Inconsequential improprieties: Filter reduction in probabilistic worlds

Fatemeh Zahra Saberifar[1] and Jason M. O'Kane[2] and Dylan A. Shell[3]

***Abstract*— We wish to minimize the information that a robot maintains to carry out its task. Filters are one way to keep stored state consistent with sensed values, though they may also capture some information about the structure of the world that the robot inhabits. This paper builds on prior work on (improper) filter minimization, but considers a new way to characterize structure in the world. By introducing a probabilistic model, one can define a notion of expected distance between two filters. Then, with such a measure, we pose the question of optimal *lossy* compression in the sense of having minimal expected distance. The problem retains the NP-hardness of the non-probabilistic worst-case minimization and, consequently, in this paper we focus on developing an effective heuristic algorithm. Our results illustrate that, in settings where the probabilities describe evolution of the world's state, the algorithm can do substantially better than existing worst-case minimization techniques oblivious to such structure.**

I. Introduction

Filters are part of the standard estimation machinery used by robots to aggregate, fuse, or otherwise process sensor data. Their outputs are typically used for decision-making. Combinatorial filters [9]—essentially discrete, finite labelled transition systems—are a subclass of filters of practical and theoretical interest. One line of recent research has explored algorithms that operate *on* combinatorial filters [11]–[13]. That is to say, these algorithms take descriptions of filters as inputs and perform a series operations thereupon to produce a variety of outputs, including new filters. This pattern of using software to mutate parts of robot controllers is promising in that it provides an abstract basis for tools to aid roboticists in making design-time decisions.

The present paper continues this line of research, addressing the filter minimization (FM) problem:

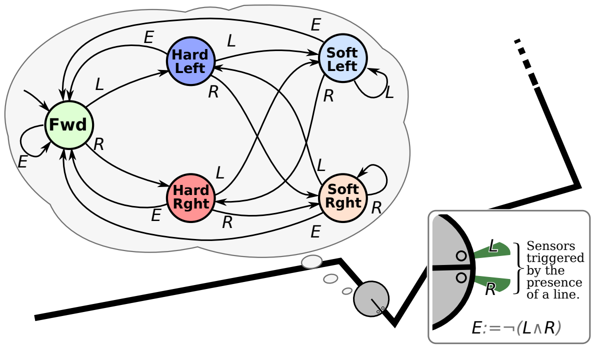
> *Given a combinatorial filter which exhibits some behavior, find the smallest filter that is equivalent.*

This is a reasonable subject to examine because, unlike, say, a Kalman filter where the space complexity of an implementation is $O(1)$ for a given state-space dimensionality, combinatorial filters can be large—so their practicality in certain settings may hinge on the achievement of such reductions. While the measure of 'size' for such filters is natural, the most useful notion of 'equivalence' is, however, far from unequivocally settled. Our work departs from prior results primarily in that we explore a new notion of equivalence between two filters.

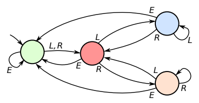
[1]Fatemeh Zahra Saberifar is with Faculty of Mathematics and Computer Science, Amirkabir University of Technology, 424 Hafez Ave, Tehran, Iran.

[2]Jason M. O'Kane is with the Department of Computer Science and Engineering, University of South Carolina, Columbia, South Carolina, USA.

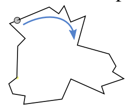
[3]Dylan A. Shell is with the Department of Computer Science and Engineering, Texas A&M University, College Station, Texas, USA.



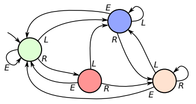
(a) A line-following controller expressed as a filter.



(b) A smaller, four-state improper reduced filter.



(c) A structured environment.



(d) A better filter with 4 states for (c).

Fig. 1: *An illustrative example*: (a) A line-following robot's controller is expressed as a combinatorial filter which distinguishes between sharp and soft turns, the latter being used to fine-tune the gross motions of the former. (b) A filter with only four states output from the reduction techniques in [13]. (c) Since most reflex angles in the environment are $\sim 270^\circ$, a clockwise tracing results in left turns that are structured. (d) A distribution over filter inputs describing such structure leads to the probabilistic reduction result shown. This filter makes fewer errors, in expectation, than that in (b) for the class of scenarios of which (c) is an exemplar.

Filter reduction was first studied in [11], where the problem of optimally minimizing a filter to produce one exactly indistinguishable on all inputs, but of smallest size, was shown to be NP-hard. This result is somewhat surprising, as the problem superficially resembles the classic minimization problem for finite automata solved by Myhill and Nerode [5, pp. 65ff.]. The subsequent work in [13] examined a relaxation of this problem termed *improper filter reduction* in which the equivalence relation on two filters tolerates some mismatches, as expressed by a metric (like the Hamming distance) on strings. But this relaxation turns out to be NP-hard also, and remains so even to approximate.

