Equivalence notions for state space minimization of combinatorial filters

Hazhar Rahmani and Jason M. O'Kane

***Abstract*— Combinatorial filters are formal structures for filtering and reasoning over discrete sensor data. This paper presents a series of results addressing the question whether the *filter minimization problem* (FM), an NP-hard problem of state space reduction on such filters, and a variant of it, the *filter partitioning minimization problem* (FPM), which requires the reduced filter to partition the state space of the original filter, can be solved via quotient operations under equivalence relations of the state space. We first consider the well-known notion of *bisimulation* and show that, although bisimulation always yields feasible solutions to FM and FPM, it does not necessarily induce optimal solutions. We also establish a connection between filter reduction and the notion of *simulation*; specifically, we show that FM is equivalent to the problem of inducing a minimal filter that simulates a given filter. We then introduce a variant of bisimulation, which we call *compatibility*, and prove that FPM can always be solved by computing the quotient of the input filter under a *compatibility equivalence* relation having a minimum number of equivalence classes. On the other hand, computing optimal solutions to FM requires to look for relations beyond equivalence relations, and in fact, FM can be solved by computing the quotient of the original filter under a closed covering of the state space with minimum number of compatibility classes. Subsequently, we introduce two special relations, *the union of all compatibility relations* and *the mergeability relation*, that are both computable in polynomial time. By analyzing where these two relations become an equivalence relation, we identify several classes of filters for which FM and FPM are solvable in polynomial time.**

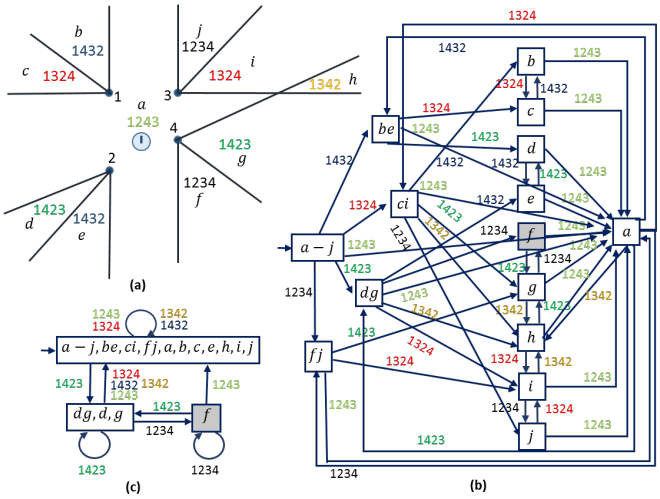

Fig. 1. **a)** An environment with four landmarks 1-4, in which a mobile robot (not pictured) moves along a continuous path. The robot can sense at any time the cyclic order of landmarks as observed from its current position, but it does not know the positions of the landmarks and it does not have a compass nor odometers. The environment is (virtually) divided into 10 regions a-j, where at all points within a region, the same cyclic order of landmarks is perceptible. The task of the robot is to provably tell at any time whether it is in region f or not. **b)** A naïve filter the robot can use to accomplish its task. Notice that each state of the filter except state $a-j$ has also a loop labelled by the same cyclic order perceptible by the regions of that state, but we have avoided drawing those loops for simplicity. **c)** The smallest filter equivalent to the naïve filter.

I. Introduction

The ability of robots to perceive and maintain salient information about their environments —and their own place within those environments— is rightly considered to be an essential ingredient for many forms of autonomy. As a result, a central thread in modern robotics research is an effort to design and understand filtering and estimation methods.

A number of robotics researchers in recent years have considered these kinds of problems from the perspective of *combinatorial filters*, which model those processes as discrete transition systems. Combinatorial filters, first proposed by LaValle [26], [27], are a general class of models for reasoning about systems that process discrete (rather than continuous) sensor data. Variations on the combinatorial filtering approach have, under various names, been used for a wide spectrum of tasks including localization [2], navigation [28], [52], [55], exploration [25], manipulation [24], mapping [50], target tracking [7], [13], [63], and story validation [62]. The essence of the approach can be understood as describing filtering processes as directed graphs, in which the vertices represent the distinct states of the information maintained by the filter, the directed edges are labeled with sensor readings that induce transitions between states, and the vertex labels indicate the output of the filtering process at each state.

Central to much of the research on combinatorial filters is the notion of *minimality*. Specifically, questions are asked about the minimal state information required to be maintained in the filter, in order to accurately express the desired behavior. This concern naturally leads to questions about *equivalences* between states in the filter: If we can construct a filter for a given task in which no pair of distinct states is equivalent, in the (thus far informal) sense that the distinction between them is irrelevant to the filter's outward behavior, then the states utilized by that filter may illuminate the information requirements of the task. Thus, for a given combinatorial filter, we are interested in the *filter minimization problem (FM)* of finding the smallest equivalent filter. This problem was addressed by O'Kane and Shell [35], [36] who proved that it is NP-hard. They proposed a heuristic algorithm to solve FM, which forms the reduced filter by merging pairs of states who are identified to be mergeable

The authors are with the Department of Computer Science and Engineering, University of South Carolina, Columbia, South Carolina, USA. hrahmani@email.sc.edu, jokane@cse.sc.edu
This material is based upon work supported by the National Science Foundation under Grant Nos. 1526862, 1659514, 1849291.

using iteratively forming and coloring conflict graphs. This algorithm can also be used for a variant of FM, called the *filter partitioning minimization problem (FPM)*, which requires the reduced filter to partition the state space of the original filter.

This paper addresses FM and FPM by considering several distinct state-equivalence relations in the context of that problem. Specifically, we show that the well-known state-equivalence concept of *bisimulation* —which is used widely for the minimization of other discrete transition structures [48]— does not, as intuition might suggest, correctly capture the notion of equivalence between states necessary to minimize a filter optimally. Nonetheless, using bisimulation we identify, in Section X, classes of filters for which FPM or FM is solvable in polynomial time. The notion of bisimulation also provided inspiration for the correct notions of *compatibility* and *mergeability* for FM and FPM, which we introduce in this paper.

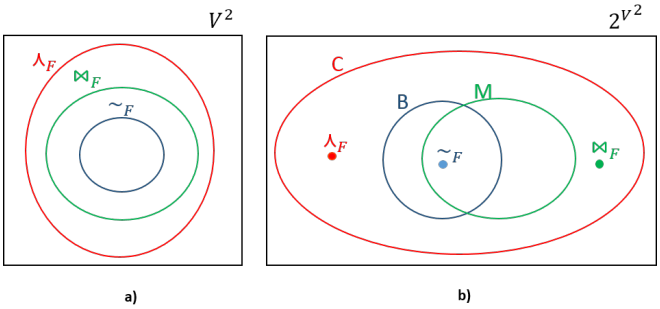

Fig. 2. **a)** A Venn diagram showing the connection between the bisimilarity relation $\sim_F$, the union of all compatibility relations $\mathfrak{f}_F$, and the mergeability relation $\bowtie_F$ for a typical filter F with state space V. Note that it is possible that two or all these three relations coincide for some filters. **(b)** A Venn diagram, over the space of state-state relations, showing the relationships between the set of all bisimulation relations (B), the set of all compatibility relations (C), and the set of all compatibility equivalence relations (M) for a typical filter F with state space V.

A. Summary of contributions

The central question we address in this article is whether there exists an equivalence relation on the state space of the original filter, which when used for constructing a quotient filter does lead to an optimally reduced filter, and if yes, what is that equivalence relation.

After reviewing related work in Section II and presenting in Section III, the formal definitions of the two problems we study, the FM problem and the FPM problem, this paper presents several new contributions.

1) In Section IV, we characterize exact solutions to FM and FPM. We show that FPM is always solved by finding over the state space of the filter, an equivalence relation, which is always a compatibility relation, that has the minimum number of equivalence classes, while solving FM requires to search for relations beyond equivalence relations and in fact solving FM is always solved by finding a *tolerance* relation that specifies over the state space of the filter, a covering with minimum number of compatibility classes. In Section V, we show that for some filters, the difference between the sizes of the optimal solutions to FM and FPM can be in the order of the size of filter.
2) In Section VI, we show that, though the bisimilarity relation always yields feasible solutions to both FM and FPM, it fails to produce optimal solutions for some filters. We then complement this negative result by describing what bisimilarity reduction actually does achieve: namely that it induces the filter with smallest number of states, among all filters who behave the same as the original filter, and whose languages are identical to the language of the original filter, a problem we call strong filter minimization (SFM). This result appears in Section VII. We show, in Section VIII, that FM is equivalent to the problem of finding a minimal filter that *simulates* the original filter, a problem we refer to as minimal simulation filter (MSF). We also introduce a variant of MSF—the minimal partitioning simulation filter (MPSF), which requires the reduced filter to partition the state space of the original filter—and show that FPM is equivalent to this variant of MSF. These results together characterize the relationship between filter reduction and the well-known concepts of (bi-)simulation.
3) Finally, we show how these concepts can be used to identify classes of filters that can be minimized in polynomial time, in spite of the hardness [36] of FM and FPM in general. Specifically, we show that if the *union of all compatibility relations*, the counterpart of the bisimilarity relation for the notion of compatibility, is an equivalence relation, then that relation induces an optimal solution for both FM and FPM. We also introduce the notion of a *mergeability relation*, which is a particular type of compatibility relation. We show that, if the mergeability relation of a filter is an equivalence relation, then that relation induces an optimal solution to FM. Because these two special relations can be computed in polynomial time, this provides an avenue to reduce certain classes of filters in polynomial time. Section X describes several such classes.

Concluding remarks appear in Section XI.

A preliminary version of this work appeared at ICRA 2018 [40]. In addition to revisions throughout, the results in Sections V, VIII, and X are new in this version.

B. A minimal motivating example

Before previewing our technical results, let us examine a simple example that illustrates the idea. Figure 1 shows the setting, inspired by the work of Tovar *et al.* [54], where the aim is to construct a combinatorial map of the environment and perform various tasks such as patrolling and navigation using robots with extremely simple sensors.

Suppose a mobile robot moves in an environment in which there are four landmarks 1-4. The robot does not know its location and it does not have a compass nor any odometer, but it does have a sensor by which it can sense the cyclic order of landmarks as observed from its current

position. That is, the sensor reports the order in which the landmarks would be observed in a counterclockwise scan starting (without loss of generality) at the landmark with the smallest number. To illustrate, suppose the robot is located as shown in Figure 1a, at a location in region a. From that position, the sensor reads 1243 as the cyclic order of landmarks.

Figure 1 shows a (virtual) decomposition of the environment into ten regions a-j, where each region is a set of positions from which the robot sees the same cyclic order of the landmarks. Notice that such a decomposition is obtained by a set of half lines, in which each half line is created for two landmarks whose order is swapped in the cyclic order observed by the robot if the robot crosses that half line.

The task of the robot in this example is to determine at any time, based on the information it receives through sensing the cyclic order of landmarks, whether it is located in region f or not.

A naïve combinatorial filter for this task appears in Figure 1b. Each of the fifteen states in this filter corresponds to a set of possible regions that might contain the agent. Edges indicate changes to that set that result from changes of the robot's perception of the landmarks' cyclic order. Colors on the states indicate the filter's output, with the darker node corresponding to certainty that the agent is in region f and the lighter nodes indicating otherwise.

For the original filter in Figure 1b, FM has only one optimal solution, with 3 states. This optimally-reduced filter, which is shown in Figure 1c, is also an optimal solution to FPM; note that the state space of the reduced filter partitions the state space of the original filter in the sense that each state in the original filter is mimicked by only a single state (rather than several states) in the reduced filter. In regard with FM, it is possible that several states in the reduced filter together play the role of a state in the original filter. This optimal solution in Figure 1c is not only preferred to the naïve filter for the robot to keep but it also reveals the fact that perhaps surprisingly the sensor reading '1423' corresponding to region d, which is not adjacent to the target region f, is a crucial one for the robot to achieve its task. In contrast, applying the technique of bisimulation minimization does not reduce the state space of the original filter at all. In Theorem 3, we describe a class of combinatorial filters on which bisimulation minimization does not reduce the state space of the original filter at all, whereas the optimally reduced filter has only two states.

C. An example of a combinatorial filter as a plan

In the specific example we consider here, a combinatorial filter is used to navigate in an environment, using an approximated generalized Voroni graph (GVG) of the environment to reach a goal location, starting from full location uncertainty. For an example, see in Figure 3 a map of a typical university building [5], [6], which we have augmented with its GVG in red. We have also shown a simpler environment with its GVG in Figure 3b for illustration. Figure 3c shows a naïvely-constructed combinatorial filter, which the robot can

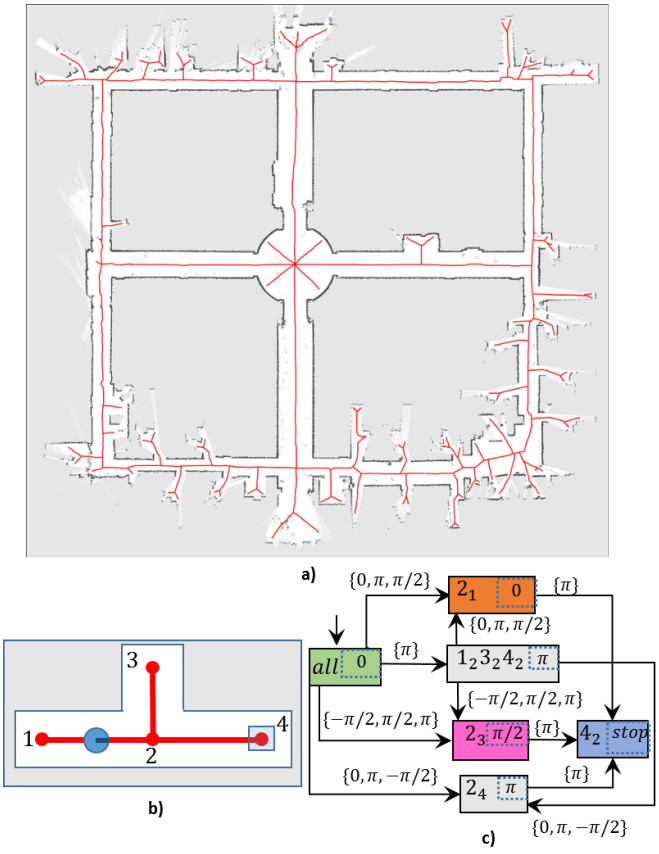

Fig. 3. **a)** A map of the third floor of ACES building of University of Texas at Austin [5], [6], [34]. The map is overlaid with the generalized Voronoi graph (GVG) of the environment in red. **b)** A simple environment and its GVG for illustration purposes. **c)** A naïvely-constructed combinatorial filter, which the robot can use to navigate through the environment in part b) to reach point 3, starting from full location uncertainty.

use to navigate through that environment to reach point 4, starting from full location uncertainty. Each state of this filter represents a set of possible locations in which the robot can be. Each location is written in the form A_B to mean that the robot has arrived at junction A from junction B. Each state is labeled with an angle (0, =2, , etc), which is, in fact, the output or the color of the state, and tells the robot which corridor to follow. For example, angle 0 tells the robot to go straight, while angle tells the robot to turn around and follow the corridor from which it has entered the junction. Each transition of the filter shows how the state changes in response to those movements. Each transition is labeled with an observation which consists of a list of directions the robot can observe to leave the next junction. For example, the label of the transition from state 'all' to state 2_1 is $f0;\ ;\ =2g$, which means that when the robot is entering junction 2 from junction 1, the set of all possible directions it observes to leave junction 2 are $f0;\ ;\ =2g$. The robot executes the plan represented by a filter by repeatedly doing the movement associated with the current state, observing the set of all possible directions in the junction using its sensor, and transitioning in the graph from the current state using a transition labeled by the perceived observation.

