Journal of Computer and System Sciences 85 (2017) 74–92

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Combinatorial filter reduction: Special cases, approximation, and fixed-parameter tractability

Fatemeh Zahra Saberifar [a], Ali Mohades [a,*], Mohammadreza Razzazi [b], Jason M. O'Kane [c]

[a] *Department of Mathematics and Computer Science, Amirkabir University of Technology, 424 Hafez Ave, Tehran, Iran*
[b] *Computer Engineering and IT Department, Amirkabir University of Technology, 424 Hafez Ave, Tehran, Iran*
[c] *Department of Computer Science and Engineering, University of South Carolina, 315 Main St, Columbia, SC, USA*

ARTICLE INFO

Article history:
Received 25 April 2016
Received in revised form 9 November 2016
Accepted 11 November 2016
Available online 22 November 2016

Keywords:
Combinatorial filters
Reduction
Automata
Approximation
Fixed-parameter

ABSTRACT

Recent research in algorithmic robotics considers *combinatorial filters*, which concisely capture the discrete structure underlying many reasoning problems for robots. An important recent result is that the *filter minimization problem*—Given a filter, find the smallest equivalent filter—is NP-hard. This paper extends that result along several dimensions, including hardness proofs for some natural special cases and for approximation, and new results analyzing the only known algorithm for this problem. We show that this problem is not fixed-parameter tractable for any of the obvious parameters, but it is fixed-parameter tractable for a certain combination of new parameters.

1. Introduction

This paper continues the thread of research initiated by LaValle [1,2] concerning *combinatorial filtering*. The idea is to model systems (such as robots or sensor networks) that must process and draw conclusions from a sequence of discrete sensor readings.

Fig. 1 shows a well-known example. In this environment, with two moving agents and three beams sensors, the task is to determine whether both agents are in the same region at each time. A naive solution requires considering $2^9 = 512$ possible states to solve the task, but Tovar, Cohen, Bobadilla, Czarnowski and LaValle [3] described a (hand-crafted) optimal combinatorial filter with just 4 states for this problem. See Fig. 1.

Part of the motivation for studying combinatorial filters can be understood by analogy to the probabilistic filters [5] commonly used by roboticists. Though such filters must, in the general case, be able to represent arbitrarily complicated probability densities, many systems behave in sufficiently structured ways that simpler representations can be used without loss of accuracy. For example, in a linear system with Gaussian noise, the resulting density is simply a Gaussian whose mean and covariance can be computed using the Kalman filter [6]. In the same way, optimized combinatorial filters can be viewed as a specialization of the general class of nondeterministic filters [1], which reason about possible states of the underlying system. If the number of states in such a filter can be reduced without altering the filter's behavior, then the result can be useful for understanding the elements of structure underlying the problem.

* Corresponding author.
E-mail addresses: fz.saberifar@aut.ac.ir (F.Z. Saberifar), mohades@aut.ac.ir (A. Mohades), razzazi@aut.ac.ir (M. Razzazi), jokane@cse.sc.edu (J.M. O'Kane).

http://dx.doi.org/10.1016/j.jcss.2016.11.007
0022-0000/© 2016 Elsevier Inc. All rights reserved.

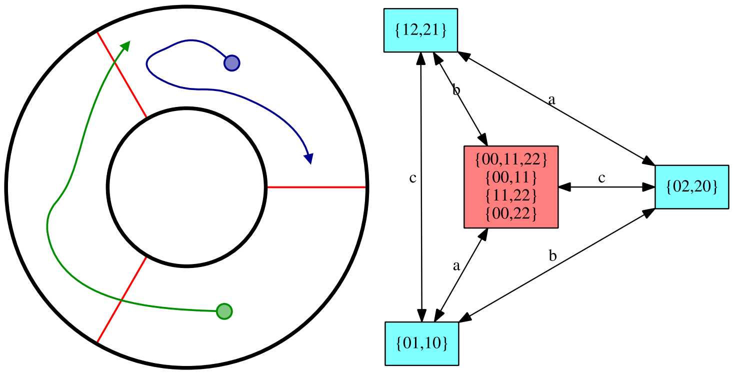

Fig. 1. (O'Kane and Shell [4]) [left] Two agents move through an environment having three beam sensors. [right] An optimal combinatorial filter reproduced by O'Kane and Shell's algorithm [4], for solving the task of whether the agents are in the same region at each time. The numbers 0, 1, and 2 denote the regions, and the letters a, b, and c denote observations of the three beam sensors. The shading of each state indicates the filter's output: red indicates that the agents are in the same region and blue indicates that the agents are not in the same region. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

A common feature of most of the early work on combinatorial filters is the need for human input to design and minimize the filters. Hence, O'Kane and Shell [4] worked on an automated method for filter reduction. Specifically, they consider the problem in which the input is a combinatorial filter expressed as a transition graph, and the goal is to produce the smallest reduced filter, measured by the number of states, equivalent to the given filter. They proved that this *filter minimization problem* is NP-hard and presented an efficient heuristic algorithm for it. This algorithm was able to reproduce the optimal filter for the problem in Fig. 1, automatically.

In spite of this progress, there remain some open problems in this area.

1. The filter minimization problem so far has been proved NP-hard only in the general case. However, this result seems very fragile, in the sense that there may exist some common special cases of it that are solvable efficiently.
2. In spite of the hardness of solving the problem optimally, there may still exist approximation algorithms, either for the general problem or for special cases. That is, perhaps we can guarantee to find, in polynomial time, solution that is within a factor α of optimal.
3. The only algorithm for this problem is presented by O'Kane and Shell. However, their analysis of the algorithm is incomplete. In particular, they conjecture, but do not prove, that if a certain graph coloring subroutine in that algorithm is performed optimally, then the resulting filter will also be optimally reduced.
4. Another way to attack this problem is to identify parameters that capture the difficulty of the problem, independent of problem size. This kind of analysis arises from the field of parameterized complexity theory. At present, nothing is known about which parameters govern the hardness of filter minimization.

The contribution of this paper is to provide some solutions for each of these four families of open problems.

1. We show that several natural special cases of the problem are also NP-hard. We also prove that a few special narrow cases belong to P. Table 1 shows a brief preview of these results.
2. We show that the filter minimization problem is NP-hard to approximate, even for the hard special cases mentioned above.
3. We show that O'Kane and Shell's conjecture—that their algorithm produces optimally reduced output filters if their graph coloring subroutine is optimal—is false, by providing a counterexample for which optimal coloring in their algorithm does not necessarily produce optimal filter reduction.
4. We show that for some natural kinds of parameters, the problem is not fixed-parameter tractable. We also introduce two new parameters and show, by describing an algorithm, that the problem is fixed-parameter tractable when it is parameterized by these two parameters.

Taken together, these results support a conclusion that filter minimization is a truly challenging problem, because it is not susceptible to any of the standard approaches for solving NP-hard problems. In practice, solving the filter minimization problem appears to require accepting at least one of three unpalatable options: (a) suboptimal heuristic solutions, (b) potentially exponential run time, or (c) restriction to a highly-constrained special case.

The balance of this paper is structured as follows. Section 2 reviews the related work and Section 3 introduces the basic definitions and the problem formulation. Section 4 presents a technique for converting the instances of the graph coloring problem to instances of the filter minimization problem, and proves several useful properties of this conversion that we reuse in some other sections. In Section 5, we prove hardness results for tree, bipartite, and planar filter reduction,

Table 1
The hardness results for several special cases of the filter minimization problem.

Problem name	Problem description	Hardness
TREE-FM	Minimization of filters with tree structure	NP-hard
BIPARTITE-FM	Minimization of filters with bipartite graph structure	NP-hard
PLANAR-FM	Minimization of filters with planar graph structure	NP-hard
NO-MISSING-EDGES-FM	Minimization of filters with no missing observations	P
ONCE-APPR-OBS-FM	Minimization of filters in which each observation appears only once	P
TWICE-APPR-OBS-FM	Minimization of filters in which each observation appears at most twice	NP-hard
m-APPR-OBS-FM	Minimization of filters in which each observation appears at most $m > 1$ times	NP-hard

and also show some hardness results for several classes of filters which have limitations on the number of repetitions of each observation in the filter. Section 6 investigates the approximation hardness of the general filter reduction problem and its NP-hard special cases. In Section 7, we analyze the optimality of O'Kane and Shell's algorithm and in Section 8, we present some fixed-parameter tractability and intractability results with the parameters of solution size, treewidth, number of appearances of each observation and two parameters of maximum weight of colors and total separation depth of input filter. Section 9 summarizes these results and plots a course of future work.

2. Related research

2.1. Combinatorial filters

Combinatorial filters were first formulated and introduced by LaValle [1,2], building upon earlier work on minimalism in robotics [7,8]. These filters are based on collapsing the information available to a robot to the smallest representation that is still sufficient to solve a task. Because the sensor data and the processing are discrete, the computation power required by a robot to execute a combinatorial filter, measured both by time and by space, is generally very small. Combinatorial filters have been designed for several tasks such as navigation [9,10], target tracking [11,12], exploration [13], planning under topological constraints [14], and story validation [15,16]. Tovar, Cohen, Bobadilla, Czarnowski, and LaValle [17,3] introduced optimal combinatorial filters for solving some inference tasks in polygonal environments with beam sensors and obstacles. Song and O'Kane [18] worked on extensions of standard combinatorial filters, which require finite information spaces, to the some limited classes of problems with infinite information spaces. As an instance of previous work which can be viewed through the lens of combinatorial filters, it can be noted that Kristek and Shell [19] showed how to extend existing methods for sensorless manipulation [7,8] to deformable objects. The filters we define in this paper are a special case of the nondeterministic and stochastic graphs explored by Erdmann [20,21] in the context of planning. Erdmann proved topological conditions for the existence of acyclic solutions in a broad class of discrete planning problems.

