Noname manuscript No.
(will be inserted by the editor)

Integer linear programming formulations of the filter partitioning minimization problem

Hazhar Rahmani · Jason M. O'Kane

Received: date / Accepted: date

Abstract Combinatorial filters, which take the form of labelled transition graphs, are a general representation for filtering and inference tasks in robotics. They are of particular interest in contexts where the objective is to minimize the computational resources needed to execute the filter. One specific problem is called the filter minimization (FM) problem, in which the goal is to find, for a given original filter, a state-minimal filter equivalent to the original filter. We consider a special case of FM, called the filter partitioning minimization (FPM) problem, in which the reduced filter must partition the state space of the original filter. This problem has been proven to be NP-hard. This paper considers the practical problem of solving FPM in spite of these hardness results. In contrast to the best known algorithm for this problem, a heuristic approach based on graph coloring proposed by O'Kane and Shell, we show how to convert an FPM instance to an instance of the well-known integer linear programming (ILP) problem. We present three distinct formulations of this reduction. Though ILP is itself a challenging problem, reducing FPM to ILP has the advantage that the ILP problem has been studied in great detail, and highly-optimized solvers are readily available. We describe experiments comparing this approach to the heuristic algorithm of O'Kane and Shell. The results show that the proposed ILP technique performs better in computing exact solutions as the filter sizes grow, and that the ILP approach obtains higher-quality feasible solutions, in contexts where time limitations prohibit the computation of exact solutions.

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

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

1 Introduction

Combinatorial filters, which are formulated as labelled transition graphs, are used for modeling and reasoning about systems in which sensor data is discrete. This kind of filters was originally proposed by LaValle [15, 16] for reasoning about the information requirements of problems in robotics.

Figure 1 shows an example of how such filters might be used in the context of mobile robotics. Suppose a mobile robot navigates through its environment guided by a topological map in the form of a generalized Voronoi diagram (GVG) [24, 30]. See Figure 1a. The robot is equipped with controllers [6] to move reliably through the environment between the junction points of this map. At each junction point, the robot can sense the relative directions of the outgoing corridors, and select one of them to travel along next. The operation of this robot can be expressed as a combinatorial filter, that is, a labeled directed graph in which states are labeled with the filter's output and edges are labeled with observations received by the robot. Each state of the filter corresponds to one of the junction points of the map (or, if the robot may be uncertain about its location, a set of possible junctions) combined with a label indicating the direction from which the robot arrived at the junctions. Each state is labeled with the direction of the corridor the robot should travel next. The directed edges correspond to the connectivity between junctions along the environment's corridors and are labeled with the observation the robot should receive at the next junction it encounters. Notice, in particular, that this graph is a model of the robot's behavior in this environment —it describes what the robot will do, rather than merely describing the environment itself. Figure 1b and 1c show a smaller environment and a corresponding filter for that environment, describing behavior in which the robot, starting from an unknown location in the environment, nonetheless navigates to E.

The question we ask is this: How many states are necessary to complete a task represented using this sort of filter? Such questions are relevant not only for minimizing the computational resources needed to execute the filter, but also for understanding the information structure underlying the problem itself. For this example, Figure 1d shows the smallest filter that can replace the naïve filter in Figure 1c and by which the system's task can still be accomplished.

O'Kane and Shell [20] first addressed this problem of automatic reduction of combinatorial filters. Specifically, they considered the algorithmic problem of finding the smallest filter, measured by the number of states, whose behavior is equivalent to a given input filter. They proved that this problem is NP-hard, and also introduced a heuristic algorithm, which forms the minimal filter by merging 'compatible' states. Their algorithm, in fact, solves FPM, a special case of FM in which the reduced filter must partition the state space of the original filter. This algorithm is (apart from naïve brute force) currently the only known algorithm for FPM.

