain. It is easily seen that if $P(x_i)=e$ for all i we have $P(LUB\{x_1,x_2,\ldots\})=LUB\{P(x_1),P(x_2),\ldots\}=e$. Similarly, if $P(x_i)=true$ for any i, we have $P(x_j)=true$ for all $j\geq i$ and $P(LUB\{x_1,x_2,\ldots\})=true$; and analogously, if $P(x_i)=false$ for any i. Similar remarks apply to F_1 and F_2. In each of these cases, it follows that F is upper semicontinuous. //

It follows that if F is any conditional function, and the f's, P's, F_1's and F_2's involved in the definition of F are all monotonic or upper semicontinuous, so is F.

We can now consider a general class of constraint satisfaction problems in which the constraints are specified by arbitrary conditional functions, rather than in terms of allowable pairs as in the case of discrete relaxation. Using conditional functions, we can concisely express constraints such as "if O_i has label ℓ_n then O_j has label ℓ_k," which would be cumbersome to express in terms of allowable pairs. Let the constraints on O_i be defined by a monotonic conditional function $F_i:S^n\to S$, where $S=2^L$, and let $F:S^n\to S^n$ be defined in terms of the F_i, just as in Section 3.* Since S^n still has height $\leq mn$, we can still find the least fixed point of F by at most mn applications of F to (e,...,e)≡(L,...,L), exactly as in Section 3. In other words, if we reformulate the standard discrete relaxation algorithm as repeated application of a monotonic function F to the least element of an upper semilattice, we see that discrete relaxation is just one example of a wide class of constraint satisfaction algorithms, characterized by the fact that they can be expressed in terms of fixed points of monotonic functions.

References

1. A. Rosenfeld, R. A. Hummel, and S. W. Zucker, Scene labeling by relaxation operations, IEEE Trans. Systems, Man, Cybernetics 6, 1976, 420-433.

2. R. M. Haralick and L. G. Shapiro, The consistent labeling problem, IEEE Trans. Pattern Analysis Machine Intelligence 1, 1979, 173-184, and 2, 1980, 193-203.

3. L. S. Davis and A. Rosenfeld, Cooperating processes for low-level vision: a survey, Artificial Intelligence 17, 1981, 245-263.

4. R. D. Smart, Fixed Point Theorems, Cambridge University Press, Cambridge, UK, 1974.

*For example, the function F_i might be defined as follows:

$\ell_h\in F_i(L_1,\ldots,L_n)$ iff either $P_i(L_1,\ldots,L_n)=e$, or $P_i(L_1,\ldots,L_n)=true$ and

$\ell_h\in F_{il}(L_1,\ldots,L_n)$, or $P_i(L_1,\ldots,L_n)=false$ and $\ell_h=F_{i2}(L_1,\ldots,L_n)$, where P_i is a proposition, and F_{il},F_{i2} are (conditional) functions, from S^n into S.

A NEW PRODUCT GRAPH BASED ALGORITHM FOR SUBGRAPH ISOMORPHISM

Funso A. Akinniyi and Andrew K.C. Wong
Department of Systems Design
University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1

ABSTRACT

This paper presents a new algorithm for detecting subgraph isomorphisms for pairs of graphs. This algorithm entails a tree searching procedure over the projections of the implicit product of the two graphs. It utilizes the minimum number of neighbours of the projected graphs to detect infeasible subtrees. The algorithm in comparison to that presented in [9] is more efficient in its storage space utilization and average running time. It does not suffer from the ambiguity which arises in [9], when cyclic graphs are matched.

I. INTRODUCTION

The graph monomorphism problem, also known as the subgraph isomorphism problem, is one which finds whether or not there exists a one-to-one vertex mapping between two graphs while preserving incidence relations. The graphs are assumed to be finite. The formalism and terminology used are based on Berge [3] and Ghahraman et al. [11].

Graph monomorphisms have wide applications in areas such as scene analysis, information storage and retrieval, chemical documentation and network analysis. In scene analysis, a graph morphism task is to compare an image to a reference scene for purposes of recognition and/or classification. Ideally, the graph morphism problem is that of isomorphism. However, if occlusion is possible, then graph monomorphism or even largest common subgraph isomorphism is more applicable. The transportation problem is that of finding an optimal cost graph monomorphism [12] in which each itinerary is mapped onto a transport network to achieve the best itinerary.

It is well known that the graph monomorphism problem belongs to the class of NP-complete problem [10]. Problems in this class are inherently intractable, that is, any algorithm designed to solve the general problem will require exponential time complexity. Over the past two decades, considerable efforts have been devoted to the problems related to graph isomorphism. A comprehensive survey of the major works can be found in [16]. Unfortunately, hitherto, no polynomial time algorithm has been found. Attempts to solve the problem have been largely concentrated in four areas:

(1) The design of polynomial time algorithms for special cases of the problem. For instance, the tree isomorphism algorithm which runs in O(n) (pp. 84-86 [1]). Also, there is the O(nlogn)algorithm of Hopcroft and Tarjan [14] for detecting isomorphism of a pair of planar graphs.

(2) The design of polynomial time heuristics which iteratively partitions the vertices of the two graphs into classes and then refine those partitioned classes until a one-to-one vertex mapping is (or is not) achieved. Unfortunately, these heuristics fail for graphs of high orders. The most celebrated example is that of Corneil and Gotlieb [8]. Another, is that of Sussenguth [19] which extends Unqers' isomorphism algorithm [23] to include graph monomorphism.

(3) Theoretical works such as finding the mathematical functions [5, 18, 21] and proving that certain classes of the problem are polynomially equivalent [6].

(4) Use of backtracking for exhaustive search. Algorithms in this category, utilize refinement techniques based on the neighbours of the two graphs to prune the search tree. Notable procedures are the depth first search algorithm of Deo [9]; the distance matrix algorithm of Schmidt [17] (which suffers from its inapplicability to graph monomorphism); Ullmann's subgraph isomorphism [22]; and recently Cheng et al. [6] which utilizes Berztiss' [4] elementary K-formula and Ullmann's tree search to develop a parallel algorithm for subgraph isomorphism.

Recently, Ghahraman et al. [11] proposed a backtracking algorithm based on the product (called the net) of two graphs, namely, the pattern graph which represents the domain of the morphism, and the base graph the range. According to this algorithm, the Star Pattern Subnet (SPS) is formed for each vertex in the net. The feasibility of each vertex is determined by solving the problem of Maximum Matching in a Bipatite Graph (MMBG) associated with the SPS of that vertex. By investigating the existence of the rooted SPS' whose projections coincide with the pattern graph, the existence of a monomorphism can be ascertained. The utilization of the MMBG criterion (called the strong necessary condition in [11]) leads to an effective and fast pruning of the search tree with the result that monomorphisms are found near the root. However, the iterative solution of the MMBG problem does not enhance the overall efficiency of

CH1891-1/83/0000/0457$1.00 © 1983 IEEE