Given a GVG and a goal location, there is a simple

algorithm by which one can make a naïve combinatorial filter the robot can use to navigate, starting from total location uncertainty, to reach that goal location. Unfortunately, those naïvely constructed combinatorial filters can be big for real-world applications, but fortunately filter reduction assists reduce resource consumption.

D. Motivation for minimization

We now consider in Figure 4, a simple example, introduced by Tovar et. al. [53], that shows a family of filters for which the minimal filter is much smaller than the original filter. In this example two agents are located in a donut-shaped environment divided into n regions by n sensor beams. The task of the system is to determine at any time whether the two agents are in the same region or not. When an agent crosses a beam, the system knows which beam it was, but it does not know which agent it was, nor does it know if the crossing was clockwise or otherwise. We also assume that at no time, the two agent simultaneously cross a single or two separate beams. Let $R = f1; 2; \quad ; ng$ be the set of all regions and let $(r; r^0)$ be a tuple showing that the first agent is in region r and the second agent is in region r^0. Accordingly, the set of all such tuples, i.e., the set of all possible region assignments, is $A = R \quad R$ and the set of all subsets of possible region assignments is $I = \mathrm{pow}(R \quad R)$. Each element of I is a configuration the environment could be in, and accordingly, each element of I represents a state of the naïve filter used for accomplishing the task of the system. Subsequently, for each n, the naïve filter for a donut-shaped environment with n regions has $2^{n^2} \quad 1$ states in the worst case (including some states that may be unreachable). A minimal filter for accomplishing the same task of the naïve filter has many fewer states. As an example, for an environment of 3 regions, the naïve filter has 511 states while the minimal filter, as shown in the right side of Figure 4, has only 4 states.

Note that although there is a process by which we can make a naïve filter for any instance of this problem with any n, it is not known how we can make the minimal filter for that instance. Subsequently, we need to make the naïve filter, but because that filter is very big, we need to minimize it.

II. Related Work

Building on a foundation of prior work on minimalism in robotics [12], [18], [29], combinatorial filters were originally formulated by LaValle [26], [27]. The key idea is to make, from the data accessible to the robot, a smallest abstraction still adequate to solve a given task.

Interest in forming combinatorial filters that are minimal, in the sense of minimizing the number of states, is motivated not only by the reduction in resources needed to execute such filters, but also by the insight into the nature of the underlying problems that arises from identifying the information required to solve those problems. The problem of performing this reduction automatically was first studied by O'Kane and Shell [35], [36], who proved via a reduction from the graph 3-coloring problem that the filter minimization problem is

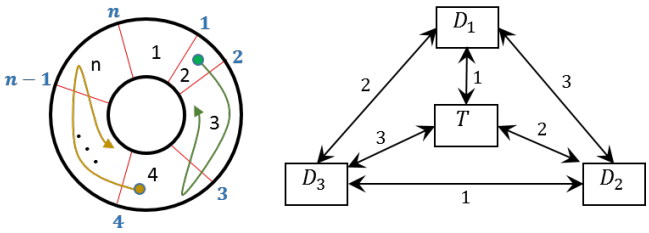

Fig. 4. **(left)** A region divided by n sensor beams, in which two agents are located. When an agent crosses a beam, the system knows which beam sensor it was, but it does not know which agent it was, nor does it know the direction of crossing. The agents do not simultaneously cross beams. The task of the system is to determine at each time weather the agents are in the same region or not. **(right)** The smallest filter the system can use for accomplishing its task for an environment of 3 sensor beams.

NP-hard. Saberifar *et al.* [47] showed that several special cases of filters, including tree and planar filters, remain hard to minimize, and that the filter minimization problem is NP-hard even to approximate. Recently, Zhang and Shell [64] proved that FM can be solved by finding a covering of the filter's state space and then presented an algorithm that uses a SAT formulation to compute an exact solution to FM. Our own prior work [41] proposed three different integer linear programming formulations of the filter partitioning minimization (FPM) problem, a variant of FM in which it is required the reduced filter to partition the state space of the original filter.

Bisimulation was discovered independently in at least three different fields: in modal logic, by van Benthem [56]; in process theory, by Milner [33] and Park [38]; and in set theory, by Forti and Honsell [15] (See the elaboration of Sangiorgi [48] upon the origins of bisimulation and simulation.) It is currently used across many fields, including automata and language theory [44], [45], coalgebra and coinduction [14], [46], and dynamical and control systems [20], [57]. Generally speaking, bisimulation can be used for at least two purposes: either to prove that two objects are behaviorally equivalent, or to minimize the size of a structure by forming the quotient under the coarsest bisimulation equivalence relation between elements of the original structure. This paper focuses on the latter application. Computing this coarsest bisimulation equivalence relation is generally performed using partition refinement algorithms [23], [37]. Details about bisimulation quotient algorithms appear in the survey by Cleaveland and Sokolsky [10].

The original notion of simulation was introduced by Milner [30]–[32], and later refined by Park [38]. This notion—which unlike bisimulation, is "uni-directed"—is used to prove that an object (for example, a state or a transition system) mimics another object; to show that a system is a correct implementation of a smaller, abstract system; and to make a smaller structure who with the original structure mutually simulate each other. In the latter case, the smaller structure is obtained by making the quotient of the original structure under the *simulation equivalence* relation—the equivalence kernel of the largest simulation relation over the state space of the original filter. Various algorithms for simulation-based

minimization and computing the largest simulation relation have been suggested by Cleaveland et al. [9], Henzinger et al. [21], Tan and Cleaveland [51], Ranzato and Tapparo [43], Bustan and Grumberg [8], Ranzato [42], and Gentilini et al. [16]. The algorithm of the latter was later corrected by van Glabbeek and Ploeger [58].

In the context of transition systems, apart from *simulation equivalence* and *bisimulation equivalence*, a variety of other kind of notions of equivalence on the state space of transition systems, including *trace equivalence*, *failure equivalence*, and *readiness equivalence*, among many others, have been introduced. For surveys, see [11], [19], [49], [59], [60].

The idea of using (bi)simulation-based equivalences for state space reduction has been used for a variety of other structures than transition systems, including tree automata [1], [1], [22], Markov chains [3], Markov decision processes [17], fuzzy transition systems [61], and timed systems [39].

III. Definitions

This section presents basic definitions used throughout the paper.

We are interested in filters that model the behavior of a robot in response to a discrete, finite sequence of observations. The following definitions are equivalent to the those introduced by O'Kane and Shell [36].

Definition 1: *A* filter *is a 6-tuple* $(V; Y; C; \ ; c; v_0)$ *in which:*

V *is a finite set of* states*,*
Y *is a set of possible* observations*, representing the input space of the filter,*
C *is a set of* outputs*, sometimes called* colors*, representing the outputs produced by the filter,*
$: V \quad Y \ ! \ V [f?g$ *is the* transition function *of filter,*
$c : V \ ! \ C$ *is a function assigning to each state* $v \ 2 \ V$ *a color, and*
$v_0 \ 2 \ V$ *is the initial state.*

The set of all possible observation sequences is denoted Y . Given an observation sequence $s = s_1 s_2 \quad s_n$, we use $s_{i::j}$ to denote the subsequence $s_i s_{i+1} \quad s_j$. For each i, the subsequence $s_{i::i}$ of s represents the observation s_i. Filters are readily shown as directed graphs, in which the states are vertices and edges are determined by the transition function. Recall the examples in Figure 1b–c.

In this definition, we use symbol $?$ to mean null, and if for a state-observation pair $(v; y)$ it holds that $(v; y) = ?$, then we mean that state v does not have any outgoing transition labeled y. We interpret this to mean that we can be sure that observation y will not, because of some structure in the robot's environment, occur when the filter is in state v. In the graph view, there would simply be no outgoing edge from v labeled y.

Note that Definition 1 ensures that from any state, for any observation, at most one transition can happen. The next definition makes this idea more precise.

Definition 2: *Let* $F = (V; Y; C; \ ; c; v_0)$ *be a filter,* $v \ 2 \ V$ *be a state, and* $s = s_1 s_2 \quad s_n \ 2 \ Y$ *be an observation sequence where each* s_i *is a member of* Y*. We say that* s *is* trackable *from* v *if there is a sequence of states* $q_0; q_1; :::; q_n$ *such that:*

$q_0 = v$*, and*
$(q_i; s_{i+1}) = q_{i+1}$ *for all* $0 \quad i < n$*.*

Notice in particular that if s is trackable from some state v, then the corresponding state sequence $q_0; q_1; :::; q_n$ is unique. Given a state $v \ 2 \ V$ and an observation sequence $s \ 2 \ Y$ trackable from v, we write $(v; s)$ to denote the state reached by tracing s starting from v. If s is not trackable from v, we write $(v; s) = ?$. For the empty string , we define $(v;) = v$ for all states v, and define that is trackable for all states v. For a state $v \ 2 \ V$, we use S_v to denote the set of all observation sequences that end in v when traced from the initial state. i.e., $S_v = fs \ 2 \ Y \quad j \quad (v_0; s) = vg$

We can now define the language of a filter, which plays a crucial role in filter reduction.

Definition 3: *The* language of a state v*, denoted* $L(v)$*, is the set of all observation sequences trackable from* v*. The* language of a filter F*, denoted* $L(F)$*, is the language of its initial state:* $L(F) = L(v_0)$.

Before we can speak meaningfully about reduction of filters, we need a definition of filter equivalence with respect to a language.

Definition 4: *Let* $F_1 = (V_1; Y; C; \ _1; c_1; v_0)$ *and* $F_2 = (V_2; Y; C; \ _2; c_2; w_0)$ *be two filters with the same observation space* Y *and the same color space* C*. Let* $L \quad Y$ *denote a language of observation sequences. We say that* F_1 is equivalent to F_2 with respect to L*, denoted* $F_1 \overset{L}{=} F_2$*, if for any observation sequence* $s \ 2 \ L$*:*

$_1(v_0; s) \ 6= \ ?$,
$_2(w_0; s) \ 6= \ ?$, *and*
$c_1(\ _1(v_0; s)) = c_2(\ _2(w_0; s))$.

This definition says that any observation sequence that is in L is trackable by both F_1 and F_2, and that both of them produce the same output while tracing that sequence. However, for observation sequences that are not in L, it does not say anything about the outputs generated by the two filters, nor does it require that the observation languages of F_1 and F_2 should be the same. The central problem this work studies is called the *filter minimization problem*.

Problem: Filter minimization (FM) [36]
Input: A filter F.
Output: A filter F such that $F \overset{L(F)}{=\!=} F$ and the number of states in F is minimal.

Notice that this problem allows the language $L(F \ )$ of the optimally reduced filter to be a proper superset of the language $L(F)$ of the original filter. This is reasonable because observation sequences in $L(F \ ) \quad L(F)$ will not occur, as we assumed above.

We also study a variant of FM in which we require the reduced filter to have a special property which we define as follows.

Definition 5: *Let* $F_1 = (V_1; Y; C; \ _1; c_1; v_0)$ *and* $F_2 = (V_2; Y; C; \ _2; c_2; w_0)$ *be two filters. Denoted* $F_1 \ \$ \ F_2$, *we say that* F_2 partitions the state space *of* F_1 *if for each* $v \ 2 \ V_1$, *there is a single state* $w \ 2 \ V_2$ *such that for any observation sequence* $s \ 2 \ Y \ ^*$, *if* $_1(v_0; s) = v$, *then* $_2(w_0; s) = w$.

We call this variant, the *filter partitioning minimization problem*.

Problem: Filter partitioning minimization (FPM)

Input: A filter F.

Output: A filter F such that $F \overset{L(F)}{=\!=} F$, $F \ \$ \ F$, and the number of states in F is minimal.

FPM requires not only the reduced filter to produce for each observation sequence trackable by the original filter, the same output produced for that observation sequence by the original filter but it also enforces that for each state s of the original filter, only a single state in the reduced filter to play the role of s.

Because much of what follows deals with relations over the set of a filter's states, we will rely on some elements of standard notation for such relations.

For a given filter F, we use I_F to denote the identity relation on the state set V of F, i.e. $I_F = f(v; v) \ j \ v \ 2 \ V g$. If $R \quad V \quad V$ is an equivalence relation on V, then it partitions V into a set of equivalence classes. For any $v \ 2 \ V$, the equivalence class of v in R is denoted $[v]_R$, so that $[v]_R = fw \ 2 \ V \ j \ (v; w) \ 2 \ Rg$. In particular, for any $v; w \ 2 \ V$, if $(v; w) \ 2 \ R$, then $[v]_R = [w]_R$. Finally, the set of all equivalence classes of R is called the quotient of V under R, denoted V/R.

IV. Filter minimization via quotient operations

In this section, we show how the process of filter reduction can be understood as a quotient operation with respect to certain kinds of relations over the states of the input filter. Accordingly, each of the filter reduction problems, FM and FPM, can be solved by finding some optimal relation over the state space of the original filter.

A. Exact solution to FPM

For FPM we consider relations that indicate which pairs of states should be 'merged' to form a reduced filter. Therefore, we must establish conditions on the relation that guarantee that this merging operation makes sense.

Definition 6: *Let* $F = (V; Y; C; \ ; c; v)$ *be a filter and let* $R \quad V \quad V$ *denote a relation over the states of* F. *We say that* R *is a* compatibility relation *for* F, *if for any* $(v; w) \ 2 \ R$:

1) $c(v) = c(w)$, *and*
2) *for any* $y \ 2 \ Y$, *if* $(v; y) \ 6= \ ?$ *and* $(w; y) \ 6= \ ?$, *then* $(\ (v; y); \ (w; y)) \ 2 \ R$.