The first results on finding optimally reduced filters automatically—rather than by human analysis—were presented by O'Kane and Shell [4]. They proved that the filter minimization problem is NP-hard and presented a heuristic algorithm to solve it. The same authors used similar techniques to solve concise planning [22] and discreet communication [23] problems.

Finally we note that there are close connections between combinatorial filters and finite automata. For example, Wu, Raman, Lafortune, and Seshia [24] use a deterministic finite automaton model to solve a more general constrained obfuscation problem than the one considered by O'Kane and Shell [23]. We discuss the relationship between combinatorial filter reduction and finite automata in more detail in Section 5.2.1.

2.2. Probabilistic filtering

The chief alternative to combinatorial filtering, at least in the context of robotics, is the large body of work that uses probabilistic filters to estimate the robot's state [5]. In both cases, incoming sensor data are processed to form a "belief" about the state of the world. The primary difference is that combinatorial filters generally consider discrete—rather than continuous—state and observation spaces. On the probabilistic side, several different methods have been proposed to reduce the complexity of the belief spaces [25–27]. These methods can be viewed as a probabilistic analog to the combinatorial filter reduction problems addressed in this paper.

We also note that combinatorial filters are readily combined with probabilistic models. Two recent examples use this approach for target tracking applications [12,28].

3. Definitions and formulation

This section reviews basic definitions for the filter minimization problem, as introduced by O'Kane and Shell [4].

Definition 1 *([4])*. An *information state (I-state) graph* I is an edge-labeled directed graph supplemented with a starting vertex, *i.e.*, $I \triangleq (V, E, l : E \to Y, v_0)$, in which

1. the finite set V contains vertices which we call “states”,
2. the set $E \subseteq V \times V$ consists of ordered pairs of vertices termed directed edges,
3. each edge is labeled with an observation, drawn from observation space Y, via the function l, and
4. the starting I-state is identified as $v_0 \in V$.

An additional and important requirement on l is that if $e_1 \triangleq (v, v_j)$ and $e_2 \triangleq (v, v_k)$ with $v_j \neq v_k$ then $l(e_1) \neq l(e_2)$. That is, no two edges originating from the same vertex have the same label. For convenience, we write $v_1 \xrightarrow{y} v_2$ for an edge from v_1 to v_2 bearing label y.

The definition ensures that I-state graph is deterministic, in the sense that for each observation sequence, there exists at most one final state that can be reached by starting at the initial state and following edges labeled with successive observations. Note that, in contrast to a deterministic finite automaton (DFA) [29], an I-state graph need not have an out-edge for each observation. The interpretation is that such ‘missing edges’ correspond to observations that cannot, based on the structure of the system being modeled, occur at a particular state. Therefore, the final state for a given observation sequence may be undefined. The set of sequences for which the final state is well-defined forms the language induced by the I-state graph.

Definition 2 *([4]).* The *language induced by an I-state graph* I, denoted $\mathcal{L}(I) \subseteq Y^\star$, is the set of all sequences of observations (*e.g.*, $y_0 y_1 \cdots y_n$) for which valid transitions may be traced on I by starting at its initial vertex.

We are now able to define filters as the following:

Definition 3 *([4]).* An I-state graph supplemented with a coloring of its vertices is termed a *filter*, *e.g.*, $F \triangleq (V, E, l : E \to Y, v_0, c : V \to \mathbb{N}^+)$, in which function c assigns a natural number to each I-state.

The intuition is that the coloring of states in a filter represents the task that filter must complete. For example, in the system shown in Fig. 1, the task is to determine whether the two agents are in the same region (states colored red) or not (states colored blue).

In this paper, we want to reduce these kinds of filters, such that the reduced filter behaves identically to the original one, for any observation sequence in the language induced by the original filter. That is, the reduced filter must perform the same filtering, producing the same output colors at each step as the input filter. The next definition formalizes this idea.

Definition 4 *([4]).* Two filters $F_1 \triangleq (V, E, l : E \to Y, v_0, c_1)$, and $F_2 \triangleq (W, F, m : F \to Y, w_0, c_2)$ with a common observation space Y are said to be *equivalent with respect to a language* $\mathcal{L} \subseteq Y^\star$ if, for every observation sequence $l \in \mathcal{L}$, the I-states v^l and w^l reached by tracing l on F_1 and F_2 respectively are both defined, and we have $c_1(v^l) = c_2(w^l)$. We denote this equivalence relation with $F_1 \overset{\mathcal{L}}{=\!=} F_2$.

Note that this definition does not require the filters to produce identical output colors for observation sequences that are not in $\mathcal{L}$. This differs, for example, from the idea of implicitly adding a uniquely-colored ‘dead state’ reached by each missing edge, and requiring the two filters to produce identical outputs for all strings in $Y^\star$.

Our primary concern in this paper is the problem of reducing a given filter to the smallest equivalent filter. That is, given an input filter F, we need to reduce it to a minimal filter $F^\star$, such that $F^\star$ is equivalent to F with respect to the language of F. We now are able to define the *filter minimization* problem as follows.

Problem: Filter Minimization (FM)

Input: A filter F.

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

The primary result of O’Kane and Shell can now be stated concisely.

Theorem 1 *([4]).* *FM is NP-hard.*

The objective of this paper is to investigate this problem in greater detail, leveraging the usual tools for making progress on NP-hard problems.

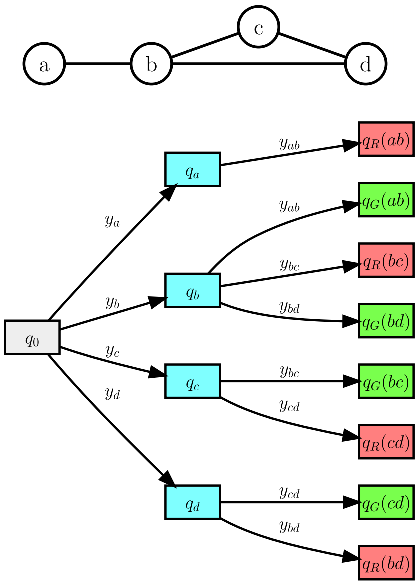

Fig. 2. [top] An instance of graph coloring problem. [bottom] The corresponding instance of FM. The states of the first layer have color 1, the middle layer has color 2, and the states in the third layer have colors 3 and 4. Note that the filter is a tree.

4. Graph coloring and filter minimization

In this section, we describe a polynomial time technique to convert an instance of the graph coloring problem to an instance of the filter minimization problem and prove several useful properties of that conversion. This algorithm, which is a modified version of O'Kane and Shell's original reduction from 3-coloring, will be reused in several proofs throughout this paper.

Consider the general graph coloring problem.

Problem: Graph Coloring (GC)

Input:	An undirected graph G with n vertices.
Output:	A coloring of G, in which no two adjacent vertices share a color, using the minimum number of colors.

Given an undirected graph $G \triangleq (V, E)$ as an instance of GC, we create an instance $F(G) \triangleq (Q, E_2, l, q_0, c)$ of FM as follows:

1. Start with an initial state q_0 in $F(G)$ with color $c(q_0) = 1$.
2. For each node v_i of G, add a state q_{v_i}. Color all of these states with color $c(q_{v_i}) = 2$. Connect the initial state q_0 to the all states q_{v_i}, and assign to each of these directed edges an observation label y_{v_i}.
3. For each edge (v_i, v_j) in G, add two states $v_{R(ij)}$ and $v_{G(ij)}$ along with two edges $q_{v_i} \xrightarrow{y_{ij}} v_{R(ij)}$ and $q_{v_j} \xrightarrow{y_{ij}} v_{G(ij)}$. The symbols R and G are meant to mnemonically indicate 'red' and 'green'. Each state $v_{R(ij)}$ is assigned color 3; each state $v_{G(ij)}$ is assigned color 4. (The ordering of v_i and v_j for each edge is arbitrary, but must be fixed.)

This construction forms a filter $F(G)$ with $1 + |V| + 2|E|$ states. See Fig. 2 for an example. Informally, we can divide the states of this filter into three 'layers': A first layer containing only the initial state (step 1 above), a middle layer containing the $|V|$ states that directly correspond to vertices of G (step 2 above), and a third layer containing the remaining $2|E|$ states (step 3 above).

The intuition is to embed the structure of G into the middle layer of $F(G)$, using the other states to force reduced filter equivalent to $F(G)$ to avoid 'merging' any middle-layer states corresponding to adjacent nodes in G. Thus, the conversion, which clearly takes time polynomial in the size of $F(G)$, derives its value from the fact that we can establish a clear connection between the number of colors needed to properly color G and the size of the smallest filter equivalent to $F(G)$.

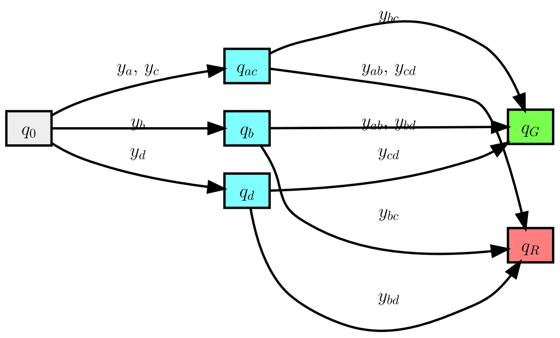

Fig. 3. An equivalent filter to the filter shown in Fig. 2. Since the original graph is 3-colorable, the filter can be reduced to the one with only six states. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

In the following, we will show that G is k-colorable if and only if there exists a reduced filter F', equivalent to $F(G)$, of at most $k+3$ states.[1]

Lemma 1. *For any instance G of* GC*, if G is k-colorable, then there exists a filter with $k+3$ states equivalent to $F(G)$.*

Proof. Suppose G is k-colorable and let c_G denote the mapping from vertices of G to colors. We construct F' from $F(G)$ via two modifications.