In this paper we enrich prior models, adding a probability distribution over observations, in order to better express real-world settings. Doing so leads to the analogous problems of probabilistic improper reduction and minimization. Our experimental results show that the additional complication introduced by the probability distribution adds only moderately to the running-time of a heuristic algorithm because a new form of lazy evaluation becomes feasible—though the fundamental hardness concerned remains, of course.

The simple example in Fig. 1 illustrates how probabilistic models manifest themselves in the context of practical improper filter reduction. A robot is tasked with following a line using its floor-facing sensors (see Fig. 1a). Its designer has solved this problem by having turning motions of two varieties: those that make quick large adjustments (hard left and hard right) and those for finer corrections. If one seeks a controller with fewer than five states, the reduction must be improper because we start with five different actions and one must therefore be lost. The question is which one should be discarded? The merger that produces Fig. 1b represents the previous state-of-the-art, wherein the set of input sequences received by the filter (here over *E*, *L*, and *R*) is the domain over which one optimizes to answer the question. When there is additional structure, as is often the case, then one can do better. The environment in Fig. 1c provides one (toy) example: a robot designer, knowing that the robot will inhabit structured environments where left-turns seldom need fine correction, can produce a better four-state filter using the techniques in this paper (see Fig. 1d for such a result).

The contributions of this paper are (i) formulation of the problem of improper filter reduction in probabilistic worlds, along with the introduction of a measure of distance between two filters modulo some probabilistic model; (ii) proof that optimal solution of such problems is NP-Hard; and (iii) the presentation and analysis of a practical algorithm which, though not always optimal, performs well in practice.

The remainder of this paper is organized along these lines: after discussing related work, we formalize the problem and demonstrate its hardness (Section III), describe a heuristic algorithm for that problem (Section IV), and present some experimental results from an implementation of this algorithm (Section V). The paper concludes with a summary of the results and some discussion of future work (Section VI).

II. Related Work

Tracking information about uncertainty in graphs with a combinatorial (or power-set) flavor grew first out of research on manipulation tasks [2], [3]. The term *combinatorial filters*, as we use it to describe discrete estimators, was introduced by LaValle in his more general formulation of the idea of information spaces [8], [9]. Several papers make use of combinatorial filters to reason about the information needed to solve a task, often for simple or minimal robots [15]. Recent years have seen the approach used for diverse applications, including manipulation [6], exploration [7], target tracking [1], [21], navigation [10], [18], validation [19], [20] and some inference tasks on passive combinatorial filters [16], [17]. In a generalization of standard filters, Song and O'Kane [14] consider problems on infinite information spaces.

Discrete filters are interesting because, in addition to being useful to fuse sensor readings directly on-board a robot, they can be used as descriptions (or as specifications) of behavior that are amenable to automated processing. For example, this paper describes an algorithm that mutates a filter given as its input. This view of filters as structured representations was articulated in some detail in [11], which also considered the problem we tackle: that of filter reduction. In light of the hardness result proven in that paper, the recent work of [13] explored the notion of improper reduction—informally, the idea that is willing to tolerate a few errors for a smaller filter—which is extended in this paper. The same authors, and again recently [12], also examined several special-cases of filters and reduction problems, finding that even quite structured instances (e.g., such as filters with a tree form) remain hard to minimize.

As this work extends [13], later sections will elaborate more fully on the relationship of that work to, and precise nature of the extension in, the algorithms developed herein.

III. Definitions and Problem Formulation

This section provides basic definitions and introduces the algorithmic problem we address. We define combinatorial filters and their languages, and then introduce the probabilistic improper filter reduction problem.

A. Combinatorial filters and their languages

Definition 1: A filter $F = (Q, E, \ell, c, q_0)$ is a directed graph in which

1) *the set Q includes the vertices of this graph, called the* states *of the filter,*
2) *the multiset E includes the edges, called the* transitions *of the filter,*
3) *the function $\ell : E \to Y$ assigns to each edge a label called an* observation*, which is drawn from the codomain called the* observation space *Y, so that no two edges with the same origin share the same label,*
4) *the coloring function $c : E \to \mathbb{N}$ assigns a natural number called the* color *to each state, and*
5) *the* initial state *is denoted q_0.*

The idea is that a filter processes a sequence of observations, starting at state q_0 and transitioning along the edge labeled with each successive observation in the sequence. At each state visited in this process, the filter produces as output the color of that state.

For a given observation sequence $s = y_1 \dots y_m$, we write $F(s, q)$ to mean the output produced by F when processing s starting from a given state q, if all of the corresponding edges exist. At the initial state q_0, we use the shorthand $F(s)$ to refer to $F(s, q_0)$.

Note that some states may be 'missing' edges, in the sense that there may be some state-observation pairs (q, y) ,for which q has no out-edge labeled with y. The interpretation is that, based on the structure of the system being modelled, it is known that observation y will not occur when the filter is in state q. For an observation sequence s that might attempt to cross such an edge starting from a state q, we say that s is *invalid* on F from q, and that $F(s, q)$ is not defined.