In this paper, we offer an alternative approach to filter partitioning minimization that improves upon the heuristic proposed by O'Kane and Shell. The basic approach is a reduction to integer linear programming (ILP). We

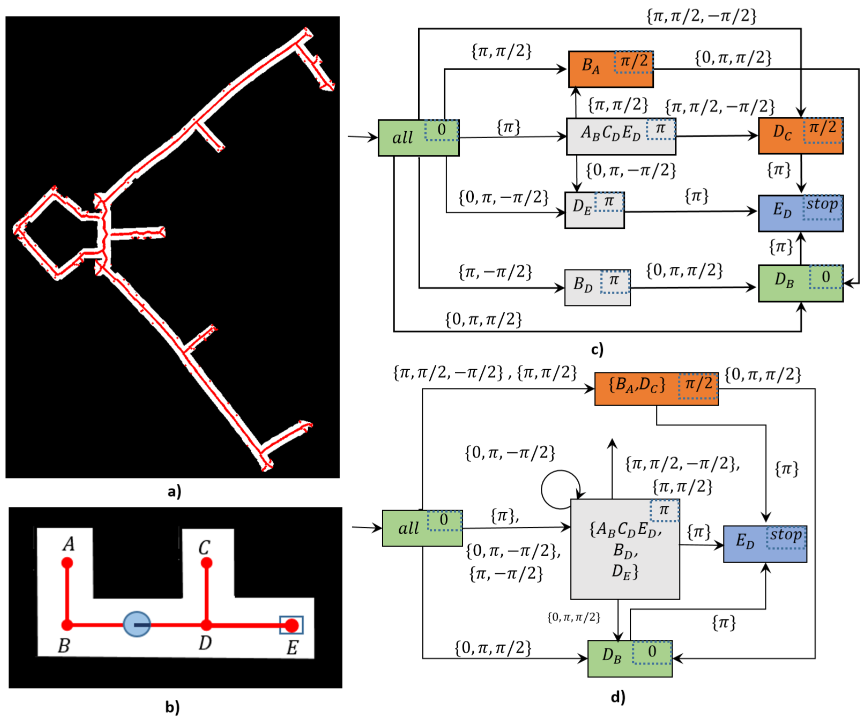


Fig. 1 a) An occupancy grid map of the UofSC Swearingen Engineering Center, generated by a differential drive robot equipped with a Hokuyo laser range finder. The map is overlaid with the generalized Voronoi graph (GVG) of that environment in red. b) A simpler environment and its GVG for illustration purposes. c) A naïvely-constructed combinatorial filter, which the robot can use to navigate, starting from full location uncertainty, to point E. In this example, each vertex in the graph (that is, each state of the filter) is labeled with a set of possible locations for the robot, each written in the form X_Y to mean that the robot has arrived at junction X from junction Y. The states are also labeled with a *color* or *output* from the filter at that state; in this example, the colors are angles (0, $\pi/2$, π, etc), telling which outgoing corridor the robot show follow. The directed edges show how the state changes in response to these movements. The labels on the edges show the information made available to the robot on that transition — in this case, a list of options for which direction to leave the next junction. For example, an edge labeled $\{\pi, \pi/2\}$ means that the robot observes two options for its next movement: It may follow a corridor to its left ($\pi/2$) or it may turn around and return the way it came (π). A robot might execute this plan by an alternating process of executing the movement attached to its current state, using its sensors to acquire a new observation, and transitioning in the graph the state reached by the outgoing edge whose label matches that observation. d) A reduced version of this filter, in which several redundant states are merged. The reduction shows that the information encoded in those different merged states was not, in fact, necessary, for the robot to behave correctly.

consider three different integer linear programming formulations for the filter partitioning minimization problem. None of these three formulations is fully superior to one another, and each might be useful for minimizing certain types of filters.

After related work and basic definitions are reviewed and recalled respectively in Sections 2 and 3, the paper makes two primary contributions: (i) Section 4 presents three integer linear programming formulations for the filter partitioning minimization problem; (ii) Section 5 presents experimental results, which show that the ILP formulation outperforms the algorithm of O'Kane and Shell [21], for both optimal and feasible solutions of the filter partitioning minimization problem. Finally, Section 6 summarizes our conclusions and discusses future work.

2 Related Work

Combinatorial filters [15,16] stem from early work about minimalism in robotics [7, 8]. This approach is based on the idea of identifying minimal configurations of resources such as sensing, actuation, and computation that enable a task to be completed. It has been applied to such tasks as exploration [14], navigation [17, 25, 27], localization [1], target tracking [2, 29], manipulation [13], and story validation [28].

The combinatorial filters considered in this paper are closely related to well-known probabilistic filtering methods such as recursive Bayesian estimation [9, 18] and Kalman filtering [12]. Each of these types of filters are used to estimate a system's internal state over time using the most recent data (observations) received from the sensors. Note in particular that probabilistic filters, when implemented with finite precision, can in principle be expressed as combinatorial filters, in which each state of the filter corresponds to a tuple of concrete values of the variables being estimated and each transition encodes the updates made to that internal state in response to new sensor data. In addition, combinatorial filters are also suitable for other forms of filtering that rely upon combinatorial, rather than probabilistic reasoning. For more details about Bayesian filtering and the Kalman filter, see [5].