First, we merge, via vertex contractions, all of the $v_R(\cdot)$ states in the third layer, replacing these n states with a single state with color 3. Likewise, we merge all of the $v_G(\cdot)$ states together into a single state of color 4. Because there are no out-edges from any state in the third layer of $F(G)$, the filter resulting from this change is equivalent to $F(G)$.

Second, we merge each pair of states in the middle layer whose corresponding G vertices that are assigned the same colors by c_G. The result of these merges is our F'. Note that F' has exactly $k+3$ states: A single initial state, single red and green states in the third layer, and exactly k states in the middle layer. See Fig. 3 for an example with $k=3$.

It remains to show that F' is a valid filter. That is, we show that F' satisfies the determinacy condition in Definition 1. Suppose F' violates that condition, so that there exists a state q_x in F' with two outgoing edges that share the same label. Since neither the first layer nor the third layer experienced any change in their out-edges, this state must be within the second layer of F'. Let y_{ab} denote the observation for which q_x has two out-edges. According to the construction of F' from $F(G)$, this implies that two states q_a and q_b in $F(G)$ were merged together in F' and form q_x, which implies that q_a and q_b have the same color under c_G. However, the fact that these two states have out-edges labeled y_{ab} implies that the nodes a and b are adjacent in G. Since a and b are both adjacent and same-colored in G, we have a contradiction. Thus, F' is a valid filter.

Finally, we argue that F' is equivalent to $F(G)$ under the language induced by $F(G)$. The observation sequences in this language range in length from 0 to 2. For the observation sequence of length 0 (that is, the empty string), according to the construction, the initial states of F' and $F(G)$ both have the same output color, namely color 1. For observation sequence with length 1, in both filters there are the identical edges labeled y_{v_x} from layer 1 to layer 2 and the outputs have the same color 2. Finally, for observation sequences with length 2, there exist the same observations $y_{v_x v_y}$ from layer 2 to layer 3 and same outputs, namely color 3 or color 4. Since the filters produce the same output colors for all observation strings in the language induced by $F(G)$, the two filters are equivalent under that language. □

Lemma 2. *For any instance G of* GC*, and any positive integer k, if there exists a filter F' with $k+3$ or fewer states that is equivalent to $F(G)$, then G is k-colorable. Moreover, a k-coloring of G can be extracted from F' in polynomial time.*

Proof. Let F' denote a filter with $k+3$ states that is equivalent to $F(G)$. We prove that G can be k-colored. Note that F' must have at least one state with color 1, namely its initial state. Similarly, F' must contain at least one state with color 3 and another with color 4. Without these states, F' would be unable to produce correct outputs (that is, the same outputs as $F(G)$) for observation sequences of length 0 and 2 respectively. Accounting for these 3 states with colors other than 2, F' contains at most k states with color 2.

Observe that, to produce correct outputs for observation sequences of length 1, F' must contain, for each node v_i in G, an edge $q_0 \xrightarrow{y_{v_i}} q'$, from the initial state to some other state q' in F'. Furthermore, this q' must have color 2. Define a coloring function c_G for G in which $c_G(a) = c_G(b)$ if and only if the edges labeled with y_a and y_b reach identical states

[1] O'Kane and Shell [4] described a similar construction in which the derived filter is a directed acyclic graph, rather than a tree. They showed that the original graph can be 3-colored if and only if the constructed filter can be reduced to 6 or fewer states. We present a stronger result, based on the chromatic number of the graph rather than the constant 3.

in F'. Since F' has at most k states of color 2, this function uses at most k distinct colors. This coloring can be efficiently extracted from F' by examining each of the out-edges of the initial state.

It remains to show that this c_G is a valid coloring of G. Suppose not, and two adjacent nodes a and b in graph G have the same color under c_G. Note that in $F(G)$, the observation sequences $y_a y_{ab}$ and $y_b y_{ab}$ should reach states with colors 3 and 4, respectively. Since $F(G) \overset{\mathcal{L}(F(G))}{=\!=} F'$, these two observation sequences must also reach the colors 3 and 4 in F', respectively. However, since the transitions in F' from the initial state under y_a and y_b reach that same intermediate state, the final states cannot differ. We conclude from this contradiction that G is k-colorable. □

Lemmas 1 and 2 establish an equivalence between each graph coloring instance G and its corresponding filter reduction instance $F(G)$: The number of colors needed for the former and the number of states needed for the latter differs by 3.

5. Special cases of filter minimization

5.1. Tree, bipartite, and planar filters

Note that the construction introduced in Section 4 builds a filter that has the form of a tree rooted at the initial state. That structure, along with Lemmas 1 and 2, leads directly to hardness results for several special cases of FM. First, we consider filters that are in the form of trees, which are of particular interest in applications because any filter that executes for only a bounded number of stages can be expressed as a tree.

Problem: Tree Filter Minimization (TREE-FM)

Input: A filter F whose states and directed edges form a tree.

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

Theorem 2. TREE-FM *is NP-hard.*

Proof. Reduction from GC, which is known to be NP-hard. Suppose there exists a polynomial time algorithm A for TREE-FM. Given an instance G of GC, use the method of Section 4 to construct $F(G)$. Because this construction builds a tree, $F(G)$ is an instance of TREE-FM. Then use algorithm A to find smallest filter F' equivalent to $F(G)$. Lemma 2 implies that we can extract an optimal coloring of G from F' in polynomial time. Therefore, a polynomial time algorithm for TREE-FM would imply a polynomial time algorithm for GC. □

Similar arguments apply for other special kinds of graphs. For example, systems that have two independent sensor systems—say, a range sensor and a bump sensor—can be modeled using filters that are bipartite graphs, in which each type of sensor provides input on alternating stages.

Problem: Bipartite Filter Minimization (BIPARTITE-FM)

Input: A filter F whose states and directed edges form a bipartite graph.

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

Corollary 1. BIPARTITE-FM *is NP-hard.*

Proof. Follows from Theorem 2 and the fact that each instance of TREE-FM is an instance of BIPARTITE-FM. □

Problem: Planar Filter Minimization (PLANAR-FM)

Input: A filter F whose states and directed edges form a planar graph.

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

Corollary 2. PLANAR-FM *is NP-hard.*

Proof. Follows from Theorem 2 and the fact that each instance of TREE-FM is an instance of PLANAR-FM. □

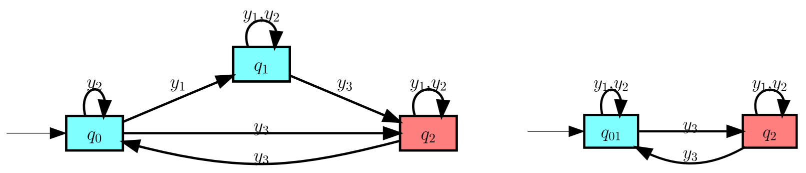

Fig. 4. [left] An example no-missing-edges filter. [right] The reduced filter of the first filter, resulting from a modified version of Hopcroft's DFA minimization algorithm.

5.2. Filters with limitations on the observations

In this subsection, we consider some special cases in which the number of appearances of each observation is constrained. Subsection 5.2.1 presents a class of filters that have at least n copies of each observation. At the other end of the spectrum, Subsections 5.2.2 and 5.2.3 introduce some kinds of filters with at most one and two copies of each observation.

5.2.1. No missing observations filters

At first glance, perhaps it might appear that FM is equivalent to the standard problem of minimizing the number of states in a deterministic finite automaton (DFA). However, though FM is NP-hard, polynomial time algorithms for DFA minimization are a well-known part of automata theory [29]. The seemingly minor difference between these two problems, namely that a filter is allowed to have 'missing' edges whereas a DFA is not, explains the difference.

To illustrate this difference, we consider the subset of filters for which no edges are missing. Such filters correspond, for example, to the situation in which the environment has no underlying structure that prevents any observation from occurring at any state.

Definition 5. A no-missing-edges filter is a filter for which every state has an out-edge for every observation.

In particular, in a no-missing-edges filter with n states, each observation appears as the label of n distinct directed edges. Fig. 4 shows an example. Note that there may be fewer than n observations.

The next lemma makes the connection between no-missing-edges filters and DFAs more precise.

Lemma 3. *If an input filter F is a no-missing-edges filter with observation space Y, then $L(F) = Y^\star$.*

Proof. Because every state has a transition for every observation, the output of the filter is well-defined for any observation sequence. □

Now, we can define and solve the problem of minimizing a no-missing-edges filter.

Problem: No-missing-edges Filter Minimization (NO-MISSING-EDGES-FM)

Input: A no-missing-edges filter F.

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

Theorem 3. NO-MISSING-EDGES-FM $\in P$.

Proof. Use an algorithm very similar to Hopcroft's DFA minimization algorithm [29], with one change: We treat a pair of states (u, v) as distinguishable in the first iteration if $c(u) \neq c(v)$, in contrast to Hopcroft's DFA minimization algorithm which checks for pairs of states of which exactly one is an accepting state. The remaining steps of the algorithm, which uses a partition refinement approach to form equivalence classes of states, remains unchanged. This algorithm takes $O(n^2 \log n)$ time. Therefore, NO-MISSING-EDGES-FM $\in P$. □

5.2.2. Filters with unique observations

In this subsection, we consider filters in which each observation is unique. That is, each observation in these filters appears as a label for just a single directed edge. Applications in which such filters might arise include those in which the observations are generated by distinct, identifiable sensors, such as the break beam sensors in Fig. 1.

Definition 6. A *once-appearing-observations filter* is a filter in which each observation appears only once.

