

On Some Tractable Cases of Logical Filtering

T. K. Satish Kumar and Stuart Russell

Computer Science Division
University of California, Berkeley
{tksk, russell}@eecs.berkeley.edu

Abstract

Filtering denotes any method whereby an agent updates its *belief state*—its knowledge of the state of the world—from a sequence of actions and observations. In *logical filtering*, the belief state is a logical formula describing the possible world states. Efficient algorithms for logical filtering bear important implications on reasoning tasks such as planning and diagnosis. In this paper, we will identify classes of transition constraints that are amenable to compact and indefinite filtering—presenting efficient algorithms wherever necessary. We will first show that connected row-convex (CRC) constraints are amenable to efficient filtering when path-consistency is enforced in appropriate steps. We will then extend this theory to provide a filtering algorithm based on repeatedly enforcing path-consistency and embedding the domain values of the related variables in tree structures to guarantee global consistency. Finally, we will identify and comment on the problem of multi-agent localization as a potential application of the theory developed in the paper (under some reasonable assumptions).

Introduction

When an agent operates in a partially observable environment, it must maintain a representation of its knowledge about the world. Filtering denotes any method whereby an agent updates its belief state—its knowledge of the state of the world—from a sequence of actions and observations. In stochastic models, for example, the *Kalman filter* (Kalman 1960) maintains a multivariate Gaussian belief state over n system variables, assuming linear Gaussian transition and observation models. In each step of the Kalman filter, the cost of updating the belief state is $O(n^3)$, and the space requirement for maintaining the belief state is $O(n^2)$. Since these costs do not depend on the length of the observation sequence, a Kalman filter can run indefinitely. In logical domains, however, the belief state is best represented as a logical formula describing the possible world states; and efficient *logical filtering* refers to the task of having to maintain a compact representation of the belief state even when we have to deal with a potentially unbounded sequence of actions and observations.

In the most general version of the logical filtering problem, the *initial state* may be only partially known; the *transition model* (which allows for actions by the agent) may be nondeterministic; and the *observation model* may be nondeterministic and partial—i.e., the agent may not be able to observe the actual state. Filtering is closely related to the computational problems arising in many important contexts: including planning, diagnosis, game playing, etc. In planning, for example, maintaining a compact representation of the reachability information (as in a planning graph) is known to be a crucial factor in the success of many recent planners—whether or not they deal with nondeterminism in the actions and/or the initial state (for examples, see (Nguyen and Kambhampati 2000), (Bryce and Kambhampati 2005) and (Cushing and Bryce 2005)). Very similar issues are also addressed in filtering when nondeterminism is allowed in the initial state, transition model, and the observation model; and in general, any tractable cases of the filtering problem would bear important implications on our ability to efficiently deal with situations where we are required to maintain a compact representation of the belief state (see (Amir and Russell 2003)).

The computational costs associated with a filtering algorithm include: (1) the time needed to update the belief state, and (2) the space required to represent it. These complexities depend on: (a) the nature of the uncertainty in the initial state, (b) the nature of the transition model (which describes how the system evolves over time), (c) the nature of the observation model (which describes the way in which the environment generates observations), and (d) the family of representations used to represent the belief state (see (Amir and Russell 2003)). It is well known that even when we restrict ourselves to propositional logic, the general filtering problem is hard; the hardness caused mainly because of the need to represent an exponentially large number of possible world states.

In this paper, we will deal with the filtering problem by abstracting it into a temporally extended constraint satisfaction problem (CSP); in particular, we will identify classes of transition constraints (and observation models) that are amenable to compact and indefinite filtering—presenting efficient algorithms wherever necessary. We will first show that CRC constraints are amenable to efficient filtering when path-consistency is enforced in appropriate steps. We will

ALGORITHM: PATH-CONSISTENCY
INPUT: A binary constraint network $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$.
OUTPUT: A path-consistent network.
(1) Repeat until no constraint is changed:
 (a) For $k = 1, 2, \dots, N$:
 (i) For $i, j = 1, 2, \dots, N$:
 (A) $R_{ij} = R_{ij} \cap \Pi_{ij}(R_{ik} \bowtie D_k \bowtie R_{kj})$.
END ALGORITHM

Figure 1: Shows the basic algorithm for enforcing path-consistency in a binary constraint network. Here, Π indicates the projection operation, and \bowtie indicates the join operation (similar to that in database theory).