Definition 2: For a given filter F and a given state q, the set of strings processed by F from q is defined as

$$S(F, q) = \{s \in Y^\star \mid F(s, q) \text{ is defined } \}.$$

In the particular case of the initial state q_0, we write

$$L(F) = S(F, q_0),$$

which we call the language of filter F.

B. The environment model

Our goal is reduce the representation size of a filter while ensuring that the expected difference in outputs between the original and reduced filters remains small. For this notion of expected distance to be well-defined, the input must include a probability distribution over input observation sequences. We define this distribution in terms of a finite state transition system, in the spirit of a Markov chain.

Definition 3: A probabilistic model $P = (V, E_p, \ell_p, v_0)$ is a directed graph in which,

1) *the set V includes vertices called the states of the model,*
2) *the multiset E_p consists of edges, called transitions,*
3) *the function $\ell_p : E_p \to Y \times [0,1]$ assigns to each edge a label, consisting of an observation and a probability,*
4) *the initial state is denoted v_0,*

and for every state that has at least one out-edge, the sum of probability labels is 1.

Note that a probability model is analogous to a filter, except that (a) we attach a probability to each edge, and (b) there are no assigned colors for each state.

This kind of model defines a probability distribution over observation sequences in the following way: For a given observation sequence $s = y_1 y_2 \ldots y_n$, one can begin at the initial state and trace the edges corresponding to the observations, in order. The product of the probabilities of these edges gives the probability that, after n outputs, the observation sequence will have been s; we write $P(s)$ for this probability.

As with filters, in the probability model we do not require an out-edge from each state labeled with each observation. For an observation sequence s that would need to traverse such a missing edge, we define $P(s) = 0$.

C. Defining distance between filters

Before defining the distance metric between pairs of filters, we first consider the difference between pairs of strings. We will be handling sets of strings, some of which may contain strings of arbitrary length. Recognizing the finiteness of any real system owing to fallibility inherent in physical artifacts, we propose to model the likelihood of termination with an exponential factor, for which we adopt the standard term of *discount factor* and use the common symbol γ. This parameter captures, after each step, the (hopefully small) constant independent chance of cessation of the entire process. Now we may define of the discounted Hamming distance of two observation sequences s and s'.

Definition 4: The discounted Hamming distance*, denoted $h^\gamma(s, s')$, between two observation sequences, both of length m, $s = y_1 \ldots y_m$ and $s' = y'_1 \ldots y'_m$, is*

$$h^\gamma(s, s') = \sum_{i=1}^{m} \gamma^i \cdot [y_i \neq y'_i], \qquad (1)$$

in which $\gamma \in (0,1)$ is the discount factor and $[\cdot]$ is the indicator function evaluating to 1 if the given proposition is true and 0 otherwise.

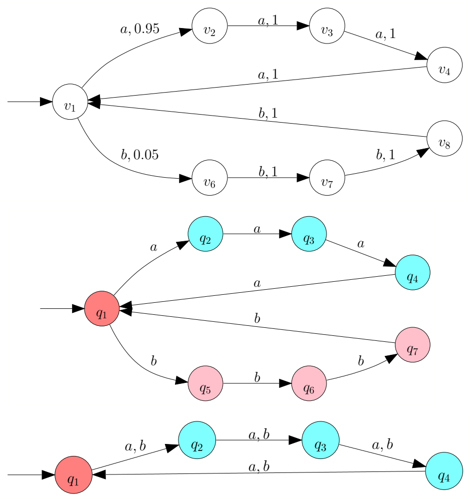


Fig. 2: An example in which the environment structure described by a probabilistic model enables a filter to be reduced effectively. [top] A probabilistic world model consisting of two equal-length cycles, one much more likely to be traversed than the other. [middle] A filter specifying an inference task in which the goal is to determine which of the two cycles is being traversed. [bottom] A reduction of this filter that merges the less likely of the cycles into the more likely one. This reduction is improper, in the sense that this reduced filter cannot guarantee to produce the same outputs as the original, but the expected error remains small.

Next, suppose we are given two filters F_1 and F_2 with the same observation space Y, along with a probabilistic model P. We define a metric to measure the distance between F_1 and F_2, based on the expected discounted Hamming distance, with discount factor γ, and with the expectation computed over the probabilistic model P. We denote this distance $D_h(F_1, F_2, P, \gamma)$.

Definition 5: The expected discounted Hamming distance *between two filters F_1 and F_2 with respect to the probability model P is defined as:*

$$D_h(F_1, F_2, P, \gamma) = \sum_{s \in L(F_1) \cap L(F_2)} P(s)\, h^\gamma(F_1(s), F_2(s)). \qquad (2)$$

Fig. 2 shows a simple example that illustrates the value of minimization using expected, rather than worst case, error.

D. Probabilistic improper filter reduction

The formal problem is thus:

Problem: Probabilistic-Improper-FM

Input: A filter F, a probabilistic model P, a discounting factor γ, and an integer k.