Apart from offering the advantage of using less computational resources, finding optimal filters in terms of the number of states also helps us better understand the nature of robotic problems, because minimal filters can reveal the minimal information required to solve those problems.

The problem of automatically constructing optimal filters from given original filters, called *the filter minimization problem*, was first considered by O'Kane and Shell [20]. They proved that the problem is NP-hard, and presented a heuristic algorithm, which solves the problem by merging pairs of 'mergeable' states. Saberifar *et al.* [23] proved that the problem is hard to approximate, even for several reasonable special cases of combinatorial filters. Our own prior work [22] considered reducing filters through making quotient filters under equivalence relations. We proved that the well-known notion of *bisimulation* for those equivalence relations does not always lead to optimally reduced filters. Quite recently, Zhang and Shell [31] introduced a generalization of combinatorial filters and their related minimization problem, which also generalizes FM. They proved that an optimal solution to the FM for some filters cannot be found by partitioning the state space of the filter under an equivalence relation but it always can be found by a covering of the state space, meaning that some states are shared among several ‘classes’ of mergeable states. Accordingly, they proposed an exact solution to FM via a SAT formulation. The differences between our approach and their approach are two: first, the problem we study here is a special case of FM, in which we require the reduced filter to partition the state space of the original filter; and second, the number of constraints of the SAT formulation is exponential to the size of filter, while each of the ILP formulations has a polynomial number of constraints. It is still not known if any SAT or programming formulation of FM can be formed in a polynomial time or not. Because of the first difference, our approach is comparable only with the heuristic algorithm of O’Kane and Shell, which essentially solves FPM. Our approach can be used to compute feasible solutions for large filters for which an optimal solution cannot be computed in a reasonable amount of time. This is because we can efficiently form any of the ILP formulations.

3 Definitions

This section presents basic definitions and notation used throughout the paper. The basic object of study is the combinatorial filter, defined as follows [21]:

Definition 1 A *combinatorial filter*, or simply a *filter*, is a 6-tuple $(V, Y, C, \delta, c, v_0)$ 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,
- $\delta : V \times Y \to V \cup \{\perp\}$ is the *transition function* of filter,
- $c : V \to C$ is a function assigning to each state $v \in V$ a color, and
- $v_0 \in V$ is the initial state.

We depict filters as edge labelled directed graphs, in which the states are shown as vertices and the transition function determines the directed edges. Examples appear in Figure 1 and Figure 2.

The meaning of $\delta(v, y) = \perp$ is that the observation y never occurs when the filter is in state v. Equivalently, in graph view, vertex v has no outgoing edge labeled y. This type of situation occurs when some structure in the problem being modeled prevents that observation from that state.

An observation sequence $s = y_1 y_2 \cdots y_n \in Y^*$, in which each y_i is a member of Y, is said to be *trackable* from $v \in V$ if there is a sequence of states $q_0, q_1, ..., q_n$ such that $q_0 = v$, and $\delta(q_i, y_{i+1}) = q_{i+1}$ for all $0 \leq i < n$. In this situation, we use $\delta^*(v, s)$ to mean the state reached to when s is traced starting from v. If s is not trackable from v, we write $\delta^*(v, s) = \perp$. By convention, the empty string ϵ is trackable for all states v, i.e., $\delta^*(v, \epsilon) = v$.

Additionally, the set of all observation sequences trackable from a state v is called as the language of v, denoted by $L(v)$. Accordingly, *the language* of F, denoted by $L(F)$, is the language of its initial state, i.e., $L(F) = L(v_0)$.

Filter reduction relies on the following definition.

Definition 2 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 and $L \subseteq Y^*$ be an observation language. 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 $s \in L$,

1. $\delta_1^*(v_0, s) \neq \bot$,
2. $\delta_2^*(w_0, s) \neq \bot$, and
3. $c_1(\delta_1^*(v_0, s)) = c_2(\delta_2^*(w_0, s))$.

This definition requires that $L \subseteq L(F_1)$ and $L \subseteq L(F_2)$, and that for any observation sequence in L, F_1 and F_2 must produce the same output. Note that this definition does not require that $L(F_1)$ to be equal to $L(F_2)$. We also require that the state space of the minimized filter has a special property in the sense of the following definition.