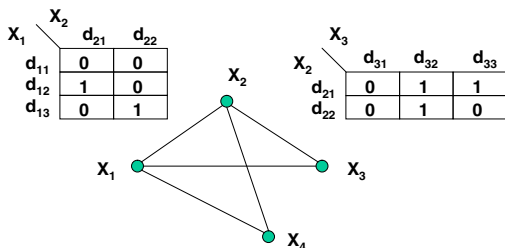


Figure 2: Shows an example of a CSP where the domains of the variables are ordered, and the binary constraints are represented as (0,1)-matrices. The ordered domains of the variables X_1, X_2 and X_3 are respectively $\langle d_{11}, d_{12}, d_{13} \rangle$, $\langle d_{21}, d_{22} \rangle$ and $\langle d_{31}, d_{32}, d_{33} \rangle$ respectively (shown in the figure). For clarity, only two of the constraints are shown in their matrix representations.

then extend this theory to provide a filtering algorithm based on repeatedly enforcing path-consistency and embedding the domain values of the related variables in tree structures to guarantee global consistency. (In turn, we will comment on generalizing this theory to perform filtering via an iterative enforcement of increasing levels of local consistency followed by appropriate geometric embeddings of the domain values of the related variables to guarantee global consistency.) Finally, we will identify and comment on the problem of multi-agent localization as a potential application of the theory developed in the paper (under some reasonable assumptions).

Preliminaries and Definitions

A CSP is defined by a triplet $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{X} = \{X_1, X_2, \dots, X_N\}$ is a set of variables, and $\mathcal{C} = \{C_1, C_2, \dots, C_M\}$ is a set of constraints on subsets of them. Each variable X_i is associated with a discrete-valued domain $D_i \in \mathcal{D}$, and each constraint C_i is a pair $\langle S_i, R_i \rangle$ defined on a subset of variables $S_i \subseteq \mathcal{X}$, called the *scope* of C_i . $R_i \subseteq D_{S_i}$ ($D_{S_i} = \times_{j \in S_i} D_j$) denotes all compatible tuples of D_{S_i} allowed by the constraint. A *solution* to a CSP is an assignment of values to all the variables from their respective domains such that all the constraints are satisfied.

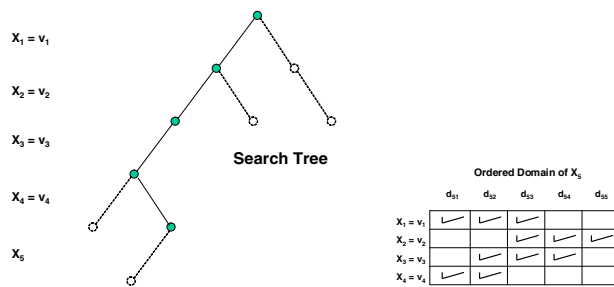


Figure 3: The left side of the figure shows a partial search tree associated with solving a CSP. The figure illustrates the instant of time when we have successfully instantiated a few of the variables (say X_1 to v_1 , X_2 to v_2 , X_3 to v_3 and X_4 to v_4), and we are searching for a consistent extension to the next variable (say X_5). The right side of the figure shows the domain elements of X_5 that are consistent with the values assigned to the previously instantiated variables ($X_1 = v_1$, $X_2 = v_2$, $X_3 = v_3$ and $X_4 = v_4$); tick marks indicate consistent combinations, and blanks indicate inconsistent combinations. We consider the case when the tick marks appear consecutively in each row (assuming an ordering on the domain values of X_5).

A network of binary constraints is *arc-consistent* if and only if for all variables X_i and X_j , and for every instantiation of X_i , there exists an instantiation of X_j such that R_{ij} is satisfied. Similarly, a network of binary constraints is *path-consistent* if and only if for all variables X_i, X_j and X_k , and for every instantiation of X_i and X_j that satisfies the direct relation R_{ij} , there exists an instantiation of X_k such that R_{ik} and R_{kj} are also satisfied. Conceptually, algorithms that enforce path-consistency work by iteratively “tightening” the binary constraints as shown in Figure 1.

The best known algorithm that implements the procedure in Figure 1 exploiting low-level consistency maintenance is presented in (Mohr and Henderson 1986), and has a time complexity of $O(N^3 K^3)$ (where K is the size of the largest domain). This algorithm is optimal, since even verifying path-consistency has the same lower bound. When binary relations are represented as matrices, path-consistency algorithms employ the three basic operations of *composition*, *intersection* and *transposition*. The (0,1)-matrix representation of a relation R_{ij} (denoted $M_{R_{ij}}$) between variables X_i and X_j consists of $|D_i|$ rows and $|D_j|$ columns when orderings on the domains of X_i and X_j are imposed. The ‘1’s and ‘0’s in the matrix respectively indicate the allowed and disallowed tuples.¹ Figure 2 presents an example of a CSP with matrix notations for the constraints.

CRC Constraints

A binary relation R_{ij} represented as a (0,1)-matrix, is *row-convex* if and only if, in each row, all of the ‘1’s are consecutive. It has been shown in (Van Beek and Dechter

¹An extension of this representation mechanism to non-binary constraints is also straightforward.