Accordingly, two states are said to be *compatible* if they are related by a compatibility relation. To illustrate the notion of compatibility, consider filter F_3, depicted in Figure 5. Some compatibility relations for F_3 are $R_1 = \ ;$, $R_2 = f(v_0; v_1)g$, and $R_3 = f(v_3; v_2); (v_5; v_4); (v_5; v_5); (v_2; v_3); (v_4; v_5)g$. [1]

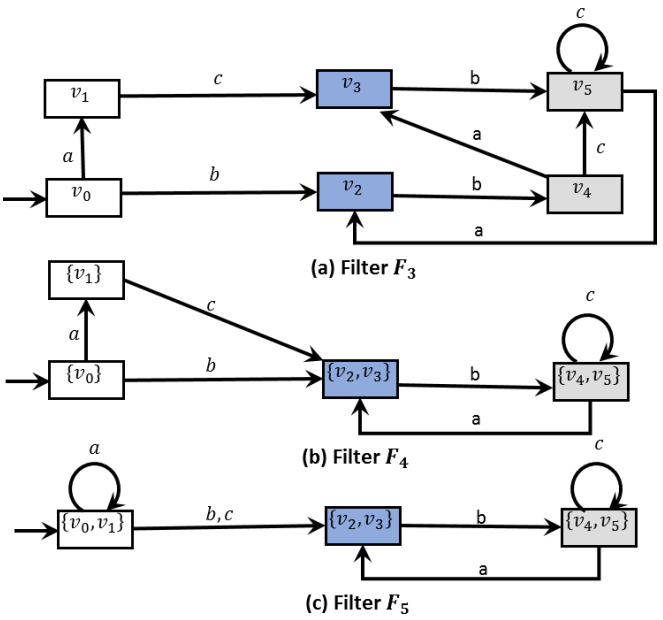

Fig. 5. **a)** A sample filter F_3. **b)** A minimal filter F_4 such that $F_3 \overset{L(F_3)}{=\!=} F_4$ and $L(F_3) = L(F_4)$. **c)** A minimal filter F_5 such that $F_3 \overset{L(F_3)}{=\!=} F_5$.

Now we can define the notion of a quotient filter with respect to a compatibility equivalence relation.

Definition 7: *For a filter* $F = (V; Y; C; \ ; c; v_0)$, *and a relation* $R \quad V \quad V$ *that is both a compatibility relation and an equivalence relation (henceforth, a compatibility equivalence relation), the* quotient of F under R *is the filter* $F/R = (V/R; Y; C; \ ^0; c^0; [v_0]_R)$, *in which*

$$ ^0([v]_R; y) = \begin{cases} [\ (w; y)]_R & \text{if } 9w \ 2 \ [v]_R \text{ with } (w; y) \ 6= \ ? \\ ? & \text{otherwise} \end{cases} $$

and $c^0([v]_R) = c(v)$.

Note that Definition 6 ensures that every transition in a quotient filter is well-defined. Because R must be a compatibility relation, if two states v and w that share some outgoing observation y are merged, then the resulting states $(v; y)$ and $(w; y)$ must be merged as well. Consider filter F_3 depicted in Figure 5. A compatibility equivalence relation for this filter is $R = I_{F_3} \ [\ f(v_3; v_2); (v_5; v_4); (v_2; v_3); (v_4; v_5)g$. The quotient of F_3 under this relation, F_3/R, is filter F_4, depicted in Figure 5.

The next three lemmas establish that, though this quotient operation may increase the language of the filter, it does not change the behavior, in the sense of Definition 4.

Lemma 1: For any filter $F = (V; Y; C; \ ; c; v_0)$ *and any compatibility equivalence relation* R *for* F, $L(F) \ L(F/R)$.

[1] Note that the notion of compatibility relation is different from the usual notion of simulation, in that its second condition is weaker than is required of a simulation relation. In fact, two states can be compatible (can exist in a compatibility relation) while neither of them simulates another. We discuss simulation in more detail in Section VIII.

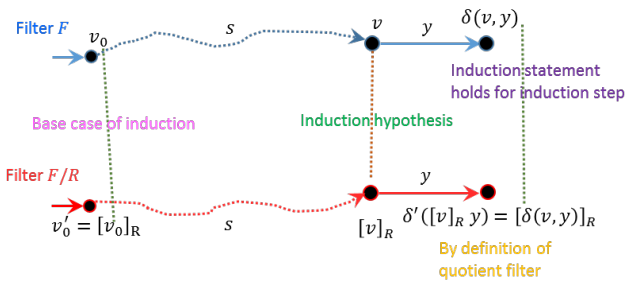

Fig. 6. It illustrates the induction in the proof of Lemma 1, Lemma 2, and Lemma 3. The induction is on the lengths of observation sequences and it states that for each observation sequence s, if filter F reaches state v by tracing s from its initial state, then filter F/R, the quotient of F under a compatibility equivalence relation R, reaches state $[v]_R$ by tracing the same observation sequence from its own initial state.

Lemma 2: For any filter $F = (V; Y; C; \ ; c; v_0)$ *and any compatibility equivalence relation* R *for* F, $F \overset{L(F)}{=\!=} F/R$.

Lemma 3: For any filter $F = (V; Y; C; \ ; c; v_0)$ *and any compatibility equivalence relation* R *for* F, $F \$ F/R$.

Proof: Let us prove Lemmas 1, 2 and 3 all together. Assume F/R is defined according to Definition 7. By induction on the length of observation sequences $s \in Y$ we show that if $s \in L(F)$ and $(v_0; s) = v$ for a $v \in V$, then $^{0}([v_0]_R; s) = [v]_R$. This means that for each state $v \in V$, there is a single state $w = [v]_R$ in F/R such that for each $s \in L(F)$, if $(v_0; s) = v$, then $^{0}([v_0]_R; s) = w$, which by Definition 5 means that $F \$ F/R$, proving Lemma 3. It also implies that if $s \in L(F)$, then $s \in L(F/R)$, meaning that $L(F) \quad L(F/R)$, proving Lemma 1. Furthermore, the conclusion that $^{0}([v_0]_R; s) = [v]_R$ will imply that $c(\ (v_0; s)) = c^{0}(\ ^{0}([v_0]_R; s))$ given that $c(v) = c^{0}([v]_R)$ by Definition 7. This, coupled with Lemma 1 and Definition 4, proves Lemma 2.

Now, we turn to the induction, which is illustrated in Figure 6. The statement holds for the base case, i.e., $s = \ $, since by definition it holds that $\in L(F)$, $(v_0; \ ) = v_0$, and $^{0}([v_0]_R; \ ) = [v_0]_R$. Assume that the statement holds for all observation sequences s with length k, and let s^{0} be any observation sequence with length $k+1$. If $s^{0} \in L(F)$, then $s^{0} = sy$ for an observation sequence s with length k and an observation $y \in Y$. Let $(v_0; s) = v$ for a state v. By the induction assumption, it holds that $^{0}([v_0]_R; s) = [v]_R$. Now, we have that $(v_0; s^{0}) = (\ (v_0; s); y) = (v; y)$, and that $^{0}([v_0]_R; s^{0}) = {}^{0}(\ ^{0}([v_0]_R; s); y)) = {}^{0}([v]_R; y)$. Given that $s^{0} \in L(F)$, it holds that $(v; y) \not\in ?$. This by the definition of ^{0} in Definition 7 implies that $^{0}([v]_R; y) = [\ (v; y)]_R$. Now, the inductive step is proved since $(v_0; s^{0}) = (v; y)$ and $^{0}([v_0]_R; s^{0}) = [\ (v; y)]_R$. ■

Next we prove that the state space of any optimal solution to FPM can be seen as the quotient of the state space of the original filter under some compatibility equivalence relation. Let the input to FPM be a filter $F_1 = (V_1; Y; C; \ _1; c_1; v_0)$ and let $F_2 = (V_2; Y; C; \ _2; c_2; w_0)$ be on optimal solution of FPM with input F_1. Given that $F_1 \$ F_2$, for each $v \in V_1$ there exists a single state $w \in V_2$ such that

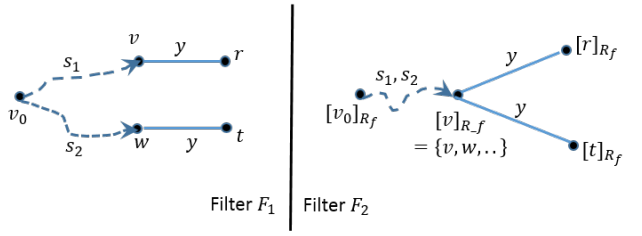

Fig. 7. An illustration of the proof of Lemma 5. Filter F_2 is a minimal filter for which $F_1 \overset{L(F_1)}{=\!=} F_2$ holds. The state space of F_2 corresponds to the quotient of the state space of F_1 under the equivalence relation R_f. The assumption for this relation is $(v, w) \in R_f$ but $(r, t) \notin R_f$.

for any observation sequence s, if $_1(v_0; s) = v$, then $_2(w_0; s) = w$. Let $f : V_1 \ ! \ V_2$ denote that one-to-one mapping, i.e., for every observation sequence $s \in L(F_1)$, $_2(w_0; s) = f(\ _1(v_0; s))$. We consider the following results.

Lemma 4: Function f *is surjective.*

Proof: We prove that each state in F_2 is mapped to by at least one state in F_1 via f. For the sake of contradiction, suppose that there exists a state z in F_2 that is not mapped to by f from any state in F_1. There are two cases.

1) If no observation sequence that ends or passes through z is in $L(F_1)$, then we can construct a new filter F_3 from F_2 by removing state z. Clearly, $F_1 \overset{L(F_1)}{=\!=} F_3$ and F_3 has fewer states than F_2. This contradicts the construction that F_2 is minimal.
2) If there exists an observation sequence $s \in L(F_1)$ that ends or passes through z when traced in F_2, then let $k \quad jsj$ be an integer such that $_2(w_0; s_{1:::k}) = z$. By the structure of filters, and given that $s \in L(F_1)$, we conclude that $s_{1:::jsj \ 1} \in L(F_1)$, $s_{1:::jsj \ 2} \in L(F_1)$, ..., and ultimately $s_{1:::k} \in L(F_1)$. This means that $z = f(\ _1(v_0; s_{1:::k}))$, which is a contradiction.

■

Given such a function f, we define an equivalence relation $R_f \quad V \quad V$ so that $(v; w) \in R_f$ if and only if $f(v) = f(w)$. Note that there is a one-to-one correspondence between the equivalence classes $[v]_{R_f}$ of R_f and the states of F_2.

Lemma 5: For any filter F_1 *and any optimal solution* F_2 *to FPM with input filter* F_1*, the equivalence relation* R_f *they induce is a compatibility relation.*

Proof: First observe that by the construction of R_f, for any $v; w \in V$, if $(v; w) \in R_f$, then v and w are mapped to a single state in F_2. Let $[v]_{R_f}$ be such a state. To show that R_f is a compatibility relation, we prove that conditions (1) and (2) of Definition 6 hold for any v and w for which $(v; w) \in R_f$. Suppose that condition (1) does not hold, that is, $c_1(v) \not\in c_1(w)$, which means $c_2([v]_R)$ is different from $c_1(v)$ or $c_1(w)$. Without loss of generality assume that $c_1(v) \not\in c_2([v]_{R_f})$. Let $s \in L(F_1)$ be an observation sequence such that $_1(v_0; s) = v$. By definition of f, we have that $_2(w_0; s) = [v]_{R_f}$. But, $c_1(\ _1(v_0; s)) = c_1(v) \not\in c_2([v]_{R_f}) = c_2(\ _2(w_0; s))$, which by Definition 4 contradicts that $F_1 \overset{L(F_1)}{=\!=} F_3$.

Now suppose that condition (2) does not hold, which means that there exists $y \in Y$, such that $\delta_1(v,y) \notin ?$ and $\delta_1(w,y) \notin ?$ but $(\delta_1(v,y), \delta_1(w,y)) \in R_f$. Let $r = \delta_1(v,y)$ and $t = \delta_1(w,y)$. We argue that if this is the case, then F_2 is not a filter, which is a contradiction. Figure 7 illustrates this proof. Let s_1 and s_2 be two observation sequences that end in v and w, respectively, when traced by F_1. This means that states v and w are in the same equivalence class of R_f, and thus, they are mapped to a single state, such as $[v]_{R_f}$, in F_2; hence, both s_1 and s_2 end in $[v]_{R_f}$ when traced by F_2. Consider also that observation sequences s_1y and s_2y end in r and t, respectively, when traced by F_1. Observe that from state $[v]_{R_f}$ there should be two outgoing edges with the same label y, one of which goes to $[r]_{R_f}$ and another goes to $[t]_{R_f}$. Because F_2 is a filter, the only way to reach r by tracing s_1y from the initial state is to have an edge labeled by y, that goes from $[v]_{R_f}$ to $[r]_{R_f}$. We can use the same argument to prove that there should be an outgoing edge labeled by y that connects $[v]_{R_f}$ to $[t]_{R_f}$. This implies that F_2 has two edges labeled y from $[v]_{R_f}$, a contradiction. ■

In particular, since R_f is both an equivalence relation and a compatibility relation for F_1, it is meaningful to consider the quotient filter F_1/R_f. Moreover, F_1/R_f is structurally identical to the minimal filter F_2. Of course, in the context of filter partitioning minimization, F_2 is unknown, so we cannot expect to compute F_1/R_f directly.

Nevertheless, the impact of Lemma 5 is that we can view the problem of filter partitioning minimization as equivalent to the problem of identifying a suitable compatibility equivalence relation with which to construct a quotient filter — there always exists some such relation for which the quotient leads to the minimal filter. This directly implies the following result.

Theorem 1: Let F be a filter and let R be a compatibility equivalence relation for F with minimum number of partitions. The filter $F' = F/R$ is a minimal filter for which $F \overset{L(F)}{=\!=} F'$ and $F' \not\subset F$ holds.