Algorithm 1: An algorithm for ONCE-APPR-OBS-FM.

```
  Input: A filter $F \triangleq (V, E, l : E \to Y, v_0, c_1)$
1 Create one state in $F'$ for each color of input filter $F$ with that color.
2 Pick the initial state of $F$ as the initial state of $F'$.
3 for each edge $u \xrightarrow{y} v$ in $F$ do
4     Add out-edges from each state in $F'$ to the (unique) state with color $c(v)$, labeled with observation $y$.
5 end
```

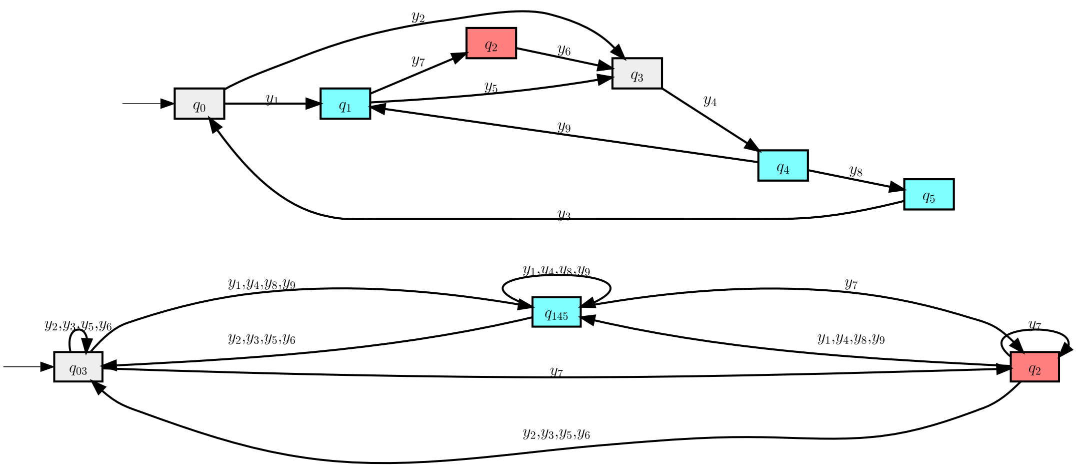

Fig. 5. [top] A once-appearing-observations filter. [bottom] Its reduced filter, computed by Algorithm 1.

Now, consider the minimization problem for such filters.

Problem: Once-appearing-observations Filter Minimization (ONCE-APPR-OBS-FM)

Input: A once-appearing-observations filter F.

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

In the following, we show that ONCE-APPR-OBS-FM can be solved in polynomial time.

Theorem 4. ONCE-APPR-OBS-FM $\in P$.

Proof. We will show that this problem is solvable in polynomial time by Algorithm 1. This algorithm produces a reduced filter whose number of states is equal to the number of colors in the input filter F. For each edge of the input filter, it creates edges from all states of F' to the state with the same color as destination of that edge in F. See Fig. 5.

Note that $L(F') = Y^\star$. We must show that $F \overset{\mathcal{L}(F)}{=\!=} F'$. We prove it by induction on the length k of the input observation sequence. When $k = 0$, only the initial states are relevant. Algorithm 1 ensures that the initial states have the same color in both F and F'. Now suppose for the sake of induction that the outputs of all input observation sequences $y_1 y_2 \ldots y_k \in L(F)$ with length k, produce the same colors in F and F'. Consider an observation string $y_1 y_2 \cdots y_{k+1}$ of length $k+1$. Let the last edge with observation y_{k+1} go in F' to a state q_x with color x. We know that q_x exists, because all observation sequences are in $L(F)$. We also know that this state has the same color as the state reached by $y_1 y_2 \cdots y_{k+1}$ in F, since the observation y_{k+1} uniquely determines the color of the target state. So, the $k+1$th observation (and, since k is a variable, the entire input sequences) produce the same output colors in both F and F'. Therefore, $F \overset{\mathcal{L}(F)}{=\!=} F'$.

Also, note that F' is minimal. Any filter with fewer states than F', must omit at least one color that appears in F, and so would not be equivalent to F. □

5.2.3. Filters with observations that appear only twice

The previous subsection considered observations that appear only once. What happens if we relax this constraint slightly, and allow each observation to appear at most twice?

Definition 7. A twice-appearing-observations filter is a filter in which every observation appears at most twice.

Problem: Twice-appearing-observations Filter Minimization (TWICE-APPR-OBS-FM)

Input: A twice-appearing-observations filter F.

Output: A filter $F^\star$ such that $F \xlongequal{\mathcal{L}(F)} F^\star$ and the number of I-states in $F^\star$ is minimal.

Theorem 5. TWICE-APPR-OBS-FM *is NP-hard.*

Proof. Similar to the proof of Theorem 2. Note in particular that the construction in Section 4 does not use any observation more than twice. Therefore, we conclude that TWICE-APPR-OBS-FM is NP-hard, by reduction from GC. □

In fact, Theorem 5 readily generalizes to the following a parameterized family of problems.

Problem: m-Appearing-observations Filter Minimization (m-APPR-OBS-FM)

Input: A filter F in which each observation appears at most m times.

Output: A filter $F^\star$ such that $F \xlongequal{\mathcal{L}(F)} F^\star$ and the number of I-states in $F^\star$ is minimal.

Corollary 3. *For any $m > 1$, m-*APPR-OBS-FM *is NP-hard.*

6. Hardness of approximate filter reduction

In this section, we show that FM is still NP-hard, even if we are willing to settle for solutions that are mere approximations of the optimal solution. We rely upon the following result due to Zuckerman [30].

Lemma 4 *(Zuckerman [30]). It is NP-hard to approximate* GC *within $n^{1-\epsilon}$, for any $\epsilon > 0$.*

Our approach builds upon the conversion from Section 4, which established a connection between filter minimization and the graph coloring problem. Recall that, given an instance $G = (V_1, E_1)$ of GC, we construct an instance $F(G) = (V_2, E_2, l, c, v_0)$ of FM, in polynomial time.

Theorem 6. FM *is NP-hard to approximate with ratio $n^{1-\epsilon}$, for any $\epsilon > 0$.*

Proof. Let $\epsilon > 0$. Suppose that there exists a polynomial time approximation algorithm A that solves FM with approximation ratio $n^{1-\epsilon}$. Let B denote an approximation algorithm for GC that works as follows:

1. Given an instance G of GC, form the filter $F(G)$ as described in Section 4.
2. Use algorithm A to reduce $F(G)$.
3. Extract a coloring of G from the reduced filter.

We now argue that B has an approximation ratio $n^{1-\frac{\epsilon}{2}}$.

Let $A(F)$ denote the size of the reduced filter produced by algorithm A from input filter F, and likewise $B(G)$ denotes the number of colors used by B to color G. Let $OPT(F)$ and $OPT(G)$ represent the minimal filter size and minimal number of colors for those instances, respectively. According to the assumption, we have $A(F) \le n^{1-\epsilon} OPT(F)$.

Thus, we have an approximation algorithm B for GC. According to Lemma 2, we have $OPT(F) = OPT(G) + 3$. Then, for sufficiently large n,

$$\begin{aligned} B(G) &= A(F(G)) - 3 \\ &\le A(F(G)) \\ &\le n^{1-\epsilon} OPT(F) \\ &\le n^{1-\epsilon}(OPT(G) + 3) \\ &\le n^{1-\frac{\epsilon}{2}} OPT(G). \end{aligned}$$

Therefore, B is an approximation algorithm for GC with approximation ratio $n^{1-\frac{\epsilon}{2}}$ which contradicts Lemma 4. Since that lemma is applies for any $\epsilon > 0$, it also applies for $n^{1-\frac{\epsilon}{2}}$. □

Note that this result carries over to each of the special cases of FM proven NP-hard in Section 5.

Corollary 4. *The following problems are NP-hard to approximate with ratio $n^{1-\epsilon}$, for any $\epsilon > 0$:*

1. TREE-FM,
2. BIPARTITE-FM,
3. PLANAR-FM,
4. TWICE-APPR-OBS-FM, *and*
5. m-APPR-OBS-FM, *for any* $m > 1$.

The results of this section show that FM and some of its special cases are very challenging, in the sense that there are some strong limits on how efficiently approximate solutions can be found. In practice, approximation may remain a useful tool for these problems, though we cannot make any guarantees. For example, there is a well-known result on approximation for graph coloring [31] presents a performance guarantee of $O(n(\log\log n)^2/(\log n)^3)$. This ratio, which is close to linear, may sometimes be reasonable in practice.

7. Analysis of the existing heuristic algorithm

In addition to the NP-hardness proof for FM, O'Kane and Shell's original paper also introduced a heuristic algorithm for this problem. Though this algorithm does not make any guarantee of optimality, it is still of interest because it is, to the authors' knowledge, the only known algorithm that attempts to solve FM. In this section, after a brief review of the algorithm, we disprove a conjecture about the conditions under which it produces optimal solutions, and introduce a different, stronger condition that *is* sufficient for optimality. This latter result will be of use in Section 8 for designing a fixed-parameter tractable algorithm for FM.

7.1. Summary of O'Kane and Shell's algorithm

This subsection sketches the algorithm introduced by O'Kane and Shell. For a complete description, we refer the reader to the original paper [4]. Consider the following definitions.

Definition 8 *(O'Kane and Shell [4]).* In a filter $F \triangleq (V, E, l : E \to Y, v_0, c)$, two vertices $v \in V$, $w \in V$ are *in conflict* if $c(v) = c(w)$ and there exists an observation y and edges $v \xrightarrow{y} v'$ and $w \xrightarrow{y} w'$ such that $c(v') \neq c(w')$. A color k is called *conflicted* if at least one pair of vertices assigned to that color are in conflict.

Definition 9 *(O'Kane and Shell [4]).* In a filter, the *conflict graph for color* k is an undirected graph with vertex set $\{v \in V \mid c(v) = k\}$ and edge set $\{(v, w) \mid v \text{ is in conflict with } w\}$.

The algorithm is based on the observation that a filter with no conflicts can be reduced by "merging" each set of same-colored states. Since there are no conflicts, there is no ambiguity how to draw the edges: The definition of (lack of) conflicts ensures that, for each color and each observation, there is at most one target color.