Definition 3 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. Denoted $F_1 \doteq F_2$, we say that F_2 partitions the state space of F_1 if for each $v \in V_1$, there is a single state $w \in F_2$ such that for any observation sequence $s \in Y^*$, if $\delta_1^*(v_0, s) = v$, then $\delta_2^*(w_0, s) = w$.

The filter partitioning minimization problem, on which this work elaborates, is defined thusly:

Problem: Filter partitioning minimization (FPM)

Input: A filter F.

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

Informally, the intuition of filter partitioning minimization is to produce for a given filter F, a state minimal filter F^* that traces all observation sequences of $L(F)$ while producing for each of them, the same output produced by the original filter F. Since this minimized filter F^* would produce the same outputs as F for any observation sequence trackable by F and that F^* partitions the state space of F, we can view F^* as a sort of optimal replacement for F with any state in F^* plays the role of one or more states in F. Note that FM is similar to FPM but it lacks the condition $F \doteq F^*$ in its output. The next section casts this problem as an integer linear programming problem with three different formulations.

4 ILP formulations of the FPM problem

In this section, we introduce three integer linear programming models for filter partitioning minimization. To do so, we first review how filter reduction relates to quotient operations.

4.1 Filter reduction as a quotient operation

The idea of these formulations stems from our previous work [22], in which we proved that an optimal filter can formed by taking the *quotient* of the original filter under some equivalence relation over the state space of the original filter. Assuming that one has the correct relation, for each equivalence class of that relation, all of the states in that class are merged to form a single state in the reduced filter.

In our prior work, we provide conditions on the relation under which the quotient operation produces a well-defined filter. Specifically, the relation must be a *compatibility relation*, defined as follows:

Definition 4 Let $F = (V, Y, C, \delta, c, v)$ be a filter. We say that a relation $R \subseteq V \times V$ is a *compatibility relation* for F, if for any $(v, w) \in R$:

1. $c(v) = c(w)$, and
2. for any $y \in Y$, if $\delta(v, y) \neq \bot$ and $\delta(w, y) \neq \bot$, then $(\delta(v, y), \delta(w, y)) \in R$.

We say state v is compatible with w if there exists a compatibility relation R for F such that $(v, w) \in R$. The set of all compatible pairs for a given filter F is denoted by $\curlywedge_F$, which is the union of all compatibility relations for F and itself is a compatibility relation for F. For a simple efficient algorithm computing $\curlywedge_F$, see [22].

To illustrate, consider filter F_1 in Figure 2. Some compatibility relations for this filter are $R_1 = \emptyset$, $R_2 = I_{F_1}$, $R_3 = \{(4, 5), (7, 7)\}$, $R_4 = \{(2, 3), (0, 1), (3, 4)\}$, and $R_5 = I_{F_1} \cup \{(2, 3), (3, 2), (0, 1), (1, 0), (3, 4), (4, 3), (4, 5), (5, 4)\}$, where I_{F_1} denotes the identity relation on the state space of F_1. Observe that $\curlywedge_{F_1} = R_5$.

Given a relation that is both a compatibility relation and an equivalence relation, we can form a quotient filter, which merges equivalent states:

Definition 5 For a filter $F = (V, Y, C, \delta, c, v_0)$, and a relation $R \subseteq V \times V$ that is both a compatibility relation and an equivalence relation (a compatibility equivalence relation), the *quotient of F under R* is the filter $F/R = (V/R, Y, C, \delta', c', [v_0]_R)$, in which

$$\delta'([v]_R, y) = \begin{cases} [\delta(w, y)]_R & \text{if } \exists w \in [v]_R \text{ with } \delta(w, y) \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

and $c'([v]_R) = c(v)$.

Notice that Definition 4 ensures that if two states v and w, both of which have an outgoing edge labeled by an observation y, are merged then their 'y-successors' —$\delta(v, y)$ and $\delta(w, y)$— must also be merged.

The following result holds the key to using this quotient operation to reduce filters.