Proof: By Lemma 2 and that R is a compatibility equivalence relation, $F \overset{L(F)}{=\!=} F'$. If F/R is not minimal, that is if there is another filter F_2 with fewer number of states than F/R, then by Lemma 5, the relation R would not be a compatibility equivalence relation with minimum number of equivalence classes. ■

B. Exact solution to FM

Zhang and Shell [64] proposed an algorithm for computing an exact solution to FM. Their algorithm first constructs the *compatibility graph* of filter, which is an undirected graph whose vertices are the states of filter and its edges connect pairs of distinct compatible states. Then, it forms a SAT formulation that aims to find a clique covering of the graph with minimum number of classes subject to a list of zipper constraints where each constraint enforces that if a set of states are merged, then the set of states they go by an observation must also be merged. An exact solution to

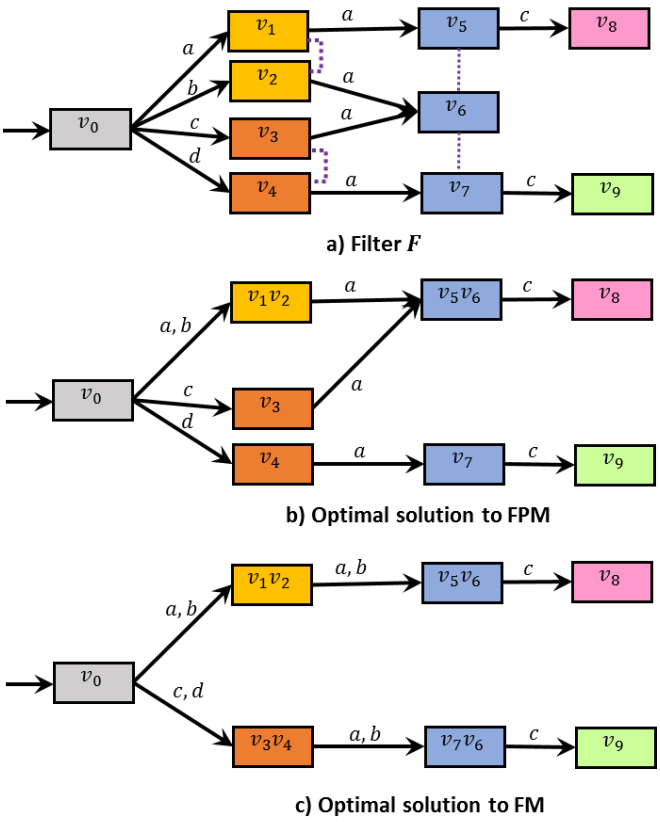

Fig. 8. **a)** A sample filter F. **b)** A minimal solution of FPM for filter F. **c)** The minimal solution of FM for filter F.

FPM is computed by finding over the same graph, a clique partitioning rather than a clique covering. The difference between a partitioning and a covering is that a state is assigned to one and only one class of the partitioning while a state can be shared between two or several classes of the covering.

To see how this differentiates between an optimal solution to FM and an optimal solution to FPM, we use a filter similar to that Zhang and Shell [64] used to demonstrate an optimal solution to FM. This filter, named F, is shown in Figure 8a. In this filter, those states that are compatible are connected by dashed lines. In this filter, states v_1 and v_2 are compatible, v_3 and v_4 are compatible, and v_6 is compatible with both v_5 and v_7. Note that v_5 and v_7 are not compatible with each other. FPM with input F has two optimal solutions, in one of which, v_6 is merged with v_5, and in the other one, v_6 is merged with v_7. Figure 8b shows the one in which v_6 is merged with v_5.

FM for the same filter has a single optimal solution, which is shown in Figure 8c. Note that because an optimal solution to FPM is computed by finding a partitioning rather than a covering, in a solution to FPM state v_6 is merged either with v_5 or with v_7, while because an optimal solution to FM is computed by finding a covering, in the optimal solution to FM, state v_6 is split between two compatibility classes $\{v_5, v_6\}$ and $\{v_6, v_7\}$.

We now formalize an exact solution of FM via the notions

of relation and covering, but before that we first consider the following definitions.

Given a filter $F = (V; Y; C; ; c; v_0)$, a *compatibility class* over V is a non-empty set $L \quad V$ in which each pair of states are compatible. A set of compatibility classes $\mathbb{M} = \{L_1; L_2; \quad ; L_k\}$ over V is closed if for every $y \in Y$ and $i \in \{1; \quad ; k\}$, there exists a $j \in \{1; \quad ; k\}$ such that $(v; y) \quad L_j$. We use $_{\mathbb{M}}(L_i; y)$ to represent one of such compatibility classes L_j. A closed covering for V is a closed set of compatibility classes $\mathbb{M} = \{L_1; L_2; \quad ; L_k\}$ such that for each $v \in V$, there exists at least an integer $i \in \{1; \quad ; k\}$ such that $v \in L_i$. Note that a closed covering for V always forms a *tolerance* relation, a reflexive and symmetric relation, which is formed by relating for each compatibility class of the covering, all pairs of states within that class and also relating each state with itself.

Given a closed covering of the state space of the filter, we can form a quotient filter as follows.

Definition 8: *For a filter $F = (V; Y; C; ; c; v_0)$, and a closed covering $\mathbb{M} = \{L_1; L_2; \quad ; L_k\}$ of V, the* quotient of F under $\mathbb{M}$ *is the filter $F/\mathbb{M} = (\mathbb{M}; Y; C; {}^0; c^0; L_0)$, in which L_0 is a compatibility class such that $v_0 \in L_0$, and for each $L \in \mathbb{M}$, ${}^0(L; y) = {}_{\mathbb{M}}(L; y)$ if ${}_{\mathbb{M}}(L; y) \neq ?$; and otherwise ${}^0(L; y) = ?$, and $c^0(L) = c(v)$, where v is a state within L.*

Accordingly, an optimal solution to FM is obtained by computing a closed covering with minimum number of classes.

Lemma 6: Let F be a filter with state space V and let $\mathbb{M}$ be a closed covering of V with minimum number of compatibility classes. The filter $F = F/\mathbb{M}$ is a minimal filter for which $F \overset{L(F)}{=\!=} F$ holds.

Proof: The proof, which we omit, combines similar results and proofs of Lemma 1, Lemma 2, and Lemma 5. ■

Consider that any compatibility equivalence relation over the state space of the filter is a closed covering of the state space, and therefore, any feasible solution to FPM is also a feasible solution to FM. As result, any heuristic or greedy algorithm for FPM can be used to compute a feasible solution to FM. The next section studies how big the difference between the sizes of optimal solutions to FPM and FM can be.

V. Maximum difference between the sizes of optimal solutions to FM and FPM

In this section, we show that there exist filters for which the difference between the sizes of optimal solutions to FM and FPM can be in the order of the size of the filter.

Theorem 2: For any $n \quad 10$, there exists a filter F with n states such that the difference between the sizes of an optimal solution to FPM with input F and an optimal solution to FM with input F is $b\frac{n \quad 6}{4}c$.

Proof: Let n be an integer such that $n \quad 10$. We first consider the case where $n \quad 6$ is divisible by 4, that is,

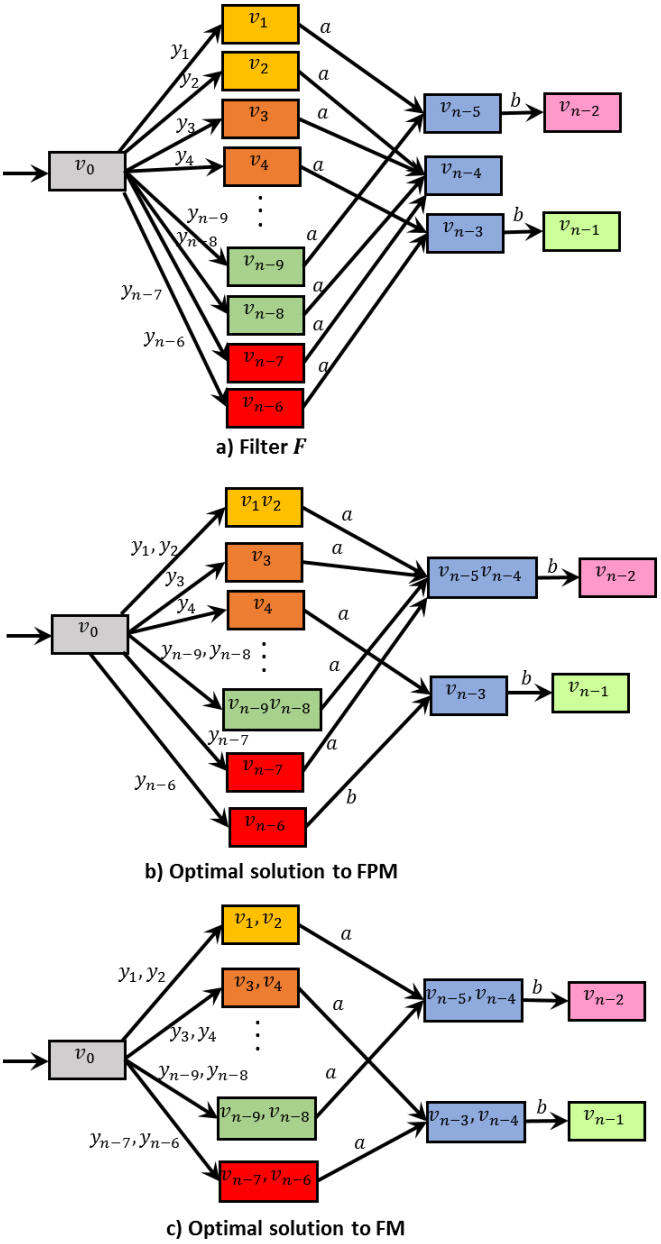

Fig. 9. An illustration of the proof of Theorem 2. **a)** A sample filter F with n states. **b)** An optimal solution of FPM with input filter F. **c)** An optimal solution of FM with input filter F.

where n is one of the numbers $10; 14; 18; 22; 26; : : :$. For that n, we construct the filter F, which has n states, as shown in Figure 9a. This filter generalizes the filter we used in Figure 8a. In fact, the number 10 in this theorem comes from the fact that the filter in Figure 8a has 10 states and that the filter in Figure 8a is the smallest filter within the family of filters described by Figure 9a. To color the states of F, we have used $\frac{n \quad 6}{2} + 4$ distinct colors: 1 color for v_0; 1 color for $v_{n \quad 1}$; 1 color for $v_{n \quad 2}$; 1 color for $v_{n \quad 5}; v_{n \quad 4}$, and $v_{n \quad 3}$; and the remaining $\frac{n \quad 6}{2}$ colors for v_1 through $v_{n \quad 6}$. Note that the states in the second column, states v_1 through $v_{n \quad 6}$, appeared in pairs such that the two states of each pair have the same color. Observe that in filter F, for each color, except

color blue, all states with that color are compatible. The states that have color blue are v_{n-3}, v_{n-4}, and v_{n-5}. States v_{n-3} and v_{n-5} are not compatible with each other, but state v_{n-4} is compatible with each of them. Each pair of the states that appear in the second column have the same color and the two states within each of those pairs are compatible only with themselves, that is, for each $i \in \{1, 3, 5, \ldots, n-7\}$, v_i is compatible with v_{i+1} and only with v_{i+1}. Half of those pairs have outgoing edges to state v_{n-4} and v_{n-5}, and the other half have outgoing edges to v_{n-3} and v_{n-5}. In an optimal solution to FPM, state v_{n-4} can be merged with either v_{n-3} or v_{n-5}. Part b of Figure 9 shows an optimal solution to FPM. In that solution, state v_{n-4} is merged with v_{n-5}, and as a result, of the pairs in the middle column, only those who have outgoing edges to v_{n-4} and v_{n-5} are merged. Accordingly, only half of the pairs in the middle column, or more precisely, $\frac{n-6}{4}$ states, are merged in the optimal solution to FPM. The FM with input filter F in this figure has only one optimal solution, which is shown in part c of this figure. Consider that since an optimal solution of FM is obtained by finding a covering of the state space rather than a partitioning of it, the states with color blue are split into two compatibility classes $\{v_{n-4}, v_{n-5}\}$ and $\{v_{n-4}, v_{n-3}\}$. State v_{n-4} is shared between the two classes. Because of this, all pairs in the middle column are merged, and thus, the optimal solution to FPM has $\frac{n-6}{4}$ states more than the optimal solution to FM.

Now consider the case where $n \geq 10$ but $n \neq 6$ is not divisible by 4. In this case, we can always choose integers n_1 and j such that $n = n_1 + j$, $n_1 \geq 10$, $n_1 - 6$ is divisible by 4, and $j \in \{1, 2, 3\}$. For example, if $n = 13$, then $n_1 = 10$ and $j = 3$. As another example, if $n = 28$, then $n_1 = 26$ and $j = 2$. For this case, we make a filter F_1 with n_1 states such that F_1 has n_1 states exactly in the form of filter F in Figure 9a and j extra states to each of which there is one and only one transition labeled y_{n_1-6+j} from state v_0 to that state. We opt that none of those states to have outgoing transitions. Additionally, we color those j extra states with j distinct colors that are not used to color other states. None of these extra j states are compatible with any other state. Accordingly, neither in the optimal solution to FM nor in an optimal solution to FPM, those states are merged. Accordingly, for each filter F_1, the difference between the optimal solutions to FPM and FM is $\frac{n_1-6}{4}$. For this case, it holds that $\lfloor \frac{n-6}{4} \rfloor = \frac{n_1-6}{4}$, and as a result, the theorem is proved. ∎

According to this theorem, a feasible solution to FPM, even an optimal one, does not necessarily give a good solution for FM. An impact of this is that FM, which because of being an NP-hard problem we can hope to find in a reasonable amount of time only feasible solutions of it for large filters, requires to be treated by heuristic and greedy algorithms or other kind of techniques that are designed specifically for FM, and those who are proposed for FPM may not compute high-quality solutions for FM.

VI. Bisimulation and filter reduction

In this section, we show that the well-known notion of bisimulation does not always yield optimal solutions to FM and FPM.

In fact, one apparently reasonable hypothesis is that the notion of bisimulation may be useful for solving FM and FPM. We show that although the bisimilarity relation is indeed a compatibility equivalence relation, and hence, can be used to compute feasible solutions to FM and FPM; it does not, in general, induce minimal filters. We begin by adapting the standard notion of bisimulation to filters.