Output: A filter F' with at most k states, such that $L(F) \subseteq L(F')$ and $D_h(F, F', P, \gamma)$ is minimal.

We now show that solutions to this problem that are both efficient and optimal are unlikely.

Theorem 1: PROBABILISTIC-IMPROPER-FM *is NP-hard.*

Proof: Reduction from the original filter minimization problem, FM [11], which is identical to PROBABILISTIC-FILTER-FM, with two important distinctions: First, FM seeks a reduced filter whose behavior is identical to the input filter, rather than one that merely minimizes the distance. Second, FM is a decision problem, asking whether a reduced filter exists at the given size k, a technical necessity for it to be in NP-complete.

Suppose a polynomial time algorithm for PROBABILISTIC-IMPROPER-FM exists. Then, given an instance (F_1, k) of FM, one can form a probabilistic model P with the same states and edges, and with arbitrary non-zero summing-to-one probabilities assigned to the edges. Using this P, along with the algorithm whose existence we have supposed we can efficiently find the filter F_2 with size k that minimizes $D(F_1, F_2, P, \gamma)$. Observe that $D_h(F_1, F_2, P, \gamma) = 0$ if and only if, for any observation sequence s for which $P(s) > 0$, we have identical outputs $F_1(s) = F_2(s)$. Therefore, F_1 can be reduced to size k if and only if $D_h(F_1, F_2, P, \gamma) = 0$.

Since FM is NP-complete, and we have a polynomial-time reduction from FM to PROBABILISTIC-IMPROPER-FM, we conclude that PROBABILISTIC-IMPROPER-FM is NP-hard. □

Consequently, we restrict our attention to heuristic algorithms that attempt, but cannot guarantee, to minimize the distance between the reduced filter and the original.

IV. ALGORITHM DESCRIPTION

This section outlines a heuristic algorithm for probabilistic improper filter reduction. The presentation has three phases. First, in Section IV-A, we address the general problem of computing the distance between two filters F_1 and F_2 with respect to a probabilistic model P and discount factor γ, i.e., $D_h(F_1, F_2, P, \gamma)$, by providing a dynamic programming recurrence whose values converge to this distance. Then, in Section IV-B, we show this recurrence can be used to efficiently bound, both from above and from below, the distance between a pair of filters in only a few iterations of dynamic programming. Lastly, in Section IV-C, we show how the resulting bounds can be used within the existing greedy sequential algorithm for improper filter reduction to determine which of two candidates is the closest to the original filter by operating lazily so that just enough iterations to accurately identify the best candidate are used.

A. Discounted Hamming distance metric

In this section, we derive a recurrence for the distance $D_h(F_1, F_2, P, \gamma)$ between two filters F_1 and F_2 with respect to a probability model P. To do so, we allow both the observation string length and the starting states to vary.

Definition 6: Let $d_h(q_1, q_2, v, k)$, in which q_1 is a state in F_1, q_2 is a state in F_2, and v is a state in P, denote the expected discounted Hamming distance, over all observation sequences s of length k in $L(F_1, q_1) \cap L(F_2, q_2)$ with respect to the probability model P starting at state v. If there are no such observation strings, $d_h(q_1, q_2, v, k)$ is defined to be 0.

Now we describe how to compute d_h via a recurrence on the observation string length k. When $k = 0$, the observation string is empty, leading to a single output. In this case we need only check whether the two states have same color:

$$d_h(q_1, q_2, v, 0) = [c_1(q_1) \neq c_2(q_2)]. \qquad (3)$$

In the general case with $k > 0$, we consider the set of common observations for q_1, q_2, and v—that is, the observations for which all three states have out-edges—denoted $Y(q_1, q_2, v)$. For each observation $y \in Y(q_1, q_2, v)$, there exist edges $q_1 \xrightarrow{y} q_1'$ in F_1, $q_2 \xrightarrow{y} q_2'$ in F_2, and $v \xrightarrow{(y, p_y)} v'$ in P, all labeled with the same observation y. This allows us to express the expected discounted Hamming distance recursively:

$$d_h(q_1, q_2, v, k) = [c_1(q_1) = c(q_2)] + \sum_{y \in Y(q_1, q_2, v)} p_y \gamma \, d_h(q_1', q_2', v', k-1). \qquad (4)$$

The idea is to directly compute the contribution to the overall distance from the first stage, and combine this value with the remaining distance, appropriately weighted according to P and discounted according to γ. Given d_h values for each state triple at a particular observation string length $k-1$, we can use (4) to compute d_h at for observation strings of length k. In the particular the case when $Y(q_1, q_2, v) = \varnothing$, we follow the standard convention of treating the vacuous sum as 0.

B. Using d to bound D

We can now establish a connection between d_h and D.