Thus, the algorithm proceeds as follows: It selects a conflicted color, computes the conflict graph for that color, uses a graph-coloring subroutine to color this conflict graph, and changes the colors of the filter accordingly. This refinement of the colors eliminates the conflicts for this color. The algorithm iterates that process until no conflicts remain, and then forms the final filter by merging all same-colored states, and restoring the original colors of the input filter.

Both the runtime of the algorithm and the size of the output filter depend on the details of the subroutine used to color the conflict graphs.

7.2. A conjecture disproved

O'Kane and Shell observed that, in a small set of tests, whenever an optimal coloring algorithm was used to color the conflict graphs, the resulting reduced filter was also optimal. Based on this observation, they made the following conjecture.

Conjecture 1. *The heuristic algorithm of O'Kane and Shell is guaranteed to find an optimal reduced filter, if the algorithm for coloring conflict graphs is optimal.*

We can now resolve this conjecture.

Theorem 7. *Conjecture 1 is false.*

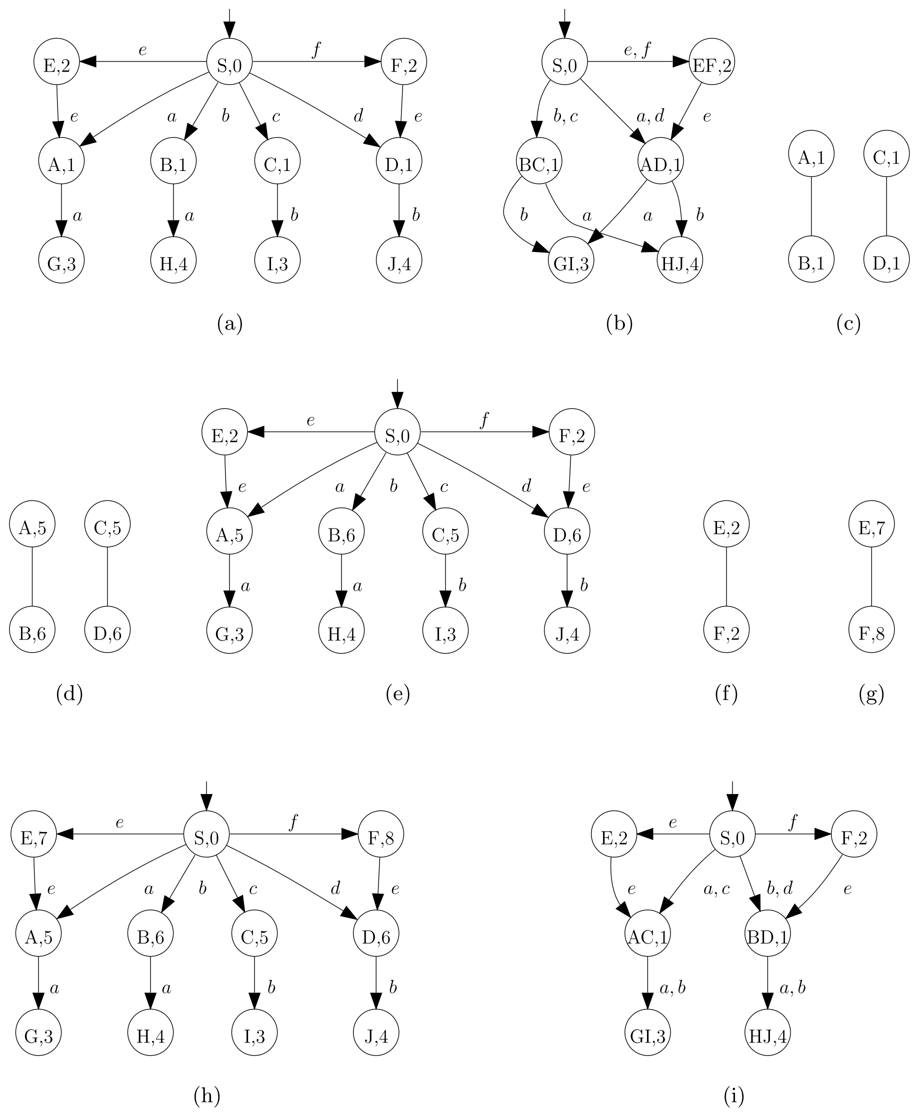

Fig. 6. A counterexample to O'Kane and Shell's conjecture. (a) The original unreduced filter. (b) There exists a solution with 6 states for the input filter. (c) The conflict graph of color 1. (d) An optimal coloring of conflict graph of color 1. (e) After the first refinement of color 1. (f) The new conflict graph for color 2. (g) The optimal coloring of conflict graph for color 2. (h) After the second refinement. (i) After merging same colors and replacing all colors to their original colors.

Proof. A counterexample is shown in Fig. 6. Fig. 6a shows the input filter. The algorithm's objective is to reduce this filter to the smallest equivalent filter. Fig. 6b indicates there exists an equivalent reduced filter with 6 states for this input filter. We now show that the algorithm can fail to reduce the input filter to six states, even if it colors the conflict graphs optimally.

The algorithm first finds conflicts for color 1 in input filter. The conflict graph is shown in Fig. 6c. Then it colors this graph to resolve these conflicts, with the optimal coloring shown in Fig. 6d. (Note that there are multiple optimal colorings for this graph. This multiplicity of optimal solutions turns out to be crucial, as Theorem 8 shows.)

Fig. 6e shows how this coloring refines the colors of original filter: Color 1 is replaced by two new colors, 5 and 6. After that, the algorithm discovers a new conflict for color 2. The conflict graph is shown in Fig. 6f, so we repeat the refinement process again (Figs. 6g and 6h). At this point, no conflicts remain.

Fig. 6i shows the resulting reduced filter produced by the algorithm. It has 7 states. This filter is equivalent to the original one, but is clearly not optimally reduced, because it has 7 states, whereas Fig. 6b shows that a reduction to 6 states is possible. □

Note that, in the example, there exists another coloring of the conflict graph for color 1 in Fig. 6c that assigns A and D to the same color, and B and C to the same color. If the algorithm does choose this coloring, then it does lead to the optimal solution as shown in Fig. 6b.

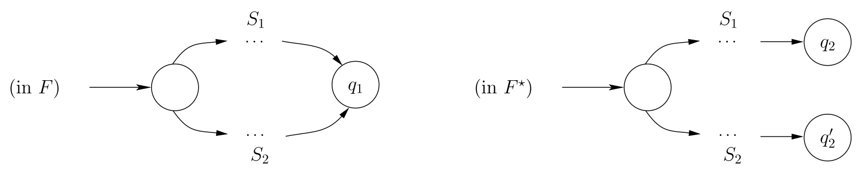

Fig. 7. The construction used in Lemma 5. [left] In filter F, observation sequences S_1 and S_2 reach state q_1. [right] In filter F^*, the same observation sequences lead to two distinct states. The proof of Lemma 5 shows that this situation occurs only if $q_2 = q_2'$.

7.3. Some optimal coloring is sufficient

The proof of Theorem 7 relies on the existence of a conflict graph that has multiple distinct optimal colorings. A natural question, then, is to ask whether the existence of multiple optimal colorings is essential for extracting suboptimal results from O'Kane and Shell's algorithm. In this subsection, we answer that question by showing that there always exists some optimal conflict graph coloring which, if produced by the graph coloring subroutine, would lead O'Kane and Shell's algorithm to the optimal solution.

The following lemma will be helpful for constructing a coloring that satisfies these conditions.

Lemma 5. *Let F denote a filter and let $F^\star$ denote the smallest filter for which $F \overset{\mathcal{L}(F)}{=\!=} F^\star$. For each state q_1 in input filter F, there exists a single state q_2 in $F^\star$, such that every observation sequence that leads to q_1 in F leads to q_2 in $F^\star$.*

Proof. Without loss of generality, we assume that every edge in $F^\star$ is crossed by at least one observation sequence in $\mathcal{L}(F)$.

For a given filter F and observation sequence $S \in \mathcal{L}(F)$, let $f(F, S)$ denote the final state reached by tracing the observations in S through F, starting from the initial state. Define a relation $\sim$ over ordered pairs of states from F and $F^\star$, in which $q_1 \sim q_2$ iff there exists an observation sequence $S \in \mathcal{L}(F)$ for which $f(F, S) = q_1$ and $f(F^\star, S) = q_2$. We need to show that if $q_1 \sim q_2$ and $q_1 \sim q_2'$, then we must have $q_2 = q_2'$.

Suppose not. That is, suppose there exists a state q_1 in F and two distinct states q_2 and q_2' in $F^\star$ for which $q_1 \sim q_2$ and $q_1 \sim q_2'$. This implies that there exist two different observation sequences, S_1 and S_2, such that in F we have $f(F, S_1) = f(F, S_2) = q_1$, but in $F^\star$ we have $f(F^\star, S_1) = q_2$ and $f(F^\star, S_2) = q_2'$. See Fig. 7.

We now show how to construct a new filter F_3 that is also equivalent to F, but smaller than $F^\star$. We can construct F_3 from $F^\star$ by a merging operation. Start by merging q_2 and q_2' in $F^\star$ to form a new state q_2q_2' in F_3. Since $F \overset{\mathcal{L}(F)}{=\!=} F^\star$, these two states must have the same color. Then we copy all incoming edges of q_2 and q_2' to be incoming edges of this new state. For outgoing edges, there are three cases.

1. Some observations may be present as labels for outgoing edges of q_2 or q_2', but not both. In this case, we add to F_3 an edge labeled by this observation to the same destination as in $F^\star$.
2. Some observations may be present as labels for outgoing edges of both q_2 and q_2', but with both of those edges going to the same state. In this case, we can add the corresponding single edge, from q_2q_2' to the shared target state, to F_3.
3. The remaining observations appear as labels for out-edges of both q_2 and q_2', with distinct target states. Observe that the equivalence of F and $F^\star$ ensures that those two different destination states must have the same colors. In this case, we recursively merge those destination states. The recursion is guaranteed to terminate, because each merge reduces number of states by one.