Definition 9: *Let* $F_1 = (V_1, Y, C, \delta_1, c_1, v_0)$ *and* $F_2 = (V_2, Y, C, \delta_2, c_2, w_0)$ *be two (not necessarily distinct) filters. A relation* $R \subseteq V_1 \times V_2$ *is said to be a* bisimulation relation *for* (F_1, F_2) *if for any* $(v, w) \in R$*:*

1) $c_1(v) = c_2(w)$,
2) *for any* $y \in Y$*, if* $\delta_1(v, y) \neq ?$*, then* $\delta_2(w, y) \neq ?$ *and* $(\delta_1(v, y), \delta_2(w, y)) \in R$
3) *for any* $y \in Y$*, if* $\delta_2(w, y) \neq ?$*, then* $\delta_1(v, y) \neq ?$ *and* $(\delta_1(v, y), \delta_2(w, y)) \in R$

We say that state v in filter F_1 is *bisimilar* to state w in filter F_2 if there exists a bisimulation relation R for (F_1, F_2) such that $(v, w) \in R$ or equivalently if $v \sim_{(F_1,F_2)} w$, where relation $\sim_{(F_1,F_2)}$, which itself is a bisimulation relation for (F_1, F_2), is the union of all bisimulation relations for (F_1, F_2). The notion of bisimulation can also be lifted to the filters themselves, according to which F_1 and F_2 are bisimilar, denoted by $F_1 \sim F_2$, if there exists a bisimulation relation R for (F_1, F_2) such that $(v_0, w_0) \in R$.

We use bisimulation between two filters in the next section; in this section, we are concerned only about bisimulation relations that relate states within a single filter, that is, where in Definition 9, $F_1 = F_2 = F = (V, Y, C, \delta, c, v_0)$. Observe that any union of bisimulation relations for a filter is itself a bisimulation relation. The union of *all* bisimulation relations for F, denoted $\sim_F$, is called the *bisimilarity relation* for F. Recall filter F_3, depicted in Figure 5. For this filter, we have $\sim_{F_3} = I_{F_3} \cup \{(v_2, v_3), (v_3, v_2), (v_4, v_5), (v_5, v_4)\}$.

Such bisimilarity relations are of interest in part because they are suitable for constructing quotient filters.

Lemma 7: The bisimilarity relation of every filter is both a compatibility relation and an equivalence relation.

Proof: It is well known that the bisimilarity is an equivalence relation (see, for example, Lemma 7.8 of Baier, Katoen, and Larsen [4] for a proof). Also, by Definition 9, any bisimulation relation —including the bisimilarity relation— is a compatibility relation in the sense of Definition 6. ∎

Because the bisimilarity relation of a given filter represents, in a certain sense, a coarsest partitioning of the states into 'mergeable' subsets, intuition might suggest that a quotient with the bisimilarity relation might perhaps produce an optimally reduced filter, in the sense of the FM or the FPM problem. The next result debunks this misconception.

Theorem 3: For any integer $n \geq 1$*, there exists a filter* F_n *with* $n + 2$ *states, such that* $F_n / \sim_{F_n}$ *is larger than the optimal solution* F_n *to FM by* n *states. The same filter*

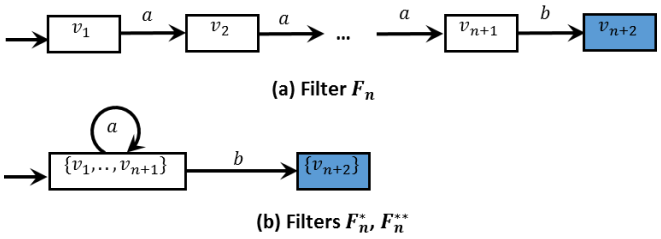

Fig. 10. a) The construction of filter F_n, mentioned in Theorem 3. The quotient of this filter under $\sim_{F_n}$ does not reduce its size. b) Filters F_n and F_n , which are identical, are respectively the optimal solutions to FM and FPM with input F_n. State v_{n+2} in filter F_n and state $\{v_{n+2}\}$ in filters F_n and F_n have color 2; all other states in both filters have color 1.

F_n= $_{F_n}$ is larger than the optimal solution F_n to FPM by n states.

Proof: For a given n, we construct a filter F_n with $n+2$ states, for which F_n= $_{F_n}$ also has $n+2$ states. Figure 10a shows the construction. In particular, note that for any pair of distinct states $(v;w)$, we have v $_{F_n}$ w; this is because if v $_{F_n}$ w, then they must have the same color, meaning that there must exist 1 i 6= j $n+1$ such that $v = v_i$ and $w = v_j$, and if this is the case then by the definition of bisimulation relation, we must have that v_{i+1} $_{F_n}$ v_{j+1}, v_{i+2} $_{F_n}$ v_{j+2}, ..., and ultimately v_{i+k} $_{F_n}$ v_{n+2}, which is a contradiction. Therefore $_{F_n}$= I_{F_n}, and F_n= $_{F_n}$ is structurally identical to F_n — no two states will be merged. In contrast, for any n, each of the optimally reduced filters F_n and F_n has exactly two states, as shown in Figure 10b. ■

In particular, Theorem 3 implies that bisimulation-quotienting does not always induce optimal solutions to the FM and the FPM problems, and in fact, that the difference in size between the optimally reduced filters and the bisimilarity-quotient filter cannot be bounded.

Note that by Definition 9, for two states to be bisimilar, not only they must agree on their outputs but also for each observation, either both of them must have an outgoing transition labeled with that observation or none of them must have an outgoing transition for that observation. Accordingly, bisimulation imposes a strong condition for two states to be merged, and this, results to the fact that bisimulation does not yield enough reduction for some filters. Yet, we can still use bisimulation to make an extent of reduction, but for real-world applications in robotics, we may not be satisfied with the amount of reduction we achieve with bisimulation. Both the examples in this theorem and Figure 1 show that for some filters, bisimulation does not reduce the size of the filter at all while the minimal filter is much smaller than the original filter. Intuitively, if the underlying system or the problem of interest does not enjoy a big deal of symmetry and indistinguishability, then bisimulation does not offer much reduction. To illustrate these notions of symmetry and indistinguishability, consider an environment similar to that in Figure 1 but in which the four landmarks are on the four corners of a square and the robot could not distinguish between some landmarks. It is usually for the filters of this kind of situations happens that some states become bisimilar, and thus, bisimulation can help to reduce the size of the filter. These notions also arise in problems similar to that in Figure 4, where the shape of the environment is symmetric and in which when a robot crosses a beam, the system knows which beam sensor it was except that it cannot distinguish between some beams.

VII. Strong filter minimization

In this section, we introduce a variant of filter minimization problem for which the bisimilarity relation always produces an optimal solution.

Section VI showed that, although quotient with the bisimilarity relation produces an equivalent filter, that filter may not necessarily be minimal. In this section, we provide some insight into why that happens, by showing that this kind of bisimilarity quotient instead solves a variant of the filter minimization problem, in which the language of the reduced filter must be identical to the language of the original filter, rather than merely a superset of it. Specifically, this section shows that bisimilarity-quotienting solves the following problem.

Problem: Strong Filter Minimization (SFM)

Input: A filter F.

Output: A filter F such that $F \overset{L(F)}{=\!=} F$, $L(F) = L(F$), and the number of states in F is minimal.

Now we can state the main result of this section.

Theorem 4: For any filter F, the bisimilarity quotient F= $_F$ is a solution to the SFM problem for F.

Proof: It suffices to prove only that SFM is equivalent to the problem of finding a minimal filter F such that F ' F . The rest is due to the well-known fact about bisimulation minimization, that the quotient of a structure under its bisimilarity relation is a smallest structure that is bisimilar to the original structure. For a proof of this result in the context of transition systems, see [4]. The only difference here is that filters are concerned with observation sequences with finite-length rather than observation sequences with infinite-length.

Therefore, we need to prove that F ' F if and only if $F \overset{L(F)}{=\!=} F$ and $L(F) = L(F$). Let $F = (V;Y;C;\ ;c;v_0)$ and F = $(V^0;Y;C;\ ^0;c^0;v_0^0)$. For the direction) , assume by contradiction that F ' F , that is, v_0 $_{(F;F}$ $_{)}$ v_0^0, but either $L(F)$ 6= $L(F$) or for an observation sequence s 2 $L(F) \setminus L(F$), $c($ $(v_0;s))$ 6= $c^0($ ^{0} $(v_0^0;s))$. Figure 11 shows the proof.

For the former case, assume, without loss of generality, that s be an observation sequence such that s 2 $L(F)$ but s 2 $L(F$). Let integer k, where 1 $k < |s|$, be the smallest index such that ^{0} $(v_0^0;s_{1::k}) = ?$ and let $y = s_{k::k}$. Clearly, $($ $(v_0;s_{1::k\ 1});y)$ 6= $?$ but $^0($ ^{0} $(v_0^0;s_{1::k\ 1});y)$ = $?$. This by the second condition of Definition 9 implies that $(v_0;s_{1::k\ 1})$ 6 $_{(F;F}$ $_{)}$ ^{0} $(v_0^0;s_{1::k\ 1})$, which, in turn, by the same condition of that definition implies that $_1(v_0;s_{1::k\ 2})$ 6 $_{(F;F}$ $_{)}$ $_2(v_0^0;s_{1::k\ 2})$. Applying the same

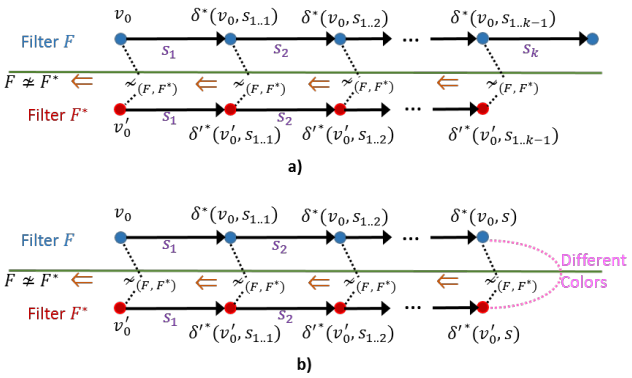

Fig. 11. Part of the proof of Theorem 4, which shows by contradiction that if $F \simeq F$ then $F \overset{L(F)}{=\!=} F$ and $L(F) = L(F)$. **a)** The contradiction assumption that $L(F) \neq L(F)$. Without loss of generality, we assume s to be an observation sequence in $L(F)$ but not in $L(F)$. Integer k is assumed to be the smallest integer such that $s_{1::k} \notin L(F)$ but $s_{1::k-1} \in L(F)$. Given that state $\delta(v_0, s_{1::k-1})$ has a transition for observation s_k but state $\delta'(v'_0, s_{1::k-1})$ does not have a transition for that observation, by the second condition of Definition 9, those two states are not bisimilar, i.e., $\delta(v_0, s_{1::k-1}) \not\sim_{(F;F')} \delta'(v'_0, s_{1::k-1})$, and this, via a series of implications ($\delta(v_0, s_{1::k-2}) \not\sim_{(F;F')} \delta'(v'_0, s_{1::k-2})$, $\delta(v_0, s_{1::k-3}) \not\sim_{(F;F')} \delta'(v'_0, s_{1::k-3})$, ...) implies that v_0 is not bisimilar with v'_0, meaning that F and F' are not bisimilar, which is a contradiction. A similar argument applies for when we assume that there is an observation sequence s such that $s \in L(F)$ but $s \notin L(F)$. **b)** The contradiction assumption that for an observation sequence $s \in L(F) \cap L(F')$, $c(\delta(v_0, s)) \neq c'(\delta'(v'_0, s))$. Given that those two states $\delta(v_0, s)$ and $\delta'(v'_0, s)$ do not have the same color, by the first condition of Definition 9, they are not bisimilar, i.e., $\delta(v_0, s) \not\sim_{(F;F')} \delta'(v'_0, s)$ and this, via a series of implications ($\delta(v_0, s_{1::jsj-1}) \not\sim_{(F;F')} \delta'(v'_0, s_{1::jsj-1})$, $\delta(v_0, s_{1::jsj-2}) \not\sim_{(F;F')} \delta'(v'_0, s_{1::jsj-2})$, ...), implies that $v_0 \not\sim_{(F;F')} v'_0$, meaning that F is not bisimilar to F', which is a contradiction.

condition $k-3$ more times implies that $v_0 \not\sim_{(F;F')} v'_0$, which means that $F \not\simeq F'$, another contradiction.

For the later case, given that $c(\delta(v_0;s)) \neq c'(\delta'(v'_0;s))$, by the first condition of Definition 9 it holds that $\delta(v_0;s) \not\sim_{(F;F')} \delta'(v'_0;s)$, which by the second condition of the same definition implies that $\delta(v_0;s_{1::jsj-1}) \not\sim_{(F;F')} \delta'(v'_0;s_{1::jsj-1})$, which in turn implies that $\delta(v_0;s_{1::jsj-2}) \not\sim_{(F;F')} \delta'(w_0;s_{1::jsj-2})$, $\delta(v_0;s_{1::jsj-3}) \not\sim_{(F;F')} \delta'(w_0;s_{1::jsj-3})$, ..., and finally $v_0 \not\sim_{(F;F')} v'_0$, which means that $F \not\simeq F'$ and is a contradiction.

For the direction $($, we need to prove that if $F \overset{L(F)}{=\!=} F'$ and $L(F) = L(F')$, then $F \simeq F'$. Consider that from $L(F) = L(F')$ and $F \overset{L(F)}{=\!=} F'$, it is implied that for all $s \in L(F)$, $c(\delta(v_0;s)) = c'(\delta'(v'_0;s))$. Accordingly, we consider the relation $R = \bigcup_{s \in L(F)} \{(\delta(v_0;s); \delta'(v'_0;s))\}$. Each tuple of states $(\delta(v_0;s)$ and $\delta'(v'_0;s)$ related by this relation have the same color, thereby satisfying the first condition of Definition 9. Also because $F \overset{L(F)}{=\!=} F'$ and $L(F) = L(F')$, for each observation y, if one of these two states has an outgoing transition labeled y, then the other one also has an outgoing transition labeled y. Then the states $(\delta(v_0;s);y)$ and $(\delta'(\delta'(v'_0;s));y)$ are likewise related by R, satisfying the second and the third conditions of Definition 9. Hence, R is a bisimulation relation for $(F;F')$ in sense of Definition 9. Also, given that $(v_0;v'_0) \in R$, it holds that $v_0 \not\sim_{(F;F')} v'_0$, which means that $F \not\simeq F'$. ■

Consider also that since the bisimilarity relation is a compatibility equivalence relation, by Lemma 3 the optimal solution to SFM always partitions the state space of the original filter, that is, $F \,\$\, F = \sim_F$.

Corollary 1: SFM can be solved in polynomial time.

Proof: Beyond Theorem 4, we need only to show that given a filter $F = (V;Y;C;\delta;c;v_0)$ both (a) the bisimilarity relation $\sim_F$ and (b) the quotient of a filter and a relation, can be computed in polynomial time.