Lemma 1 *[22] For any filter $F = (V, Y, C, \delta, c, v_0)$, and any compatibility equivalence relation R for F, we have $F \overset{L(F)}{=\!=} F/R$ and that $F_1 \mathrel{\mathring{=}} F_2$.*

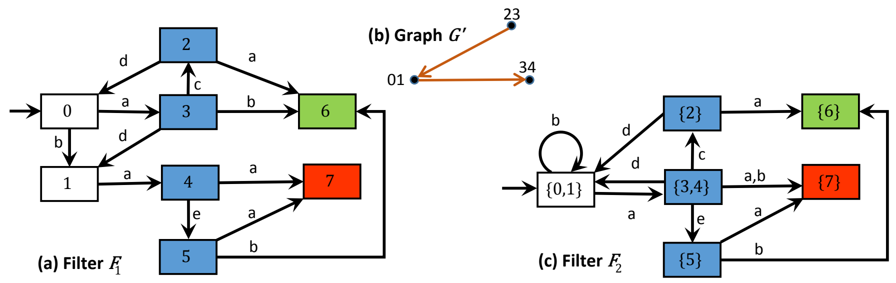


Fig. 2 **a)** Filter F_1. States in the left column have color 1, states in the middle column have color 2, state 6 has color 3, and state 7 has color 4. **b)** The compatibility enforcement graph of filter F_1. **c)** A minimal filter equivalent to F_1.

This upshot is that, if the relation R in Lemma 1 has the minimum number of equivalence classes, then it is guaranteed that F/R is a minimal filter equivalent to F. As a result, the FPM problem is reduced to the following problem:

Problem: Minimum-partition compatibility equivalence relation (MPCER)

Input: A filter F.
Output: A compatibility equivalence relation for F with a minimum number of equivalence classes.

Note that, due to Lemma 1, any feasible solution to the MPCER problem identifies a feasible solution to the FPM problem. Thus, even if we can find only a feasible (rather than optimal) solution to MPCER, we can still use that feasible solution to find a feasible solution to the corresponding FPM instance. The ILP models introduced below are constructed by leveraging this connection between FPM and MPCER.

4.2 Assignment-based ILP

Our first formulation of FPM as an ILP is inspired by the classical ILP for the graph coloring problem [11, 19]. To describe this formulation, we first describe how to cast the problem as a mathematical program that happens to contain some nonlinear constraints. Then we show how to linearize those constraints to form an ILP and describe some simple optimizations that reduce the complexity of the program.

4.2.1 Nonlinear optimization formulation

In this approach, each state of F is assigned to a label from a set of $n = |V|$ labels $1, \ldots, n$. These labels form an equivalence relation on V by relating pair of states that are assigned to the same label. Considering this, in the formulation, for any state v and an integer $1 \leq j \leq n$, a binary variable x_{vj} is introduced. This variable receives value 1 if state v is assigned to label j, and it receives 0 otherwise. Furthermore, n binary variables $p_1, p_2, \ldots, p_n$ are introduced, where for each integer $1 \leq j \leq n$, variable p_j receives value 1 if label j is used for some state in the assignment, or it receives 0 otherwise. Accordingly, the MPCER problem with input F can be solved via the following (nonlinear) mathematical programming model.

Minimize:

$$\sum_{j=1}^{n} p_j \tag{1}$$

Subject to:

– For all $v \in V$,

$$\sum_{j=1}^{n} x_{vj} = 1. \tag{2}$$

– For all $j \in \{1, \ldots n\}$ and all $v, w \in V$ such that $v \not\curlywedge_F w$,

$$x_{vj} + x_{wj} \leq p_j. \tag{3}$$

– For all $v, w \in V$ and all $y \in Y$ such that $\delta(v, y) \neq \perp$ and $\delta(w, y) \neq \perp$,

$$\sum_{j=1}^{n} x_{vj} x_{wj} \leq \sum_{k=1}^{n} x_{\delta(v,y)k} x_{\delta(w,y)k}. \tag{4}$$

– For all $v \in V$ and all $j \in \{1, \ldots, n\}$,

$$p_j \in \{0, 1\} \text{ and } x_{vj} \in \{0, 1\}. \tag{5}$$

The objective function of this model minimizes the number of labels that are used. That is, it minimizes the size of the partition specified by the assignment. Observe that constraints of type (2) ensure that each state of the filter is assigned to one and only one label.

Thus, to see that an optimal solution to this program would yield an optimally reduced filter, we need only to verify that the relation induced by the x_{vi} variables is indeed a compatibility equivalence relation. We write R to denote this relation, defined as

$$R = \{(v, w) \mid x_{v,j} = x_{w,j} = 1 \text{ for some } 1 \leq j \leq n\}. \tag{6}$$

The fact that R is an equivalence relation follows directly from this definition.