Lemma 2: Consider two filters F_1 and F_2 along with a probabilistic model P. Let $q_0^{(1)}$, $q_0^{(2)}$, and v_0 denote the respective initial states of these graphs. Then for any positive integer k,

$$d_h(q_0^{(1)}, q_0^{(2)}, v_0, k) \leq D_h(F_1, F_2, P, \gamma), \qquad (5)$$

and

$$D_h(F_1, F_2, P, \gamma) \leq d_h(q_0^{(1)}, q_0^{(2)}, v_0, k) + \frac{\gamma^{k+1}}{1-\gamma}. \qquad (6)$$

Proof: For (5), the lower bound, observe that in the best case, F_1 and F_2 may produce the same colors at every stage beyond stage k, for every string in $L(F_1) \cap L(F_2)$ accumulating no additional error. Likewise, for the upper bound (6), we appeal to a worst case in which F_1 and F_2 disagree at *every* position beyond stage k. Thus—recalling that $0 < \gamma < 1$ implies that the series converges—the additional error, as the observation strings grow longer, is bounded by

$$\begin{aligned} \sum_{i=k+1}^{\infty} \gamma^i &= \sum_{i=0}^{\infty} \gamma^i - \sum_{i=0}^{k} \gamma^i \\ &= \frac{1}{1-\gamma} - \frac{1-\gamma^{k+1}}{1-\gamma} \\ &= \frac{\gamma^{k+1}}{1-\gamma}. \end{aligned}$$

□

Corollary 1: For any filters F_1 and F_2 and probabilistic model P, $D_h(F_1, F_2, P, \gamma) = \lim_{k\to\infty} d_h(q_0^{(1)}, q_0^{(2)}, v_0, k)$.

Proof: Follows from Lemma 2, the definition of the limit, and the convergence of $\gamma^{k+1}/(1-\gamma)$ to 0 as $k \to \infty$. □

C. Lazy greedy sequential reduction

Prior research by some of the present authors [13] introduced the problem of improper filter reduction using worst-case, rather than probabilistic, reasoning. That work introduced an approach called *local greedy sequential (LGS) reduction* that works by iteratively reducing the size of the original filter, one state at a time, until the desired size is reached. This algorithm choose the best candidate to merge in each iteration for reducing one state. In this section, we adapt that general approach for the current probabilistic setting. The bounds from Lemma 2 enable dynamic programming that does not need to compute the distance between each candidate solution to any great precision. This form of 'laziness' was not feasible in the original worst-case, undiscounted version of the algorithm.

Algorithm 1 shows the lazy greedy sequential algorithm for Probabilistic-Improper-FM. The idea is to form smaller filters by way of a MERGE operation that combines a pair of states, much like a vertex contraction operation but with an additional step performed afterwards. This post-processing step ensures that the language of the merged filter contains the language of the original and is achieved via the addition of extra edges. Fig. 3 provides an example: a forward search discovers any observations that are missing and introduces an appropriate edge for each one. Utilizing this merge operation, the algorithm considers all state pairs as candidates for merger. It selects the merged filter whose distance from the original input is the smallest, preferring merges that do not leave any states unreachable, if any such merges exist.

The heart of Algorithm 1 is its call to the subroutine BESTCANDIDATE, whose job is to determine whether the newly-created candidate F' is closer to the original filter F_{orig} than the current best known candidate $F^\star$ (see Algorithm 2). BESTCANDIDATE maintains, for each of the two candidate filters, the largest k for which d_h has been computed for that filter. This k, along with the d_h table giving distances from F_{orig} and the upper and lower bounds from Lemma 2 denoted $\overline{D}$ and $\underline{D}$ respectively, are stored as attributes of each filter.

If there is a clear separation between the two candidate filters, that is, if the lower bound for one of the filters exceeds the upper bound for the other, BESTCANDIDATE can terminate. If not, the algorithm then performs an additional iteration of dynamic programming on one of the filters—specifically, the one whose k value is the smallest, because

Algorithm 1: LAZYGREEDYIMPROPER(F, P, γ, k)

```
$f \leftarrow$ False
$F_{\text{orig}} \leftarrow F$
while $|V(F)| > k$ do
    $F^\star \leftarrow \varnothing$
    for $(q_1, q_2) \in V(F) \times V(F)$ do
        if $q_1 = q_2$ then
            continue
        $F' \leftarrow$ MERGE$(F, q_1, q_2)$
        if (not $f$) and $F'$ has unreachable states then
            continue
        if $F^\star = \varnothing$ then
            $F^\star \leftarrow F'$
        else
            $F^\star \leftarrow$ BESTCANDIDATE$(F_{\text{orig}}, F^\star, F')$
    if $F^\star = \varnothing$ then
        $f \leftarrow$ True
    else
        $f \leftarrow$ False
        $F \leftarrow F^\star$
return $F$
```