A simple efficient algorithm for constructing the bisimilarity relation starts with assigning the set $\{(v;w) \in V \times V \mid c(v) = c(w) \wedge \forall y \in Y, (\delta(v;y) = \delta(w;y) = ? \vee c(\delta(v;y)) = c(\delta(w;y)))\}$ as the initial value to a variable R. Then, in each iteration of a loop, all members of R that fail to satisfy all three conditions of Definition 9 are removed from R. This loop continues until no additional members of R can be removed; at that time, we have $R = \sim_F$. Clearly, the time complexity of this algorithm is $O(|V|^2 \cdot |Y|)$. This relation has at most $|V|^2$ members, hence, the filter $F/\sim_F$ is constructed in $O(|V|^4 \cdot |Y|)$ time. ■

As an example of this theorem, consider again filter F_3 from in Figure 5. The quotient of this filter under $\sim_{F_3}$ is filter F_4, depicted in the same figure. The language of F_3 is equal to the language of F_4. Filter F_5, depicted in the same figure, represents the smallest filter who is equivalent to F_3 with respect to the language of F_3. In this case, we have $L(F_3) \subset L(F_5)$.

Knowing now that making the quotient of a filter under bisimilarity relation does not always optimally reduce the size of that filter, in the next section, we are interested in realizing the connection between filter reduction and a weaker notion of bisimulation called simulation.

VIII. Simulation and Filter Reduction

In this section, we prove that the FM problem is equivalent to the problem of finding a minimal filter that simulates the original filter.

To do so, we introduce a definition of simulation relation between states of two filters, adapting the standard notion for transition systems.

Definition 10: *Let* $F_1 = (V_1;Y;C;\delta_1;c_1;v_0)$ *and* $F_2 = (V_2;Y;C;\delta_2;c_2;w_0)$ *be two filters. A relation* $R \subseteq V_1 \times V_2$ *is said to be a* simulation relation *for* $(F_1;F_2)$ *if for any* $(v;w) \in R$*:*

1) $c_1(v) = c_2(w)$, *and*
2) *for all* $y \in Y$, *if* $\delta_1(v;y) \neq ?$, *then* $\delta_2(w;y) \neq ?$ *and* $(\delta_1(v;y);\delta_2(w;y)) \in R$

We say that state w *simulates* state v if there exists a simulation relation R for $(F_1;F_2)$ such that $(v;w) \in R$. The union of all simulation relations for $(F_1;F_2)$, which is called the *similarity relation* for $(F_1;F_2)$ and denoted $\sim_{(F_1;F_2)}$, is itself is a simulation relation. It also can be computed in time polynomial to the size of F_1 and F_2.

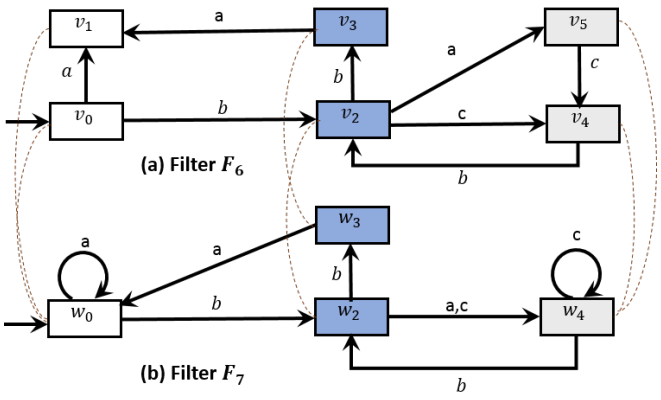

Fig. 12. a) A sample filter F_6 b) An example of a filter that simulates F_6. For both filters, states in the left column have color 1, states in the middle column have color 2, and states in the right column have color 3. The similarity relation for (F_6, F_7) has been depicted by dashed lines.

Just as with bisimulation, the notion of simulation, which is defined thus far between states of two filters, can be lifted to filters themselves. We write $F_1 \quad F_2$, indicating that filter F_2 simulates filter F_1, if there exists a simulation relation R for $(F_1; F_2)$ such that $(v_0; w_0) \; 2 \; R$. Equivalently, $F_1 \quad F_2$ if $v_0 \quad _{(F_1;F_2)} \; w_0$.

To illustrate this notion of simulation, consider filters F_6 and F_7, depicted in Figure 12. Some simulation relations for $(F_6; F_7)$ are:

$R_1 = f(v_1; w_0)g,$
$R_2 = f(v_1; w_0); (v_3; w_3)g,$
$R_3 = f(v_1; w_0); (v_3; w_3); (v_2; w_2); (v_5; w_4); (v_4; w_4)g,$
and
$R_4 = R_3 \; [\; f(v_0; w_0)g.$

Note that $_{(F_6;F_7)} = R_4$.

The following theorem establishes a strong connection between simulation (in the sense of Definition 10) and filter equivalence (in the sense of Definition 4).

Theorem 5: For any filters F_1 and F_2, $F_1 \quad F_2$ if and only if $F_1 \overset{L(F_1)}{=\!=\!=} F_2$.

Proof: We first prove that if $F_1 \quad F_2$ then $F_1 \overset{L(F_1)}{=\!=\!=} F_2$. Assume for a contradiction that the statement $F_1 \quad F_2$ holds but the statement $F_1 \overset{L(F_1)}{=\!=\!=} F_2$ does not hold. By Definition 4, the statement $F_1 \overset{L(F_1)}{=\!=\!=} F_2$ does not hold in two cases: either (1) where $L(F_1) \; * \; L(F_2)$ or (2) where there exists an observation sequence $s \; 2 \; L(F_1) \; \backslash \; L(F_2)$ such that $c_1(\; _1(v_0; s)) \; \not\in \; c_2(\; _2(w_0; s))$.

For the former case, assume that s be an observation sequence such that $s \; 2 \; L(F_1)$ but $s \; \not\in \; L(F_2)$. Let integer k, where $1 \quad k < jsj$, be the smallest index such that $_2(w_0; s_{1::k}) = ?$ and let $y = s_{k::k}$. Clearly, $_1(\; _1(v_0; s_{1::k \; 1}); y) \; \not\in \; ?$ but $_2(\; _2(w_0; s_{1::k \; 1}); y) = ?$. This by the second condition of Definition 10 implies that $_1(v_0; s_{1::k \; 1}) \quad _{(F_1;F_2)} \quad _2(w_0; s_{1::k \; 1})$. This by the same condition implies that $_1(v_0; s_{1::k \; 2}) \quad _{(F_1;F_2)} \quad _2(w_0; s_{1::k \; 2})$. Again by applying the same condition $k \quad 3$ more times, we conclude that $v_0 \quad _{(F_1;F_2)} \; w_0$. And, this contradicts that $F_1 \quad F_2$.

For the later case, given that $c_1(\; _1(v_0; s)) \; \not\in \; c_2(\; _2(w_0; s))$, by the first condition of Definition 10 it holds that $_1(v_0; s) \quad _{(F_1;F_2)} \quad _2(w_0; s)$. This by the second condition of the same definition implies that $_1(v_0; s_{1::jsj \; 1}) \quad _{(F_1;F_2)} \quad _2(w_0; s_{1::jsj \; 1})$. Again by applying the second condition, we concludes that $_1(v_0; s_{1::jsj \; 2}) \quad _{(F_1;F_2)} \quad _2(w_0; s_{1::jsj \; 2})$, $_1(v_0; s_{1::jsj \; 3}) \quad _{(F_1;F_2)} \quad _2(w_0; s_{1::jsj \; 3})$, ..., and finally $v_0 \quad _{(F_1;F_2)} \; w_0$, which contradicts that $F_1 \quad F_2$.

We now prove that if $F_1 \overset{L(F_1)}{=\!=\!=} F_2$ then $F_1 \quad F_2$. For the sake of contradiction assume that $F_1 \overset{L(F_1)}{=\!=\!=} F_2$ but $F_1 \quad _{(F_1;F_2)} \; F_2$, that is, $v_0 \quad _{(F_1;F_2)} \; w_0$. It is easy to observe that since $v_0 \quad _{(F_1;F_2)} \; w_0$, there must exist an observation sequence $s \; 2 \; L(v_0) \; \backslash \; L(w_0)$ such that $_1(v_0; s) \quad _{(F_1;F_2)} \quad _2(w_0; s)$. This by Definition 10 means that either (1) $c_1(\; _1(v_0; s)) \; \not\in \; c_2(\; _2(w_0; s))$ or (2) for an observation $y \; 2 \; Y$, $_1(\; _1(v_0; s); y) \; \not\in \; ?$ while $_2(\; _2(w_0; s); y) = ?$, which means that $sy \; 2 \; L(F_1)$ while $sy \; \not\in \; L(F_2)$. If any of these two cases holds, then it means that the statement $F_1 \overset{L(F_1)}{=\!=\!=} F_2$ does not hold, which is a contradiction. ■

To see an impact of Theorem 5, we first consider the following problems.

Problem: Minimal simulation filter (MSF)
Input: A filter F.
Output: A filter $F \quad$ such that $F \quad F \quad$ and the number of states in $F \quad$ is minimum.

Problem: Minimal partitioning simulation filter (MPSF)
Input: A filter F.
Output: A filter $F \quad$ such that $F \quad F \; , F \; \$ \; F \quad$ and the number of states in $F \quad$ is minimum.

Next we establish the following result.

Corollary 2: Given a filter F, any optimal (feasible) solution to FM with input F is an optimal (feasible) solution to MSF with the same input, and vice versa. Also, any optimal (feasible) solution to FPM with input F is an optimal (feasible) solution to MPSF with the same input, and vice versa.

Proof: The result follows from Theorem 5 and the definitions of the filter minimization problem and the filter partitioning minimization problem. ■

As a result, the filter minimization problem can be thought of finding a minimal filter simulating the original filter.

In the context of transition systems, the problem of computing a minimal transition system that simulates a given transition system is a trivial problem since transition systems are, in general, "nondeterministic" and, therefore, the minimal solution has always jCj states—one state for each "color"—between any pair of which there is a transition labeled y for any "nondeterministic" (action) y. That approach, however, cannot be used here since combinatorial filters are "deterministic" while the constructed structure (the

Algorithm 1: UNIONOFALLCOMPRELATIONS

Input : A filter $F = (V; Y; C; \; ; c; v_0)$
Output: The union of all compatibility relations for $(F_1; F_2)$

```
1  R ← ;
2  forall (v; w) ∈ V × V do
3  |  if c(v) ≠ c(w) then
4  |  |  continue
5  |  add ← true
6  |  forall y ∈ Y do
7  |  |  if (v; y) ∉ ? and (w; y) ∉ ? then
8  |  |  |  if c(δ(v; y)) ≠ c(δ(w; y)) then
9  |  |  |  |  add ← false
10 |  if add = true then
11 |  |  R ← R ∪ {(v; w)}
12 updated ← true
13 while updated = true do
14 |  updated ← false
15 |  forall (v; w) ∈ R do
16 |  |  forall y ∈ Y do
17 |  |  |  if (v; y) ∉ ? and (w; y) ∉ ? then
18 |  |  |  |  if (δ(v; y); δ(w; y)) ∉ R then
19 |  |  |  |  |  R ← R − {(v; w)}
20 |  |  |  |  |  updated ← true
21 return R
```

one with $|C|$ states) is "nondeterministic"; in other words, the final structure does not fit into Definition 1.

IX. THE UNION OF ALL COMPATIBILITY RELATIONS AND THE MERGEABILITY RELATION

In this section, we introduce two special compatibility relations which are used to identify a taxonomy of filters that are minimizable in polynomial time.

By the discussions of the Section IV, to solve FPM for given filter F —one without any unreachable states, of course— one can make the quotient filter under some compatibility equivalence relation, and to solve FM for F, one need to make a quotient filter under a closed covering of the state space of F. Section VI proved that the bisimilarity relation $\sim_F$ is not always the appropriate relation for this job, and Section VIII showed that any minimal filter equivalent to the original filter is always a minimal filter that simulates the original filter.

Another intuitive possibility would be to use the union of all compatibility relations, analogous to the definition of the bisimilarity relation as the union of all bisimulation relations. As an example, this relation for filter F_3 depicted in Figure 5, is $I_{F_3} \cup \{(v_0; v_1); (v_1; v_0); (v_2; v_3); (v_3; v_2); (v_4; v_5); (v_5; v_4)\}$. For a given filter F, we write f_F to denote this the union of all compatibility relations for F. Trivially, f_F is itself a compatibility relation for F.

In addition, we can compute f_F in time polynomial in the size of F. See Algorithm 1 for a simple approach to doing

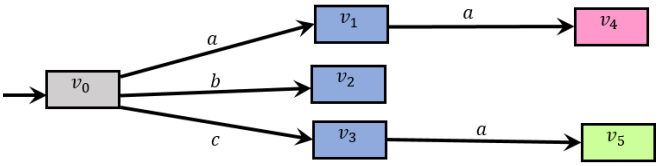

Fig. 13. A filter for which the union of all compatibility relations is not an equivalence relation. State v_0 has color 1, states in the middle column have color 2, state v_4 has color 3, and state v_5 has color 4.

so. The intuition is to begin with a relation containing state pairs that are compatible for observation strings of length at most one, and then to iteratively delete state pairs that violate Definition 6 for successively longer strings. The time complexity of this algorithm is $O(|V|^4|Y|)$, where V and Y are, respectively, the state space and the observation space of the input filter.

The next lemma shows that, unfortunately, f_F may not be suitable for solving FPM, because for some filters it is not an equivalence relation. (Recall Definition 7, under which quotient filters are well-defined only for compatibility equivalence relations.)

Lemma 8: For any filter $F = (V; Y; C; \; ; c; v)$, the relation f_F is reflexive and symmetric. However, there exist filters F for which f_F is not transitive.

Proof: For the first claim, consider that in sense of Definition 6, the identity relation $I_F = \{(v; v) \mid v \in V\}$ is a compatibility relation for F. By definition of f_F, it is a superset of I_F, and therefore f_F is reflexive. To prove that f_F is symmetric, one need to show that if $v \; \mathsf{f}_F \; w$, then $w \; \mathsf{f}_F \; v$. Suppose that $v \; \mathsf{f}_F \; w$. This means that there exists a compatibility relation R for F such that $(v; w) \in R$. By the symmetry of conditions (1) and (2) of Definition 6 with respect to v and w, if R is a compatibility relation for F, then so is R^{-1}. The relation R^{-1} contains $(w; v)$, and so does f_F given the definition of f_F.

For the second claim, to observe that f_F may not be transitive, let F be the filter depicted in Figure 13. For this filter, we have $\mathsf{f}_F = I_F \cup \{(v_1; v_2); (v_2; v_1); (v_2; v_3); (v_3; v_2)\}$. This relation is not transitive since $v_1 \; \mathsf{f}_F \; v_2$ and $v_2 \; \mathsf{f}_F \; v_3$ hold but $v_1 \; \mathsf{f}_F \; v_3$ does not. ∎

In an important sense, Lemma 8 should not be a surprise. Since the filter partitioning minimization problem is NP-hard [36], [47], and F/f_F can be computed in polynomial time, if Lemma 8 were false, that would imply that $P = NP$.