This construction clearly leads to a valid filter, since it specifically eliminates situations in which a state has multiple out-edges for a single observation. Moreover, note since each transition is guaranteed to reach the same colors in both $F^\star$ and F_3, we have $F^\star \overset{\mathcal{L}(F)}{=\!=} F_3$. This implies, by transitivity of filter equivalence, that $F \overset{\mathcal{L}(F)}{=\!=} F_3$. This is a contradiction to the assumed minimality of $F^\star$. □

The intuition of Lemma 5 is that we can always think of an optimally reduced filter as being formed by a collection of equivalence classes of states in the original filter, determined by their unique corresponding state in the reduced filter. These equivalence classes allow us to form a coloring for each conflict graph, under which O'Kane and Shell's algorithm produces optimal results.

Theorem 8. *In O'Kane and Shell's heuristic algorithm for* FM, *there always exists a coloring for each conflict graph, under which the algorithm generates an optimally reduced filter.*

Proof. Consider an arbitrary input filter F, and let $F^\star$ denote an optimal equivalent reduced filter. This $F^\star$ is, of course, unknown to the algorithm. Recall that the algorithm works by resolving all conflicts of F and then merging same colored states. As an overview of the structure of the proof, the intuition is to show that there exists a coloring c for all of the

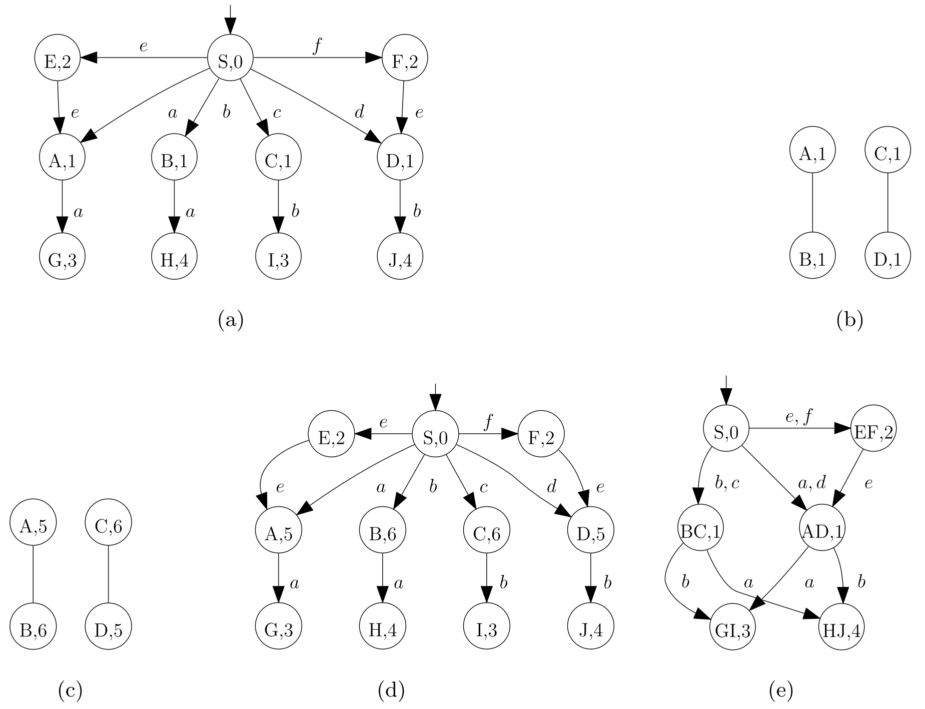

Fig. 8. An example for which O'Kane and Shell's algorithm produces an optimal reduced filter. This problem is the same as Fig. 6, but the sequence of conflict graph colorings differs. (a) The original unreduced filter. (b) The conflict graph of color 1. (c) We color the conflict graph of color 1 by coloring of optimal reduced filter, such that the states A and D which are merged in $F^\star$, have the same color 5 and likewise the states B and C have the same color 6. (d) The colored original filter after refinement of colors. (e) The optimal reduced filter.

states of F that simultaneously has two properties: first, that the colors specified by c form a valid coloring of any conflict graph reached by O'Kane and Shell's algorithm (in the sense of assigning different colors to conflicting states), and second, that the final reduced filter produced by this process is indeed $F^\star$. The latter property is implied by our construction of c; we then show that the former also holds.

For each state q_1 of F, Lemma 5 implies that this state maps to a single state q_2 in $F^\star$. This mapping induces an equivalence relation on the states of F, in which two states q_1 and q_1' are equivalent if they map to the same q_2. Based on this equivalence relation, we form a coloring function c for the states of F under which, for any two states q_1 and q_1' in F, we have $c(q_1) = c(q_1')$ iff q_1 and q_1' share the same corresponding state in $F^\star$. Specifically, we consider an arbitrary ordering $1, 2, 3, \ldots$ on the equivalence classes of nodes in F, then define c so that any state q in the first equivalence class has $c(q) = 1$, any state q' in the second equivalence class has $c(q') = 2$, and so on. During the execution of O'Kane and Shell's algorithm, we can color the conflict graphs as follows. Each node in a conflict graph is a state in F. In the previous paragraph, we defined a coloring c on the states of F. We use that coloring c to color the conflict graphs. See Fig. 8 for an example.

Moreover, by construction, the resulting filter will be $F^\star$. It remains to show that c is a valid coloring for every conflict graph encountered by the algorithm. That is, for every conflict between a pair of states u and v reached during the algorithm's execution, we have $c(v) \neq c(u)$.

Note that every conflict can be classified as a primary conflict or a secondary conflict. The primary conflicts are ones which already exist in the original filter; secondary conflicts arise from refinements to the colors of F's states at earlier iterations of the algorithm.

Suppose for a contradiction that we reach a conflict graph C in F such that two nodes u and v of C are connected by an edge and have the same color under c. This implies that u and v are merged in $F^\star$. This conflict must be either primary or secondary.

1. If it is a primary conflict, then u and v by the same observation go to two different destinations u' and v' with different output colors in F. In this case, u' and v' can not merge, because they have different colors in the original filter F. See Fig. 9. This implies that u and v can not be merged in any filter equivalent to F, which is a contradiction to the construction of c.
2. If it is a secondary conflict, then u and v were in conflict in original filter. Therefore, there exist two states u' and v' in F such that u' and v' had a conflict (either primary or secondary) at an earlier iteration. Thus, in the induced coloring c of F, we have two nodes u' and v' with different colors such that u and v by the same observation go to them, that

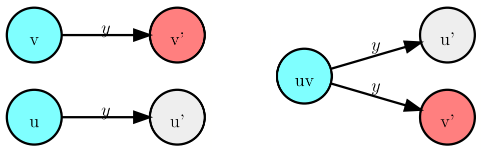

Fig. 9. [right] A part of an induced coloring c of F in which u and v have a conflict, either primary or secondary. [left] Two states u and v are merged in $F^\star$, because they have the same color.

is $u \xrightarrow{y} u'$ and $v \xrightarrow{y} v'$. The left side of Fig. 9 illustrates this case. Because u' and v' were separated by c in some previous iteration of the algorithm, we know that $c(v) \neq c(u)$ and therefore, u' and v' are not merged in $F^\star$.
Recall that u and v merge together in reduced final filter $F^\star$. See the right side of Fig. 9. In this case, the merged state uv in $F^\star$ has two identical observations, going to two different destinations. This implies that $F^\star$ is not a valid filter, since it has a single state with two out-edges with the same observation. This is a contradiction.

We conclude therefore, that if O'Kane and Shell's algorithm uses c to color each of its conflict graphs, then it will be a valid coloring at every iteration, and that the final reduced filter will indeed be $F^\star$. □

In this section, we proved that O'Kane and Shell's heuristic filter reduction algorithm cannot guarantee to reach the optimal solution merely by relying on the optimality of its graph coloring subroutine. We also showed that there always exists a coloring that leads to optimal solution by the algorithm. This second insight leads to the fixed-parameter tractable algorithm in the next section, which is based on enumerating the valid colorings of each conflict graph.

8. Fixed-parameter approach

So far, we have considered several of the usual tools for attacking the NP-hardness of FM—special cases, approximation, and heuristic algorithms—and showed that these techniques are, at best, of limited use for solving this problem. In this section, we consider one final approach, namely parameterized complexity. The idea behind this approach is to identify parameters, apart from the problem size, that govern hardness of a problem. We hope to show that algorithms exist whose run time, while possibly very high as a function of the selected parameters, depends only polynomially on the problem size.
Consider the standard definitions of parameterized and fixed-parameter tractable problems.

Definition 10 *(Niedermeier [32]).* A *parameterized problem* is a language $L \subseteq \Sigma^* \times \Sigma^*$, where Σ is a finite alphabet. The second component is called the parameter of the problem.

The parameter usually has been considered as a nonnegative integer, so $L \subseteq \Sigma^* \times \mathbb{N}$.

Definition 11 *(Niedermeier [32]).* A parameterized problem L is *fixed-parameter tractable* if it can be determined in $f(k) \cdot n^{O(1)}$ time whether or not $(x, k) \in L$, where f is a computable function only depending on k. The corresponding complexity class is called FPT.

In this section, we will present some parameters for FM and check their fixed-parameter tractability.

8.1. Solution size

Before considering any parameters for FM specifically, first we recall the basic ideas of parameterized reductions.

Definition 12 *(Flum and Grohe [33]).* Let (Q, k) and (Q', k') be parameterized problems over the alphabets Σ and Σ', respectively. An fpt-reduction (more precisely, fpt many-one reduction) from (Q, k) to (Q', k) is a mapping $R : \Sigma^* \to (\Sigma')^*$ such that:

1. For all $x \in \Sigma^*$ we have $(x \in Q \iff R(x) \in Q')$.
2. R is computable by an fpt-algorithm (with respect to k). That is, there is a computable function f and a polynomial $p(X)$ such that $R(x)$ is computable in time $f(k(x)) \cdot p(|x|)$.
3. There is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that $k'(R(x)) \leqslant g(k(x))$ for all $x \in \Sigma^*$.