Algorithm 2: BESTCANDIDATE$(F_{\text{orig}}, F_1, F_2, P, \gamma)$

```
while True do
    if $F_2.\underline{D} > F_1.\overline{D}$ then
        return $F_1$
    if $F_1.\underline{D} > F_2.\overline{D}$ then
        return $F_2$
    if $F_1.\underline{D} \approx F_1.\overline{D} \approx F_2.\underline{D} \approx F_2.\overline{D}$ then
        return $F_1$
    if $F_1.k < F_2.k$ then
        IMPROVEBOUNDS$(F_{\text{orig}}, F_1, P, \gamma)$
    else
        IMPROVEBOUNDS$(F_{\text{orig}}, F_2, P, \gamma)$
```

Algorithm 3: IMPROVEBOUNDS$(F_{\text{orig}}, F, P, \gamma)$

```
$F.k \leftarrow F.k + 1$
$Q \leftarrow (q_0(F), q_0(F_{\text{orig}}), v_0(P))$
while $Q$ in not empty do
    $(q_1, q_2, v) \leftarrow Q.get()$
    Compute $d_h(q_1, q_2, v, F.k)$ using Eq. 3 or Eq. 4
    for each out-going edge $q_1 \xrightarrow{y} q_1'$ in $F_{\text{orig}}$ do
        if $q_2 \in F$ has an out-going edge $q_2 \xrightarrow{y} q_2'$ then
            if $v \in P$ has an out-going edge $v \xrightarrow{y,p} v'$ and
              $(q_1', q_2', v')$ has not visited before in $Q$ then
                $Q \leftarrow (q_1', q_2', v')$
$F.\underline{D} \leftarrow d_h(q_0(F), q_0(F_{\text{orig}}), v_0(P), F.k)$
$F.\overline{D} \leftarrow d_h(q_0(F), q_0(F_{\text{orig}}), v_0(P), F.k) + \gamma^{k+1}/(1-\gamma)$
```

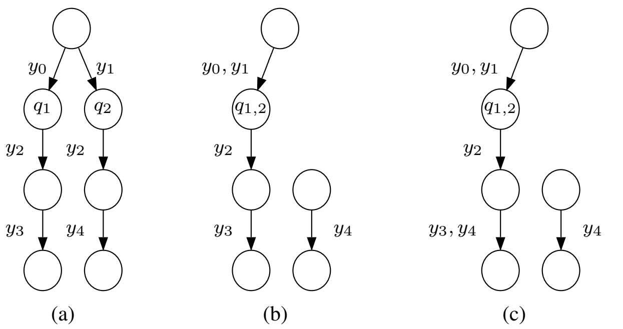


Fig. 3: An example of MERGE(F, q_1, q_2), and the post-processing it performs to ensure that $L(F) \subseteq L(F')$. (a) A filter F, before merging q_1 and q_2. (b) An intermediate filter generated by merging q_1 with q_2. Here, F can process the observation sequence $y_1y_2y_4$, but the intermediate filter fails on this input. (c) Post-processing inserts an edge, found by searching forward, to obtain F'.

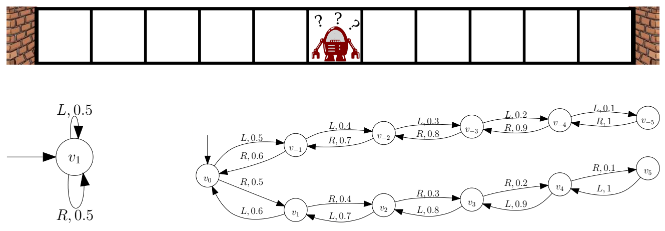


Fig. 4: [top] A robot determining whether it is in the center of its environment. [bottom left] A probabilistic model with a uniform distribution observation strings, which we call UNIFORM; [bottom right] A probabilistic model biased toward movement back to the center, which we call CENTER.

earlier iterations have greater impact on the bounds—and repeats. This process continues until one of the two candidates is clearly determined to be closer to F_{orig}, or until the bounds are sufficiently close that the difference between the two candidates is now known to be negligible.

This process—generation of candidates by merging pairs of states and selection of the best candidate using lazy dynamic programming—continues until the filter is reduced to the desired size.

V. Experiments

We implemented Algorithms 1–3 in Python. This section reports the performance an example problem: We executed Algorithm 1 on a filter constructed for the problem shown in the top portion of Fig. 4. The input filter represents an agent moving in an environment with two basic actions *move left* and *move right*. At the end states 5 and -5, the agent encounters the walls of environment and further actions in those directions have no effect. The probabilistic models in Fig. 4 represent two different distributions on the observations. The first model shows a uniform distribution (called UNIFORM hereafter); the second represents a distribution in which the agent moves toward the center with probability proportional to the distance from the center (called CENTER).

We reduced this basic filter from size 11 down to each target size $k \in \{10, \dots, 2\}$. For each size, we ran Algorithm 1 using each of the probabilistic models in Fig. 4, using $\gamma = 0.95$ throughout. For comparison, we attempted the same reductions using the original LGS reduction algorithm [13], which uses worst-case, rather than probabilistic reasoning. For each of these runs, across both algorithms, we computed the expected discounted Hamming distance of the output filter from the original input filter. The results are shown in Table I and Table II.