Constraints of types (3) and (4) together guarantee that R must be a compatibility relation in the sense of Definition 4:

– By the constraints of type (3), if two states v and w are not compatible, then to any label j, at most one of the states v or w is assigned. Therefore, it prevents states x and y from being related by R. Thus, $R \subseteq \curlywedge_F$. But, relation $\curlywedge_F$ does not relate states of different colors, and thus, relation R satisfies the first part of Definition 4.

– The constraints of type (4) ensure that if $x_{vj}x_{wj} = 1$ for a j —that is, if both $x_{vj} = 1$ and $x_{wj} = 1$— then $x_{\delta(v,y)k}x_{\delta(w,y)k} = 1$ for some k. This means that if v and w are merged (related by R), then $\delta(v, y)$ and $\delta(w, y)$ must also be merged. Thus, relation R satisfies the second part of Definition 4.

We conclude that a solution to this mathematical program does indeed induce a compatibility equivalence relation R with a minimal number of equivalence classes. Therefore, that relation can be used to solve MPCER, and therefore, FPM.

4.2.2 Linearizing the constraints

This model has all the properties to be an integer linear program except that constraints of type (4) are not linear. To linearize these constraints, we introduce for each pair of states $v, w \in V$, two binary variables α_{vw} and β_{vw}, the values of which are constrained as follows:

– For all $v, w \in V$ and all $j \in \{1, \dots, n\}$,

$$\alpha_{vw} \geq x_{vj} + x_{wj} - 1, \tag{7}$$

$$\beta_{vw} \geq x_{vj} - x_{wj}, \tag{8}$$

$$\alpha_{vw} + \beta_{vw} = 1, \text{ and} \tag{9}$$

$$\alpha_{vw}, \beta_{vw} \in \{0, 1\}. \tag{10}$$

According to these constraints, the variable α_{vw} becomes 1 only when $x_{vj} = x_{wj} = 1$ for a j. In this case, β_{vw} receives 0. Consider that if for no j it holds that $x_{vj} = x_{wj} = 1$, then no constraint of type (7) enforces α_{vw} to receive value 0 since in this case the only restrictions on α_{vw} are $\alpha_{vw} \geq 0$ and $\alpha_{vw} \geq -1$, not preventing α_{vw} from receiving 1. In this case, however, it is guaranteed that β_{vw} gets value 1 by a constraint of type (8) because for a j it holds that $x_{vj} = 1$ and $x_{wj} = 0$. Subsequently, constraint (9) makes α_{vw} receive 0.

Knowing that for each state pair v, w, the variable α_{vw} takes value 1 when and only when v and w are chosen to be merged, we can replace the inequality $\sum_{j=1}^{n} x_{vj}x_{wj} \leq \sum_{k=1}^{n} x_{\delta(v,y)k}x_{\delta(w,y)k}$ with the following inequality:

$$\alpha_{vw} \leq \alpha_{\delta(v,y)\delta(w,y)} \tag{11}$$

After these changes, the original formulation becomes a ILP.

4.2.3 Optimizing the ILP

Though we now have a correct ILP for FPM, there are several straightforward changes that can make that program more efficiently solvable.

First, observe that the current formulation introduces two variables α_{vw} and β_{vw} for each pair of states v and w, whether v and w can be merged or not.

However, if we know that two states can never be merged, or even if merged, they do not enforce via the second condition of Definition 4 any other pairs to be merged, then we do not need to introduce this kind of extra variables for them, and by doing so, we may help the solver to eliminate a considerable amount of computations.

To identify pairs (v, w) for which we require variables α_{vw} and β_{vw}, we first construct an auxiliary graph, denoted by $G' = (V', E')$, which we call *the compatibility enforcement graph* for F. To construct that graph, we set $V' = \{vw \mid v \neq w \text{ and } v \curlywedge_F w\}$. Then, for each pair of distinct vertices $vw, rz \in V'$, if for some $y \in Y$ it holds that $\delta(v, y) = r$ and $\delta(w, y) = z$, then we add edge (vw, rz) to V'. Finally, we remove the isolated vertices from V'.

The vertices of this graph, V', are the pairs (v, w) for which we require variables α_{vw} and β_{vw}. An edge (vw, rz) of this graph means that in making a smaller filter, if states v is merged with state w, then state r must also be merged with z. More precisely, if $(v, w) \in R$, then it must hold that $(r, z) \in R$.

To illustrate, consider again filter F_1 in Figure 2. The compatibility enforcement graph of this filter is shown in Figure 2b. Notice that although states 4 and 5 are compatible, the graph does not have a vertex 45 since even if they are merged they do not enforce any other pair of states to be merged through the second condition of Definition 4.