However, any filter F with state space V for which f_F is indeed an equivalence relation, then F/f_F is guaranteed to be an optimal solution to FPM with input F, and because in this case f_F is a minimal closed covering of V, we conclude that F/f_F is an optimal solution for FM too. Moreover, given any filter F, it takes polynomial time to check whether f_F is an equivalence relation or not. This implies that solving FM and FPM for any filter for which f_F is an equivalence relation takes polynomial time in size of F. This fact gives a roadmap to recognize some classes of filters for which the filter minimization problem and the filter partitioning minimization problem are in P, specifically

by looking for classes of filters for which the union of all compatibility relations can be proven to be an equivalence relation. Section X uses this fact to identify a several of these kind of classes.

A question here is does there exist any other efficiently computable relation other than the union of all compatibility relations, that if it has a special property, then FM or FPM for the given input filter is solvable in polynomial time.

In the sequel, we answer that question, but first consider the following definition.

Definition 11: *Let v and w be two states in a filter F. We say that v is* mergeable *with w if there exists a compatibility equivalence relation for F such that $(v; w)\ 2\ R$, and we say that v is* not mergeable *with w otherwise.*

The idea of this definition is that by Definition 7, making quotient filters is well-defined only under compatibility equivalence relations. Accordingly, if v and w cannot be related by any compatibility equivalence relation F, then they are not collapsed into a single state in any correctly reduced filter.

Clearly, any pair of mergeable states are compatible, but the reverse does not necessarily hold. For an example of a filter in which a pair of compatible states are not mergeable, consider Figure 14. The union of all compatibility equivalence relations for the filter F_8 in this figure is $\mathsf{f}_{F_8} = I_{F_8}\ [\ f(2;3);(3;2);(0;1);(1;0);(3;4);(4;3)g$, which has been depicted in part (b) of that figure by the graph CG_{F_8}. Notice that the mentioned graph does not need to have loops and it does not need to be a directed graph given that we implicitly know that the union of all compatibility relations for any filter is reflexive and symmetric. Although states 2 and 3 in this filter are compatible, we argue that they are not mergeable. To do so, we need to show that no compatibility relation containing $(2;3)$ is an equivalence relation. Let R be any compatibility relation that contains $(2;3)$. Due to the assumption that R is a compatibility relation and contains $(2;3)$, by the second condition of Definition 6, relation R must contain $(0;1)$. Given the same condition and that R contains $(0;1)$, it must contain $(3;4)$ too. But, relation R cannot be transitive since it has $(2;3)$ and $(3;4)$ but cannot contain $(2;4)$ duo to that $2\ 6\mathsf{f}_{F_8}4$. Thus, R is not an equivalence relation.

This example shows that in the quest for an optimal solution to FPM via a compatibility equivalence relation with minimum number of partitions we can immediately throw away unmergeable pairs since we know that such pairs cannot be related by any compatibility equivalence relation. As a result, to find a compatibility equivalence relation with minimum number of partitions, we can search among subsets of a relation stronger than f_F. That relation is defined as follows:

Definition 12: *The* mergeability relation *for filter F, denoted $./_F$, is the union of all compatibility equivalence relations for F.*

Equivalently, relation $./_F$ contains all state pairs $(v; w)$ such that v is mergeable with w. Given this, v is mergeable with w in the sense of Definition 11 if and only if $v\ ./_F\ w$.

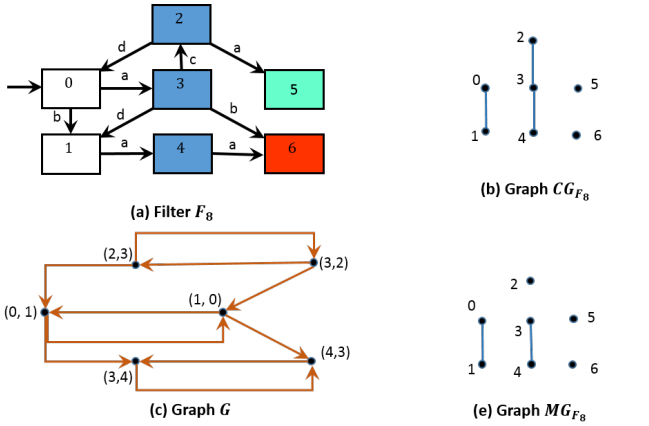

Fig. 14. **a)** A filter for which the union of all compatibility relations does not coincide with its mergeability relation **b)** The graph of the union of all compatibility relations for filter F_8 **c)** The helper graph created for filter F_8 by Algorithm 2. **d)** The graph of the mergeability relation for filter F_8. In this example, states 2 and 3 are compatible, but they are not mergeable. States 0 and 1 have color white. States in the middle column have color blue. State 5 has color light-green, and state 6 has color red.

A point worth mentioning about Definition 12 is that $./_F$ is a superset of what we are looking for—a compatibility equivalence relation with minimum number of partitions—to solve FPM. Moreover, relation $./_F$ is a compatibility relation given that it is the union of some compatibility relations. Unfortunately, for some filters, it is not an equivalence relation, and hence, may not be always used for making quotient filters.

Lemma 9: For any given filter F, relation $./_F$ is reflexive and symmetric. However, there exist filters F for which $./_F$ is not transitive.

Proof: For reflexivity, observe that for any filter F, relation $I_F = f(v; v)\ j\ v\ 2\ V g$ is an equivalence relation and is also a compatibility relation for F in the sense of Definition 6. Thus, by definition of $./_F$, it holds that $I_F\ \ ./_F$. For symmetricity, observe that by the definition of $./_F$, for any v and w such that $v\ ./_F\ w$, there exists a compatibility equivalence relation R for F such that $(v; w)\ 2\ R$. Given that R is an equivalence relation, we have that $(w; v)\ 2\ R$. Now, since $R\ \ ./_F$, it holds that $w\ ./_F\ v$.

With respect to the second claim, observe that the mergeability relation of the filter F in Figure 13 is $./_F = I_F\ [\ f(v_1; v_2);(v_2; v_1);(v_2; v_3);(v_3; v_2)g$, which is clearly not transitive. ■

An important point about the mergeability relation is that if for a filter F the relation $./_F$ is an equivalence relation, then $F = ./_F$ is a minimal filter that is equivalent to F and also partitions the state space of F. This is due to the fact that in this case, the relation $./_F$ is the coarsest compatibility equivalence relation for F. More importantly, there exist filters F for which f_F is not an equivalence relation while $./_F$ is an equivalence relation. An example of this kind of filters is filter F_8, depicted in Figure 14.

A question remaining here is whether we can compute the mergeability relation, which is the union of all compatibility

equivalence relations, in time polynomial in the size of the filter. The question is relevant because if the answer is yes, then the filter partitioning minimization problem is solvable in polynomial time for the class of filters F for which $./_F$ is an equivalence relation. Note that an algorithm that constructs the mergeability relation by constructing and unioning all compatibility equivalence relations seems likely to require superpolynomial time, since otherwise one could solve FPM in polynomial time merely by making the quotient operation under one of those constructed compatibility equivalence relations—one with the minimum number of equivalence classes.

We answer this question in positive by providing Algorithm 2. This algorithm computes the mergeability relation by removing all unmergeable pairs from f_F, i.e., it sets $./_F = \mathsf{f}_F \quad N_F$, where N_F is the set all unmergeable pairs that are compatible.

To compute N_F, we check any pair of compatible states $(v;w)$—only those in which $v \not\equiv w$, of course—to see if they can be related by a compatibility equivalence relation for F or not. Specifically, we try to construct a compatibility equivalence relation with minimum number of members that contains $(v;w)$. If we fail in constructing such a relation, then it means that $v \not\equiv_F w$, and thus, we put $(v;w)$ in N_F. Lines 2-12 of this algorithm construct as a helper, a directed graph $G = (V';E')$, used for the construction of those compatibility equivalence relations. Each vertex of this graph is a pair of states $(v;w) \in V^2$ such that $v \; \mathsf{f}_F \; w$ and $v \not\equiv w$. Each edge $((v;w);(x;z))$ of this graph indicates that any relation R containing $(v;w)$ needs to contain $(x;z)$ too in order to be a symmetric compatibility relation. Lines 6-8 of the algorithm add those edges that enforce the second condition of compatibility relation (See Definition 6). Lines 9-11 add edges enforcing the symmetricity of the constructed relation. The reader can check why for no $v \in V$ we need to add a vertex $(v;v)$ to that graph, and that why the enforcement of condition 2 of Definition 6 is made only between nodes $(v;w)$ and $(x;z)$ where $(v;w) \not\equiv (x;z)$.

To illustrate this kind of graph, see filter F_8 in Figure 14, the helper graph for which has been drawn in part c of the same figure.

Lines 17-24 of this algorithm use this graph to check for each pair of distinct compatible states $(v;w)$ if v is mergeable with w or not. This is done by constructing relation $R_{v;w}$, whose initial value is the union set of I and $DFS(G;(v;w))$, where the procedure call $DFS(G;(v;w))$ performs the standard depth first search algorithm and returns as a relation, the union of all vertices in G that are reachable from vertex $(v;w)$. Notice that the procedure call $DFS(G;(v;w))$ returns all tuples $(x;z) \in V'$ that must be contained in any symmetric compatibility relation that relates v to w. Lines 19-24 of this algorithm check if $R_{v;w}$ is an equivalence relation or not, and if not, check if it can be extended to an equivalence relation being a compatibility relation too. If the while loop broke in line 22, it means that we could not add a tuple $(a;c)$ to $R_{v;w}$ to resolve the intransitivity of $(a;b)$ and $(b;c)$ and thus $R_{v;w}$ cannot be a compatibility equivalence relation. But, if the while loop did not break in that line, it means that when the while loop is exited, relation $R_{v;w}$, which by that time is transitive, is the smallest compatibility equivalence relation that relates v to w. At line 25, the relation $./_F$ is computed by performing a set subtraction.

Algorithm 2: MERGEABILITYRELATION

Input : A filter $F = (V;Y;C;\ ;c;v_0)$
Output: The mergeability relation for F

```
1  f    UNIONOFALLCOMPRELATIONS(F)
2  V' ;; E' ;
3  forall (v;w) ∈ f do
4  |  if v ≢ w then
5  |  |  V' V' ∪ {(v;w)}
6  forall distinct vertices (v;w);(x;z) ∈ V' do
7  |  if (v;y) = x and (w;y) = z for a y ∈ Y then
8  |  |  E' E' ∪ {((v;w);(x;z))}
9  forall (v;w) ∈ V' do
10 |  if ((v;w);(w;v)) ∈ E' then
11 |  |  E' E' ∪ {((v;w);(w;v))}
12 Create a helper graph G := (V';E')
13 I ;
14 forall v ∈ V do
15 |  I I ∪ {(v;v)}
16 N ;
17 forall (v;w) ∈ V' do
18 |  R_{v;w} DFS(G;(v;w)) ∪ I
19 |  while R_{v;w} contains some tuples (a;b) and (b;c)
   |   such that (a;c) ∉ R_{v;w} do
20 |  |  if (a;c) ∈ f then
21 |  |  |  N N ∪ {(v;w)}
22 |  |  |  break
23 |  |  else
24 |  |  |  R_{v;w} R_{v;w} ∪ DFS(G;(a;c))
25 ./ f N
26 return ./
```

The time complexity of this algorithm is $O(|V|^4|Y| + |V|^8)$. Notice that computing f and G each takes $O(|V|^4|Y|)$. At any time of the algorithm, relation $R_{v;w}$ has a size of $O(|V|^2)$. The for-loop in line 17 is performed $O(|V|^2)$, in each iteration of which it takes $O(|V|^2)$ to compute $R_{v;w}$ in line 18 and it takes $O(|V|^6)$ time to run lines 19-24. Note that Algorithm 2 is presented in a simple form for clarity. The time complexity can be improved by using some simple optimizations. For example, if the algorithm did not break in line 22, then we can mark all pairs in $R_{v;w}$ as mergeable and we do not need to do the while-loop of line 17 on them, and if the algorithm did break in line 22, then we mark all pairs in $R_{v;w}$ as unmergeable and again we do not run the while-loop in line 17 for any pair in $R_{v;w}$.

Notice that for filters F for which f_F is an equivalence relation, it holds that $./_F = \mathsf{f}_F$. This is because for this kind of filters, we have that (1) $\mathsf{f}_F \quad ./_F$, which is due to the definition of $./_F$ and that f_F is a compatibility equivalence

relation for F, and (2) $./_F \quad \mathsf{f}_F$, which holds for any filter given the definition of f_F and that $./_F$ is a compatibility relation.

In the next section, we identify several classes of filters for which filter minimization or filter partitioning minimization is solvable in polynomial time.

X. Special class of filters

In this section, we identify several classes of filters for which FM or FPM can be solved in polynomial time. To do so, we identify where the union of all compatibility relations or the mergeability relation becomes an equivalence relation.

One such special class of filters consists of filters is one we call *observation-at-most-once-in-a-color* filters. An observation-at-most-once-in-a-color filter is a filter in which for any observation, from all states with the same color, there is at most one outgoing edge labeled by that observation. Such filters are a generalization of the class that Saberifar *et al.* [47] called *once-appearing-observations* filters. The difference between once-appearing-observations and observation-at-most-once-in-a-color filter is that in the former each observation appears only once while in the latter an observation can appear more than one time in the filter, but only once from the states of each color. This kind of filters might arise in applications where the observations are produced by distinct and identifiable sensors, and for each group of "related" situations or locations, a distinct sensory data is observable from at most a single situation or location of that group. In particular, this can happen in environments similar to those in Figure 4 but with different shapes. The following theorem proves that solving the filter minimization problem and the filter partitioning minimization problem for this class takes polynomial time in size of the input filter.

Problem: Observation-at-most-once-in-a-color Filter Minimization (OBS-AT-MOST-ONCE-IN-A-COL-FM)

Input: An observation-at-most-once-in-a-color filter F.

Output: A filter F such that $F \overset{L(F)}{=\!=} F$, and the number of states in F is minimal.

Theorem 6: OBS-AT-MOST-ONCE-IN-A-COL-FM 2 P.

Proof: According to the discussion above, if we prove that for any observation-at-most-once-in-a-color filter $F = (V; Y; C; \ ; c; v_0)$ the relation f_F is an equivalence relation, then the proof is complete. It is easy to observe that since in F no distinct states with the same color shares an outgoing edge labeled with the same observation, we have $\mathsf{f}_F = f(v; w) \ j \ c(v) = c(w)g$. This relation is clearly an equivalence relation. ■

A similar result holds for filter partitioning minimization of the same family of filters.