A. A comparison of output quality

Table I reports expected discounted Hamming distance of the output filters based on the CENTER and UNIFORM probabilistic models. With the former model, the differences between the two algorithms are less pronounced than the latter. This is indicative of the fact that Algorithm 1 is able to exploit the structure in the CENTER model; this structure is absent in the UNIFORM model. As a specific example, when $k = 3$ with the CENTER model, Algorithm 1 has

	Quality		
Probabilistic Model	k	Algorithm 1	LGS [13]
CENTER	2	7.805	7.805
CENTER	3	0.575	6.773
CENTER	4	0.331	3.501
CENTER	5	0.331	0.297
CENTER	6	0.212	0.278
CENTER	7	0.095	1.717
CENTER	8	0.034	1.705
CENTER	9	0.018	0.146
CENTER	10	0.006	0.142
UNIFORM	2	2.964	6.688
UNIFORM	3	2.082	4.010
UNIFORM	4	1.841	2.121
UNIFORM	5	1.670	1.259
UNIFORM	6	1.198	1.038
UNIFORM	7	0.685	0.819
UNIFORM	8	0.495	0.565
UNIFORM	9	0.276	0.328
UNIFORM	10	0.148	0.177

TABLE I: The expected discounted Hamming distance for two algorithms under the CENTER and UNIFORM probabilistic models.

an error of 0.575, while LGS achieves a value of 6.773, a factor of more than 10. For the same target size with the UNIFORM model, Algorithm 1 has an error of 2.082, while LGS produces 4.010, a more modest factor of 2.

In both CENTER and UNIFORM models, there are some values of k where changing the parameter causes no change in quality. This is explained by looking at the filters actually produced by the algorithms: for example, see Fig. 5, where as k increases from $k = 3$ to $k = 4$, we see that the filter outputs (i.e., colors) are indistinguishable.

The results show generally better performance for the new algorithm, especially when the amount of reduction is small.

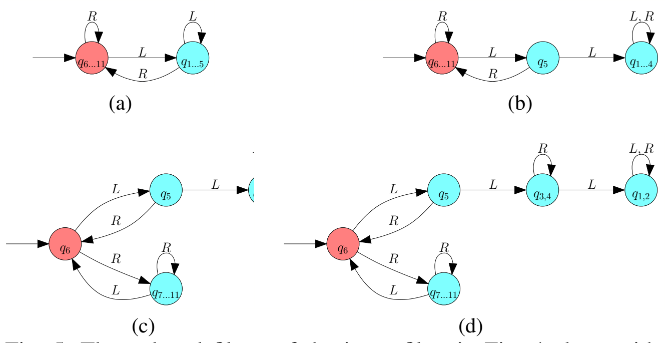


Fig. 5: The reduced filters of the input filter in Fig. 4 along with CENTER model to size (a) $k = 2$ with 7.805 error, (b) $k = 3$ with 0.575 error, (c) $k = 4$ with 0.331 error and (d) $k = 5$ with 0.331 error. The change from (c) to (d) does not affect the error as they always output the same colors.

B. Comparative efficiency

Table II shows the run times of these algorithms on a single core of a 3.1GHz processor. The data show that Algorithm 1 is slower than LGS [13]. Some of this difference can be attributed to the extra computation required to account for the various states in the probabilistic model, manifest in the need to consider state triples, rather than merely state pairs, in Algorithm 3. The difference is more pronounced for the CENTER model, owing to the greater number of states for that model.

COMPUTATION REQUIRED			
Probabilistic Model	k	Algorithm 1	LGS [13]
CENTER	2	2.293 s	0.176 s
CENTER	3	2.294 s	0.152 s
CENTER	4	2.265 s	0.152 s
CENTER	5	2.157 s	0.145 s
CENTER	6	1.929 s	0.139 s
CENTER	7	1.649 s	0.122 s
CENTER	8	1.296 s	0.101 s
CENTER	9	0.898 s	0.081 s
CENTER	10	0.477 s	0.047 s
UNIFORM	2	2.048 s	0.156 s
UNIFORM	3	2.059 s	0.152 s
UNIFORM	4	1.940 s	0.151 s
UNIFORM	5	1.854 s	0.146 s
UNIFORM	6	1.590 s	0.137 s
UNIFORM	7	1.420 s	0.121 s
UNIFORM	8	1.136 s	0.102 s
UNIFORM	9	0.797 s	0.081 s
UNIFORM	10	0.359 s	0.047 s

TABLE II: The run time of two algorithms with the CENTER and UNIFORM probabilistic models.

VI. CONCLUSION

This paper is concerned with reducing the information needed for a robot to perform its tasks effectively. Toward this end, we have generalized a previous notion of distance between two combinatorial filters by including a probability distribution over the inputs that provides a way of expressing structure of the world as it manifests itself through observations. This allows one to construct filters that are smaller because, though they make some mistakes (i.e., are improper reductions), those mistakes have a low likelihood of occurrence. As is widely recognized, average-case analysis of algorithms is typically more meaningful than worst-case bounds when a distribution over the inputs is available. The present work provides an analogous distinction for particular filter reduction instances, contributing a new algorithm for reducing the size of a discrete filter via minimization of an expected distance. Prior to the formulation in this paper, no such expectation existed with well-defined form.