Lemma 6 *(Flum and Grohe [33]). FPT is closed under fpt-reductions. That is, if a parameterized problem (Q, k) is reducible to a parameterized problem (Q', k') and $(Q', k') \in$ FPT, then $(Q, k) \in$ FPT.*

Now we can show that FM, parameterized by the size of the smallest equivalent filter, is not FPT. To do this, we present a parameterized reduction from GC to FM.

Lemma 7. GC *parameterized by the minimum number of colors is fpt-reducible to* FM *parameterized by the size of smallest equivalent filter.*

Proof. Parameterize GC as (Q, k), in which Q is an undirected graph and k is chromatic number of the graph. Parameterize FM as (Q', k'), in which Q' is a filter and k' is the size of the smallest equivalent filter. Let R denote the GC-to-FM conversion algorithm from Section 4.

We want to show that R is an fpt-reduction. Definition 12 has three conditions. Condition 1 is confirmed by Lemmas 1 and 2. For condition 2, we note that R runs in time polynomial in its input size. For condition 3, we use $g : n \mapsto n+3$, which is trivially computable. Again, Lemmas 1 and 2 show that, for any GC instance x, we have $k'(R(x)) = g(k(x))$. Therefore, R is an fpt-reduction. □

Lemma 8. *If $P \neq NP$, then* FM *parameterized by the size of the smallest equivalent filter is not FPT.*

Proof. Suppose for a contradiction that there exists an FPT algorithm for FM parameterized by the size of the smallest equivalent filter. Based on Lemmas 7 and 6, this implies that GC parameterized by the minimum number of colors is FPT. This, in turn, implies that there is an algorithm for the 3-coloring problem in time $f(3) \cdot n^{O(1)}$, which is a polynomial time algorithm for this problem. If $P \neq NP$, then this is a contradiction, because the 3-coloring problem is NP-Complete [34]. □

8.2. Treewidth

We now consider the parameter of treewidth of an input filter. Recall that treewidth is an integer measure of how similar a graph is to a tree that ranges from 1 (for trees and forests) to $|V| - 1$ (for complete graphs) [32]. Though many graph problems are FPT when parameterized by treewidth, this is not the case for FM.

Lemma 9. *If $P \neq NP$, then* FM *parameterized by the treewidth of the input filter is not FPT.*

Proof. We use a contradiction. Assume that there exists a FPT algorithm for FM parameterized by treewidth. This problem includes, as a special case, filters that are trees. Since trees have $k = 1$, this implies an algorithm with runtime $f(1) \cdot n^{O(1)}$—a polynomial time algorithm for TREE-FM. If $P \neq NP$, this contradicts Theorem 2. □

Therefore, the treewidth of a filter is not a promising parameter. This may be somewhat surprising, because GC parameterized by treewidth is FPT. In fact, GC is polynomially solvable for trees, but FM is not even FPT with this parameter.

8.3. Number of appearances of each observation

Next, we revisit the restrictions on observations from Subsection 5.2.3.

Lemma 10. *If $P \neq NP$, then* FM *parameterized by the maximum number of appearances of any observation in the input filter is not FPT.*

Proof. Another contradiction. Suppose FM with this parameter is FPT. Then for the specific case of $k = 2$, we have an algorithm for TWICE-APPR-OBS-FM that runs in time $f(2) \cdot n^{O(1)}$. This contradicts Theorem 5, unless $P = NP$. □

Thus, this parameter is also apparently useless to study fixed-parameter tractability of FM, at least as a single parameter.

8.4. A fixed-parameter tractable algorithm

In this subsection, we present a fixed-parameter tractability result for the general problem FM, parameterized with two parameters simultaneously. The general idea is to leverage Theorem 8, which guarantees that in O'Kane and Shell's heuristic algorithm, there always exists a conflict graph coloring that leads to an optimal solution. We describe an algorithm that enumerates each of these colorings, recursively branching to consider all of the possibilities. We introduce a pair of parameters that bound the size of this recursion tree.

The following definition describes the first parameter precisely.

Definition 13. The weight of each color in a filter is the number of states with that color in the filter. The *maximum weight of colors* is the maximum weight among all the colors' weights.

The first parameter is the maximum weight of colors in input filter. The intuition is that when this parameter is bounded, number of conflicts for each color in input filter and number of optimal colorings of conflict graphs of that color are also

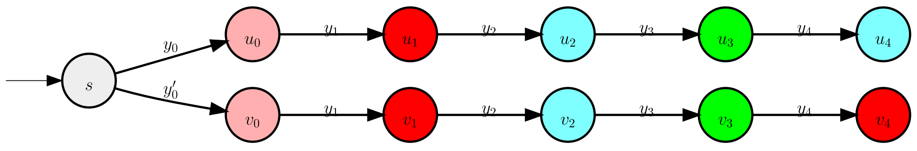

Fig. 10. An example of fourth-order separation for u_4 and v_4. The two states u_4 and v_4 with different colors cause a primary conflict between u_3 and v_3. We need to four refinements to resolve all conflicts and so, the value of total separation depth in this filter is 4.

Algorithm 2: A FPT algorithm for FM which is bounded by parameters of maximum weight of colors and total separation depth of input filter.

```
Input  : A filter $F_1 \triangleq (V, E, l : E \to Y, v_0, c_1)$
Output: The smallest filter $F_2{}^*$, such that $F_1 \overset{\mathcal{L}(F_1)}{=\!=\!=} F_2{}^*$
1 $F_2{}^* \leftarrow \emptyset$
2 RESOLVE($F_1$)
3 return $F_2{}^*$
```

bounded. For example, if the value of the parameter is two, then the conflict graph of each color in the filter has at most two nodes and the chromatic number is at most two, and there is just one optimal coloring.

Consider the following definition to describe the second parameter.

Definition 14. There is a *kth order separation* between two states q_1 and q_2, if $c(q_1) \neq c(q_2)$ and there exists an observation sequence $y_1, ..., y_k$ of length k, along with two starting states $q_0^{(1)}$ and $q_0^{(2)}$, such that following that observation sequence starting from $q_0^{(1)}$ reaches a sequence of states $q_0^{(1)}, q_1^{(1)}, ..., q_k^{(1)}$ without repetition; and starting from $q_0^{(2)}$ reaches a sequence of states $q_0^{(2)}, q_1^{(2)}, ..., q_k^{(2)}$ without repetition; such that $c(q_i^{(1)}) = c(q_i^{(2)})$ for all $i = 1, ..., k-1$ and $q_k^{(1)} = q_1$ and $q_k^{(2)} = q_2$.

Definition 15. For any pair of states with different colors, *separation depth* is the largest k for which that pair has a kth order separation. For any pair of states with the same colors, separation depth is defined to be zero.

The second parameter is the *total separation depth* in input filter F_1, that is the summation of separation depths across all pairs of states in F_1. See Fig. 10. In this figure, the value of first parameter is three and the primary conflict between u_3 and v_3 has just one optimal coloring. However, this coloring and its refinement to F_1 causes a secondary conflict between u_2 and v_2. Therefore, tracing the observation sequence of $y_1, ..., y_4$ with starting from u_0 and v_0 with color pink, leads to a separation depth of four for u_4 and v_4 and the value of total separation depth in this filter is four. Thus, in filters with small total separation depth, the number of necessary refinements to the filter's coloring is also small.

We now show our fixed-parameter tractability result by the following theorem.

Theorem 9. FM, *parameterized simultaneously by maximum weight of colors and total separation depth, is FPT.*

Proof. Given a filter F_1, with maximum weight of colors k and total separation depth k', we present an algorithm that is FPT for k and k'. This algorithm is an adaptation of O'Kane and Shell's heuristic algorithm. See Algorithm 2 and its subroutine Algorithm 3. This algorithm picks one conflicted color in F_1 and creates its conflict graph C. It then tries all possible colorings for C and refines those colorings on F_1 and continues recursively until F_1 filter has no conflicts. At each leaf of this recursion tree, the algorithm forms a reduced filter F_2; the final return value is the smallest such F_2.

The correctness of this algorithm follows from Theorem 8. Because we try all coloring possibilities for each conflict graph, we are sure that at least one leaf of the recursion tree reaches the minimum reduced equivalent filter F_2.

How many nodes are in the recursion tree? The number of children for each node is equal to the number of valid colorings for the selected conflict graph C. To enumerate all possible colorings C, we count all i-colorings of C for $i = 1, ..., k$. The number of all different ways to color a graph with i colors, known as the *chromatic polynomial* [35] and denoted $P(C, i)$, is polynomial in terms of i and of degree equal to graph's size. So, we need to compute $\sum_{i=1}^{k} P(C, i) \leq \sum_{i=1}^{k} i^k$ that is at most k^{k+1}. Therefore, each node in the tree has at most k^{k+1} children.

The height of tree is at most the total separation depth, k'. This is true because any simple path through the tree from its root to a leaf corresponds a sequence of conflict graph colorings, starting from the input filter and ending at a filter with no conflicts. The length of such a path is at most the same as number of re-colorings of conflict graphs.