Second, we add two additional types of constraints (21 and 22 below) are intended to reduce symmetry, as suggested by Méndez-Díaz and Zabala [11]. The final assignment-based ILP, combining each of these elements appears below.

Minimize:

$$\sum_{j=1}^{n} p_j \tag{12}$$

Subject to:

- For all $v \in V$,

$$\sum_{j=1}^{n} x_{vj} = 1 \tag{13}$$

- For all $v, w \in V$ such that $u \not\curlywedge_F w$ and for all $j \in \{1, \ldots, n\}$,

$$x_{vj} + x_{wj} \leq p_j. \tag{14}$$

- For all $v, w \in V$ such that $vw \in V'$ and for all $j \in \{1, \ldots, n\}$,

$$\alpha_{vw} \geq x_{vj} + x_{wj} - 1, \tag{15}$$

$$\beta_{vw} \geq x_{vj} - x_{wj}. \tag{16}$$

- For all $v, w \in V$ such that $vw \in V'$,

$$\alpha_{vw} + \beta_{vw} = 1. \tag{17}$$

- For all $v, w, r, z \in V$ such that $(vw, rz) \in E'$,

$$\alpha_{vw} \leq \alpha_{rz}. \tag{18}$$

– For all $v \in V$ and for all all $j \in \{1, \ldots, n\}$,

$$x_{vj}, p_j \in \{0, 1\} \tag{19}$$

– For all $v, w \in V$ such that $vw \in V'$

$$\alpha_{vw}, \beta_{vw} \in \{0, 1\} \tag{20}$$

– For all $j \in \{1, \ldots, n\}$

$$p_j \leq \sum_{v \in V} x_{vj}. \tag{21}$$

– For all $j \in \{2, \ldots, n\}$

$$p_j \leq p_{j-1}. \tag{22}$$

This formulation has $|V|^2 + |V| + 2|V'|$ variables— $|V|^2$ variables for x's, $|V|$ variables for p's, $|V'|$ variables for α's, and $|V'|$ variables for β's.

4.3 Representative ILP

Our second approach uses ideas based on those of Campêlo *et al.* [3,4]. Observe that to make an equivalence relation on the state space of a filter F, we can choose among the states, a set of distinct representatives so that each equivalence class be represented by a representative and then assign each state of the filter to a single representative (in the case that a state is chosen to be a representative, then it can be assigned only to itself). Also consider that one necessary condition for that equivalence relation to be a compatibility relation is that a state cannot be assigned to a representative with which is not compatible. More precisely, any state $v \in V$ can be represented only by those states that are in $S(v)$ where $S(v) = \{u \mid (u, v) \in \curlywedge_F\}$. Observe that v itself is in $S(v)$. Given these, in the representative ILP formulation of FPM problem, for any state v and any state $u \in S(v)$, a binary variable x_{uv} is defined. This variable receives 1 if v is represented by u, and receives 0 otherwise. Moreover, similar to the assignment-based formulation, for each state pair $v, w \in V$ such that $vw \in V'$, two binary variables α_{vw} and β_{vw} are introduced. If v and w are assigned to the same representative, then variable α_{vw} receives value 1, while β_{vw} receives value 0. Otherwise, α_{vw} receives value 0 and β_{vw} receives value 1.

These three kinds of variables participate in the representative ILP formulation of the MPCER problem as follows:

Minimize:

$$\sum_{u \in V} x_{uu} \tag{23}$$

Subject to:

– For all $v \in V$,

$$\sum_{u \in V} x_{uv} = 1. \tag{24}$$

– For all $v, w \in V$ such that $u \not\simeq_F w$ and for all $u \in S(v) \cap S(w)$,