Problem: Observation-at-most-once-in-a-color Filter Partitioning Minimization (OBS-AT-MOST-ONCE-IN-A-COL-FPM)

Input: An observation-at-most-once-in-a-color filter F.

Output: A filter F such that $F \overset{L(F)}{=\!=} F$, $F \$ F$, and the number of states in F is minimal.

Theorem 7: OBS-AT-MOST-ONCE-IN-A-COL-FPM 2 P.

Proof: It follows from the fact that FM and FPM share the same optimal solution for a filter for which the union of all compatibility relations is an equivalence relation. ■

The second class consists of filters which we call *at-most-two-comp-states-in-a-col*—a filter that for each color, at most two states with that color are compatible. An example of filters that fall into this class are the class of filters in which for each color, at most two states with that color exist.

Problem: At-most-two-comp-states-in-a-col Filter Minimization (AT-MOST-TWO-COMP-STATES-IN-A-COL-FM)

Input: An at-most-two-comp-states-in-a-col filter F.

Output: A filter F such that $F \overset{L(F)}{=\!=} F$, and the number of states in F is minimal.

Theorem 8: AT-MOST-TWO-COMP-STATES-IN-A-COL-FM 2 P.

Proof: It follows from the fact that the union of all compatibility relations for any at-most-two-comp-states-in-a-col filter F is always transitive, and that for any filter F, that relation is reflexive and symmetric by Lemma 8. ■

The same result holds for filter partitioning minimization of the same family of filters.

Problem: At-most-two-comp-states-in-a-col Filter Partitioning Minimization (AT-MOST-TWO-COMP-STATES-IN-A-COL-FPM)

Input: An at-most-two-comp-states-in-a-col filter F.

Output: A filter F such that $F \overset{L(F)}{=\!=} F$, $F \$ F$, and the number of states in F is minimal.

Theorem 9: AT-MOST-TWO-COMP-STATES-IN-A-COL-FPM 2 P

A similar class consists of filters which we call *at-most-two-merg-states-in-a-col*—a filter that for each color, at most two states with that color are mergeable. Note that this class is different than at-most-two-comp-states-in-a-col filters because a filter that has more than two compatible states per color may not be a at-most-two-comp-states-in-a-col, but it could be an at-most-two-merg-states-in-a-col filter.

Problem: At-most-two-merg-states-in-a-col Filter Partitioning Minimization (AT-MOST-TWO-MERG-STATES-IN-A-COL-FPM)

Input: An at-most-two-merg-states-in-a-col filter F.

Output: A filter F such that $F \overset{L(F)}{=\!=} F$, $F \$ F$, and the number of states in F is minimal.

Theorem 10: AT-MOST-TWO-MERG-STATES-IN-A-COL-FPM 2 P.

Proof: It follows from the fact that the mergeability relation for any at-most-two-merg-states-in-a-col filter F is always transitive, and that for any filter F, that relation is reflexive and symmetric by Lemma 9. ■

At-most-two-comp-states-in-a-col and at-most-two-merg-states-in-a-col filters arise, for example, in applications where the task of interest might need a filter with a high number of colors (outputs) and each color is assigned to only a few states. This usually happens in localization and state estimate settings, where each state of the filter represents en estimation of the robot's position and only those states who estimate close positions are assigned the same color.

Another class consists of filters which we call *mergeability-is-bisimilarity*— a filter for which the mergeability relation coincides with the bisimilarity relation.

Problem: Mergeability-is-bisimilarity Filter Partitioning Minimization (MERG-IS-BISIM-FPM)

Input: A mergeability-is-bisimilarity filter F.

Output: A filter F such that $F \overset{L(F)}{=\!=} F$, $F \$ F$, and the number of states in F is minimal.

Theorem 11: MERG-IS-BISIM-FPM 2 P.

Proof: For any filter F in this class, $./_F = \ _F$. By Lemma 7, the relation $\ _F$ is an equivalence relation. ■

A subclass of class mergeability-is-bisimilarity filters consists of filters *largest-compatibility-is-bisimilarity*— a filter for which the union of all compatibility relations coincides with the bisimilarity relation. In fact, this subclass consists of filters F for which, $./_F = \mathsf{f}_F = \ _F$.

Both FM and FPM for this class of filters are solvable in time polynomial to the size of the filter.

These two kinds of filters, mergeability-is-bisimilarity and largest-compatibility-is-bisimilarity filters, arise in applications where the only feature of the system that can help reduce the size of the filter is due to some symmetry or indistinguishability in the environment or the underlying problem. This usually happens in tracking problems.

A subclass of largest-compatibility-is-bisimilarity filters are filters Saberifar *et al.* [47] called *no-missing-edges*– filters for which, from any state, for any observation, there is an outgoing edge labeled by that observation. This kind of filters can be generalized to a class we call *color-no-missing-edges* filters. A filter is color-no-missing-edges if for any two states v and w for which $c(v) = c(w)$, for any observation y, if $(v;y) \not\in ?$ then $(w;y) \not\in ?$. An application in which color-no-missing-edges arise is navigation in grid environments, where from all cells of a row or column, the same set of observations are observable, and from each of them, the same set of actions are available.

Problem: Color-no-missing-edges Filter Minimization (COL-NO-MIS-EDG-FM)

Input: A color-no-missing-edges filter F.

Output: A filter F such that $F \overset{L(F)}{=\!=} F$, and the number of states in F is minimal.

Theorem 12: COL-NO-MIS-EDG-FM 2 P.

Proof: By the definition of color-no-missing-edges for each observation y, and for any two states v and w that share the same color, we have that either $(v;y) = (w;y) = ?$ or $(\ (v;y) \not\in ?) \wedge (\ (w;y) \not\in ?)$. In this case, the three conditions of Definition 9, taking $F_1 = F_2 = F$, are identical to the conditions of Definition 6. Therefore, $\mathsf{f}_F = \ _F$, which by Lemma 7, relation f_F is an equivalence relation. ■

A similar result holds for filter partitioning minimization of that kind of filters.

Problem: Color-no-missing-edges Filter Partitioning Minimization (COL-NO-MIS-EDG-FPM)

Input: A color-no-missing-edges filter F.

Output: A filter F such that $F \overset{L(F)}{=\!=} F$, $F \$ F$, and the number of states in F is minimal.

Theorem 13: COL-NO-MIS-EDG-FPM 2 P.

XI. Conclusion

In this paper, we showed that the bisimulation quotient, which is widely used for reducing the size of transition systems, is not always appropriate for optimally reducing the size of combinatorial filters. However, we also showed that it is useful when one needs to prevent expansion of the language of a filter under minimization. We conclude that both filter minimization and filter partitioning minimization problems can be done by constructing a quotient filter, but for filter partitioning minimization we need to look for an equivalence relation while for filter minimization we need to look for a covering. While any feasible solution to FPM is a feasible solution to FM, a feasible or even an optimal solution to FPM may not be a good feasible solution to FM.

If the union of all compatibility relations or the mergeability relation for a filter is an equivalence relation, then one can optimally reduce the size of that filter. By way of example, we identified several classes of filters for which this is the case.

Knowing that making the quotient of a filter under a compatibility equivalence relation (closed covering) with minimum number of classes produces an optimal solution to FPM (FM), future work might consider the design of efficient heuristic algorithms for finding a such relation (covering). It is also interesting to attempt to identify practical filters for which finding a compatibility equivalence relation with minimum number of classes can be done in polynomial time.

Finally, the filter partitioning minimization problem for filters for which the helper graph in Algorithm 2 has no edges is reduced to the problem of *vertex clique partitioning* for the graph of the mergeability relation, and several classes of graphs for which the vertex clique partitioning are in P have been recognized. This approach may provide a roadmap for finding additional classes of filters that can be minimized in polynomial time. In addition, given that previous works do not consider computing feasible solutions to FM and they only attempt to make feasible solutions to FPM by heuristic algorithms and integer linear programming formulations, future work may consider designing heuristic algorithms for FM.

There are several related problems that remain open. Consider that any optimal solution to FPM can be obtained

by finding within the mergeability relation, a compatibility equivalence relation that has a minimum number of equivalence classes as a subset of the mergeability relation. Also, one can show, by example, that there exist filters for which not all pairs related by the mergeability relation are related by a compatibility equivalence relation with minimum number of classes. Accordingly, there exists a relation that relates only those pairs that are related by a compatibility equivalence relation with minimum number of equivalence classes. This relation is a subset of the mergeability relation and one can always find an optimal solution to FPM, by searching for a compatibility equivalence relation within this relation. A question that remains open is that whether that relation can be computed in time polynomial in the size of the input filter or not. Another problem that remains open is the problem of whether a similar relation containing only pairs of states that are within a compatibility class of a closed covering with minimum number of compatibility classes can be computed in polynomial time. Such a relation can be used to search within it for a closed covering with minimum number of compatibility classes to make an optimal solution to FM. In addition, it is not known whether, for each of those two relations, the conditions that assure a pair is related by that relation can be posed in a natural way as conditions to define a variant of the notion of compatibility relation whose definition should be similar in nature to Definition 6.

Note that both closed coverings and compatibility equivalence relations are, in fact, sets of compatibility classes in which all the states within each class are compatible with each other. It is not known under what conditions a compatibility class is within a closed covering (a compatibility equivalence relation) that yields an optimal solution to FM (FPM).

Another interesting open problem is to establish more general conditions that determine for which classes of filters FM can be solved in polynomial time. The approach utilized in Section X to identify some such classes was based on analyzing where the union of all compatibility relations becomes an equivalence relation. Another approach, however, would be to impose conditions that guarantee a filter has only a polynomial number of distinct closed coverings. Under such conditions, FM could be solved in polynomial time by enumerating the closed coverings. Additionally, notice that for each of those filters in Section X for which FM is solvable in polynomial time, there is a single unique closed covering with minimum number of classes. Accordingly, another interesting line of inquiry would be to find classes of filters that have more than one closed covering with minimum number of classes, but still one such closed covering can be constructed in polynomial time. Similar questions may be posed for FPM, considering the mergeability relation instead of the union all compatibility relations and also considering compatibility equivalence relations instead of closed coverings.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1526862.

REFERENCES

[1] P. A. Abdulla, J. Högberg, and L. Kaati, "Bisimulation minimization of tree automata," *International Journal of Foundations of Computer Science*, vol. 18, no. 04, pp. 699–713, 2007.

[2] T. Alam, L. Bobadilla, and D. A. Shell, "Space-efficient filters for mobile robot localization from discrete limit cycles," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 257–264, 2018.

[3] C. Baier, H. Hermanns, J.-P. Katoen, and V. Wolf, "Bisimulation and simulation relations for markov chains," *Electronic Notes in Theoretical Computer Science*, vol. 162, pp. 73–78, 2006.

[4] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of model checking*. MIT press, 2008.

[5] P. Beeson, M. MacMahon, J. Modayil, A. Murarka, B. Kuipers, and B. Stankiewicz, "Integrating multiple representations of spatial knowledge for mapping, navigation, and communication." in *Interaction Challenges for Intelligent Assistants*, 2007, pp. 1–9.

[6] P. Beeson, J. Modayil, and B. Kuipers, "Factoring the mapping problem: Mobile robot map-building in the hybrid spatial semantic hierarchy," *The International Journal of Robotics Research*, vol. 29, no. 4, pp. 428–459, 2010.

[7] L. Bobadilla, O. Sanchez, J. Czarnowski, and S. M. LaValle, "Minimalist multiple target tracking using directional sensor beams," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3101–3107.

[8] D. Bustan and O. Grumberg, "Simulation-based minimization," *ACM Transactions on Computational Logic (TOCL)*, vol. 4, no. 2, pp. 181–206, 2003.

[9] R. Cleaveland, J. Parrow, and B. Steffen, "The concurrency workbench: A semantics-based tool for the verification of concurrent systems," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 15, no. 1, pp. 36–72, 1993.

[10] R. Cleaveland and O. Sokolsky, "Equivalence and preorder checking for finite-state systems," in *Handbook of Process Algebra*, 12 2001, pp. 391–424.

[11] R. De Nicola, "Behavioral equivalences," *Encyclopedia of Parallel Computing*, pp. 120–127, 2011.

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

[13] L. H. Erickson, J. Yu, Y. Huang, and S. M. LaValle, "Counting moving bodies using sparse sensor beams," in *Algorithmic Foundations of Robotics X*. Springer, 2013, pp. 427–442.

[14] M. P. Fiore, "A coinduction principle for recursive data types based on bisimulation," *Information and Computation*, vol. 127, no. 2, pp. 186–198, 1996.

[15] M. Forti and F. Honsell, "Set theory with free construction principles," *Annali della Scuola Normale Superiore di Pisa-Classe di Scienze*, vol. 10, no. 3, pp. 493–522, 1983.

[16] R. Gentilini, C. Piazza, and A. Policriti, "From bisimulation to simulation: Coarsest partition problems," *Journal of Automated Reasoning*, vol. 31, no. 1, pp. 73–103, 2003.

[17] R. Givan, T. Dean, and M. Greig, "Equivalence notions and model minimization in markov decision processes," *Artificial Intelligence*, vol. 147, no. 1-2, pp. 163–223, 2003.

[18] K. Y. Goldberg, "Orienting polygonal parts without sensors," *Algorithmica*, vol. 10, no. 2, pp. 201–225, 1993.

[19] R. Gorrieri and C. Versari, "Transition systems and behavioral equivalences," in *Introduction to Concurrency Theory*. Springer, 2015, pp. 21–79.

[20] E. Haghverdi, P. Tabuada, and G. J. Pappas, "Bisimulation relations for dynamical, control, and hybrid systems," *Theoretical Computer Science*, vol. 342, no. 2-3, pp. 229–261, 2005.

[21] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke, "Computing simulations on finite and infinite graphs," in *Proceedings of IEEE 36th Annual Foundations of Computer Science*. IEEE, 1995, pp. 453–462.

[22] J. Högberg, A. Maletti, and J. May, "Backward and forward bisimulation minimization of tree automata," *Theoretical Computer Science*, vol. 410, no. 37, pp. 3539–3552, 2009.

[23] P. C. Kanellakis and S. A. Smolka, "Ccs expressions finite state processes, and three problems of equivalence," *Information and Computation*, vol. 86, no. 1, pp. 43–68, May 1990.

[24] S. M. Kristek and D. A. Shell, "Orienting deformable polygonal parts without sensors," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 973–979.