To analyze the algorithm's total run time, note that the total number of nodes in the recursion tree is at most $(k^{k+1})^{k'}$. In each internal node, the algorithm creates the conflict graph C and then uses the coloring of C to refine F_1. To create a conflict graph from a filter F with n states, we try all pairs of states and for each pair we check all observations of outgoing edges for those two states. Since any state has at most n observations, checking all outgoing edges takes $O(n^2)$ time. Thus,

Algorithm 3: RESOLVE(F).

```
Input: A filter $F \triangleq (V, E, l : E \to Y, v_0, c)$
1  if $F$ has a conflicted color then
2      Pick a color $c$ that has conflict.
3      Compute the conflict graph $C$ for color $c$ in $F$.
4      for each coloring $c'$ of $C$ do
5          Refine the coloring of $F$, replacing $c$ with coloring $c'$.
6          RESOLVE(F)
7      end
8  else
9      Reduce $F_1$ by merging same-colored states.
10     if $F_2{}^* = \emptyset$ or $|F_2| \le |F_2{}^*|$ then
11         $F_2{}^* \leftarrow F_2$
12     end
13 end
```

creating a conflict graph takes $O(n^4)$ time. Also, refinement of the conflict graph on F takes $O(n)$ time. At each leaf node of the recursion tree, we have to spend $O(n)$ time to form the reduced filter by merging same-colored nodes. Therefore, Algorithm 2 is a FPT algorithm for FM, because it runs in time $(k^{k+1})^{k'} \cdot O(n^4)$. □

9. Conclusion

9.1. Summary

The problem of automatic reduction of combinatorial filters is NP-hard. In this paper, we tried to cope with its hardness in various ways. Most of the results can be classified, informally, as “bad news.”

We proved that this problem remains NP-hard for some special cases in which the input filter has a specific structure such as tree, bipartite, and planar filters. We also studied some other particular filters that have limitations on their observations and showed that the problem is still NP-hard for filters with at most m appearances of each observation, for any $m > 1$. We showed that if there are no missing edges in the input filter, the problem is essentially the same as DFA minimization problem and can be solved in polynomial time. We also proved that if the input filter has observations with just one appearance, then filter reduction problem belongs to complexity class P, and we presented a polynomial time algorithm to solve it.

In another thread, we investigated the possibility of approximation, and showed that the general problem and the special cases of tree, bipartite and planar filters, are NP-hard to approximate with ratio $n^{1-\epsilon}$, for any $\epsilon > 0$.

We were interested to study the only existing filter reduction algorithm. The idea was that if we could find in which cases we can produce the optimal solution then, we would introduce some nice parameter for solving FM using fixed-parameter approach. So, we showed that the conjecture of “If the algorithm colors conflict graphs optimally, then it is guaranteed to find an optimal reduced filter” is false. We also proved that there exists a way that this algorithm leads to the optimal solution.

Then, we turned fully to the fixed-parameter approach, because this method tries to find parameters that cause the hardness of a problem and to hold them fixed to make it tractable in terms of input size. With this approach, we showed that filter reduction problem is not FPT with parameters of solution size, treewidth, number of appearances of each observation. We showed that the problem is FPT with two simultaneous parameters of maximum weight of colors and total separation depth of the input filter, and we presented a FPT algorithm for it.

9.2. Future work

For future work, there are some other classes for filters that have not been addressed, for example, grid filters or filters with just two distinct observations. Also, checking approximation hardness and fixed-parameter tractability of these sort of filters is interesting.

It seems that studying fixed-parameter tractability of filter reduction problem with some other parameters is useful. More precisely, considering the parameter of number of missing observations in an input filter and number of distinct observation seems worthy of study.

Also, we note that there is a strong connection between graph coloring and filter minimization. If we can characterize the set of filters for which the conflict graph can be colored efficiently, such as a perfect graph, this would lead to a polynomial time algorithm for this sort of filters.

Another thread of research is studying all of the above mentioned open problems for “improper” filter reduction, that is, reducing a given filter to a given size such that the resulted filter is “close to” the input filter, according to some metric, tolerating an amount of error. In this paper and in O’Kane and Shell’s paper [4], the problem is finding the optimal reduced filter which is equivalent to the input filter. To extend this idea to tolerate solutions that are merely close to the input filter, we need a definition, analogous to this paper's Definition 4, of similarity between filters. A paper on this subject by the present authors, using a notion of worst-case edit distance between pairs of filters as the similarity metric, is under review elsewhere.

Acknowledgments

Jason M.O'Kane is supported by the National Science Foundation under Grants 0953503 and 1526862.

References

[1] S.M. LaValle, Planning Algorithms, Cambridge University Press, Cambridge, UK, 2006, available at http://planning.cs.uiuc.edu/.
[2] S.M. LaValle, Sensing and filtering: a fresh perspective based on preimages and information spaces, Found. Trends® Robot. 1 (4) (2012) 253–372.
[3] B. Tovar, F. Cohen, L. Bobadilla, J. Czarnowski, S.M. LaValle, Combinatorial filters: sensor beams, obstacles, and possible paths, ACM Trans. Sens. Netw. 10 (3) (2014) 47.
[4] J.M. O'Kane, D.A. Shell, Automatic reduction of combinatorial filters, in: Proc. IEEE International Conference on Robotics and Automation, 2013.
[5] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, Cambridge University Press, MIT Press, Cambridge, MA, 2005.
[6] R. Kalman, A new approach to linear filtering and prediction problems, J. Basic Eng. (1960) 35–45.
[7] M. Erdmann, M.T. Mason, An exploration of sensorless manipulation, IEEE Trans. Robot. Autom. 4 (4) (1988) 369–379.
[8] K.Y. Goldberg, Orienting polygonal parts without sensors, Algorithmica 10 (1993) 201–225.
[9] R. Lopez-Padilla, R. Murrieta-Cid, S.M. LaValle, Optimal gap navigation for a disc robot, in: Proc. Workshop on the Algorithmic Foundations of Robotics, 2012.
[10] B. Tovar, Minimalist models and methods for visibility-based tasks, Ph.D. thesis, University of Illinois at Urbana Champaign, 2009.
[11] L. Bobadilla, O. Sanchez, J. Czarnowski, S.M. LaValle, Minimalist multiple target tracking using directional sensor beams, in: Proc. IEEE International Conference on Intelligent Robots and Systems, 2011.
[12] J. Yu, S.M. LaValle, Shadow information spaces: combinatorial filters for tracking targets, IEEE Trans. Robot. 28 (2) (2012) 440–456.
[13] G. Lagunaa, R. Murrieta-Cid, H.M. Becerra, R. Lopez-Padilla, S.M. LaValle, Exploration of an unknown environment with a differential drive disc robot, in: Proc. IEEE International Conference on Robotics and Automation, 2014.
[14] V. Narayanan, P. Vernaza, M. Likhachev, S.M. LaValle, Planning under topological constraints using beam-graphs, in: Proc. IEEE International Conference on Robotics and Automation, 2013.
[15] J. Yu, S.M. LaValle, Story validation and approximate path inference with a sparse network of heterogeneous sensors, in: Proc. IEEE International Conference on Robotics and Automation, 2011.
[16] J. Yu, S.M. LaValle, Cyber detectives: determining when robots or people misbehave, in: Proc. Workshop on the Algorithmic Foundations of Robotics, 2010.
[17] B. Tovar, F. Cohen, S.M. LaValle, Sensor beams, obstacles, and possible paths, in: Proc. Workshop on the Algorithmic Foundations of Robotics, 2008.
[18] Y. Song, J.M. O'Kane, Comparison of constrained geometric approximation strategies for planar information states, in: Proc. IEEE International Conference on Robotics and Automation, 2012.
[19] S. Kristek, D. Shell, Orienting deformable polygonal parts without sensors, in: Proc. IEEE International Conference on Intelligent Robots and Systems, 2012.
[20] M. Erdmann, On the topology of discrete strategies, Int. J. Robot. Res. 29 (7) (2010) 855–896.
[21] M. Erdmann, On the topology of discrete planning with uncertainty, in: Advances in Applied and Computational Topology, Proceedings of Symposia in Applied Mathematics, American Mathematical Society, 2012.
[22] J.M. O'Kane, D.A. Shell, Finding concise plans: hardness and algorithms, in: Proc. IEEE International Conference on Intelligent Robots and Systems, 2013.
[23] J.M. O'Kane, D.A. Shell, Automatic design of discreet discrete filters, in: Proc. IEEE International Conference on Robotics and Automation, 2015.
[24] Y.-C. Wu, V. Raman, S. Lafortune, S.A. Seshia, Obfuscator synthesis for privacy and utility, in: Proc. 8th NASA Formal Methods Symposium, 2016.
[25] N. Roy, G. Gordon, S. Thrun, Finding approximate POMDP solutions through belief compression, J. Artif. Intell. Res. 23 (2005) 1–40.
[26] J. Ballesteros, L. Merino, M.A. Trujillo, A. Viguria, A. Ollero, Improving the efficiency of online POMDPs by using belief similarity measures, in: Proc. IEEE International Conference on Robotics and Automation, 2013.
[27] P.A. Crook, S. Keizer, Z. Wang, W. Tang, O. Lemon, Real user evaluation of a POMDP spoken dialogue system using automatic belief compression, Comput. Speech Lang. 28 (4) (2014) 873–887.
[28] L. Erickson, Y.H.J. Yu, S.M. LaValle, Counting moving bodies using sparse sensor beams, IEEE Trans. Autom. Sci. Eng. 10 (4) (2014) 853–861.
[29] J.E. Hopcroft, R. Motwani, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, 3rd edition, Addison-Wesley, 2006.
[30] D. Zuckerman, Linear degree extractors and the inapproximability of max clique and chromatic number, Theory Comput. 3 (2007) 103–128.
[31] M.M. Halldorsson, A still better performance guarantee for approximate graph coloring, Inf. Process. Lett. 45 (1993) 19–23.
[32] R. Niedermeier, Invitation to Fixed-Parameter Algorithms, Oxford University Press, New York, 2006.
[33] J. Flum, M. Grohe, Parameterized Complexity Theory, Springer, Berlin Heidelberg New York, 1998.
[34] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 3rd edition, The MIT Press, 2009.
[35] G. Birkhoff, A determinant formula for the number of ways of coloring a map, Ann. Math. 14 (1/4) (1912) 42–46.