A separate advantage of a probability distribution over filter inputs is that algorithms may use computational resources wisely, only investing time on important decisions and avoiding time wasted on determining distinctions that are mostly irrelevant (i.e., occurring only with low probability). This is part of the reason we have applied the sobriquet 'lazy' to our algorithm, as it need not compute a precise distance measure if it can make the determination sufficiently well given the probabilities involved. But, as the recorded running-times from our experiments show, further optimization is needed to fully realize the potential to spend less computational time. Leveraging a recent result in [4], we already know that there are a class of filters and probabilistic models for which the computation time is never any worse than the non-probabilistic case, namely ones for which one of the filters and probabilistic model are homomorphic. This, and other performance improvements, await treatment as future work.

Acknowledgements: This material is based upon work supported by the National Science Foundation under Grants IIS-1527436, IIS-1526862, IIS-0953503, IIS-1453652.

REFERENCES

[1] L. Bobadilla, O. Sanchez, J. Czarnowski, and S. M. LaValle, "Minimalist multiple target tracking using directional sensor beams," in *Proceedings IEEE International Conference on Intelligent Robots and Systems*, 2011.

[2] M. Erdmann, "Understanding action and sensing by designing action-based sensors," *International Journal of Robotics Research*, vol. 14, no. 5, pp. 483–509, 1995.

[3] M. Erdmann and M. T. Mason, "An exploration of sensorless manipulation," *IEEE Transactions on Robotics and Automation*, vol. 4, no. 4, pp. 369–379, Aug. 1988.

[4] S. Ghasemlou, F. Z. Saberifar, J. M. O'Kane, and D. Shell, "Beyond the planning potpourri: reasoning about label transformations on procrustean graphs," in *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2016.

[5] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 1st ed. Addison-Wesley, 1979.

[6] S. Kristek and D. Shell, "Orienting deformable polygonal parts without sensors," in *Proc. International Conference on Intelligent Robots and Systems*, 2012.

[7] G. Lagunaa, R. Murrieta-Cid, H. M. Becerra, R. Lopez-Padilla, and S. M. LaValle, "Exploration of an unknown environment with a differential drive disc robot," in *IEEE International Conference on Robotics and Automation*, 2014.

[8] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[9] ——, "Sensing and filtering: A fresh perspective based on preimages and information spaces," *Foundations and Trends in Robotics*, vol. 1, no. 4, pp. 253–372, 2012.

[10] R. Lopez-Padilla, R. Murrieta-Cid, and S. M. LaValle, "Optimal gap navigation for a disc robot," in *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2012.

[11] J. M. O'Kane and D. Shell, "Concise planning and filtering: hardness and algorithms," *IEEE Transactions on Automation Science and Engineering*, 2017, to appear.

[12] F. Z. Saberifar, A. Mohades, M. Razzazi, and J. M. O'Kane, "Combinatorial Filter Reduction: Special Cases, Approximation, and Fixed-Parameter Tractability," *Computer and System Sciences*, 2017.

[13] ——, "Improper Filter Reduction," *arXiv*, 2017.

[14] Y. Song and J. M. O'Kane, "Comparison of constrained geometric approximation strategies for planar information states," in *Proc. International Conference on Robotics and Automation*, 2012.

[15] B. Tovar, "Minimalist models and methods for visibility-based tasks," Ph.D. dissertation, University of Illinois at Urbana Champaign, 2009.

[16] B. Tovar, F. Cohen, L. Bobadilla, J. Czarnowski, and S. M. LaValle, "Combinatorial filters: Sensor beams, obstacles, and possible paths," *ACM Transactions on Sensor Networks*, vol. 10, no. 3, p. 47, 2014.

[17] B. Tovar, F. Cohen, and S. M. LaValle, "Sensor beams, obstacles, and possible paths," in *Algorithmic Foundations of Robotics, VIII*. Berlin: Springer-Verlag, 2009.

[18] B. Tovar, R. Murrieta-Cid, and S. M. LaValle, "Distance-optimal navigation in an unknown environment without sensing distances," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 506–518, Jun. 2007.

[19] J. Yu and S. M. LaValle, "Cyber detectives: Determining when robots or people misbehave," in *Algorithmic Foundations of Robotics IX, Springer Tracts in Advanced Robotics (STAR)*. Springer Berlin/Heidelberg, 2011, vol. 68, pp. 391–407.

[20] ——, "Story validation and approximate path inference with a sparse network of heterogeneous sensors," in *IEEE International Conference on Robotics and Automation*, 2011.

[21] ——, "Shadow information spaces: Combinatorial filters for tracking targets," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 440–456, Apr. 2012.