$$x_{uv} + x_{uw} \leq x_{uu}. \tag{25}$$

– For all $v, w \in V$ such that $vw \in V'$ and for all $u \in S(v) \cap S(w)$,

$$\alpha_{vw} \geq x_{uv} + x_{uw} - 1, \tag{26}$$

$$\beta_{vw} \geq x_{uv} - x_{uw}. \tag{27}$$

– For all $v, w \in V$ such that $vw \in V'$,

$$\alpha_{vw} + \beta_{vw} = 1. \tag{28}$$

– For all $v, w, r, z \in V$ such that $(vw, rz) \in E'$,

$$\alpha_{vw} \leq \alpha_{rz}. \tag{29}$$

– For all $v \in V$ and for all all $u \in S(v)$,

$$x_{uv} \in \{0, 1\} \tag{30}$$

– For all $v, w \in V$ such that $vw \in V'$

$$\alpha_{vw}, \beta_{vw} \in \{0, 1\} \tag{31}$$

The objective function of this formulation minimizes the number of representatives and thus the number of equivalence classes induced by the solution. While constraints of type (24) all together ensure all states are assigned to representatives and that any state is assigned to exactly on representative, any two *incompatible* states are prevented to be assigned to a single representative by a constraint of type (25). The constraints involving α_{vw} and β_{vw} have a similar meaning they had in their corresponding constraints of the assignment-based ILP. Given the x's part of a solution to this problem, the equivalence relation R induced by x is as follows:

$$R = \{(v, w) \mid \exists u \in S(v) \cap S(w) \text{ s.t. } x_{uv} = x_{uw} = 1\}. \tag{32}$$

4.4 Partial-ordering based ILP

Our third formulation is similar to the approach of Jabrayilov and Mutzel [11], who proposed a partial-ordering based ILP formulation of the graph coloring problem. This approach is similar to the assignment-based approach in that an equivalence relation over V is constructed by relating pairs of states that are assigned the same label among a set of n labels $1, \ldots, n$. The difference, however, is that in this approach, states are assigned labels indirectly (rather than directly) via making a partial order on a set containing all states and all labels. It is assumed that the sequence of labels is linearly ordered, and accordingly, to make that said partial order, one can specify for each given state $v \in V$ and label $j \in \{1, \ldots, n\}$ that if v is greater than j, denoted $v \succ j$, or v is smaller than j, denoted $v \prec j$. Subsequently, for each state v and label $j \in \{1, \ldots, n\}$, two variables g_{jv} and l_{vj} are introduced. Variable g_{jv} receives value 1 if it is assumed that $v \succ j$, and receives value 0 otherwise. In contrast, variable l_{vj} receives value 1 if it is assumed that $v \prec j$, and otherwise it receives 0. Based on these two kind of variables, state v is assigned label j if and only if $g_{j,v} = l_{v,j} = 0$, that is, when v is neither greater nor smaller than j. To see the connection between these two kinds of variables and the variables of assignment-based approach, one can think of $x_{vj} = 1 - (g_{j,v} + l_{v,j})$. Also, in this formulation an arbitrary state $q \in V$ is chosen to assign the largest label used in the assignment.

Finally, the partial-ordering base ILP for the MPCER problem is as follows:

Minimize:

$$1 + \sum_{j=1}^{n} g_{j,q} \tag{33}$$

Subject to:

– For all $v \in V$,

$$l_{v,1} = 0, \tag{34}$$

$$g_{n,v} = 0. \tag{35}$$

– For all $v \in V$ and $j \in \{1, \dots, n-1\}$,

$$g_{j,v} - g_{j+1,v} \geq 0, \tag{36}$$

$$g_{j,v} + l_{v,j+1} = 1, \tag{37}$$

$$g_{j,q} - g_{j,v} \geq 0. \tag{38}$$

– For all $v, w \in V$ such that $v \not\sim w$ and for all $j \in \{1, \dots, n\}$,

$$g_{j,v} + l_{v,j} + g_{j,w} + l_{w,j} \geq 1. \tag{39}$$

– For all $v, w \in V$ such that $vw \in V'$ and for all $j \in \{1, \dots, n\}$,

$$\alpha_{vw} \geq 1 - g_{j,v} - l_{v,j} - g_{j,w} - l_{w,j}, \tag{40}$$

$$\beta_{vw} \geq -g_{j,v} - l_{v,j} + g_{j,w} + l_{w,j}. \tag{41}$$

– For all $v, w \in V$ such that $vw \in V'$,

$$\alpha_{vw} + \beta_{vw} = 1. \tag{42}$$

– For all $v, w, r, z \in V$ such that $(vw, rz) \in E'$,

$$\alpha_{vw} \leq \alpha_{rz}. \tag{43}$$

– For all $v \in V$ and for all all $j \in \{1, \dots, n\}$,

$$g_{v,j}, l_{j,v} \in \{0, 1\} \tag{44}$$

– For all $v, w \in V$ such that $vw \in V'$,

$$\alpha_{vw}, \beta_{vw} \in \{0, 1\} \tag{45}$$

Types (34) and (35) of constraints ensure that no state is less than label 1 and that no state is greater than label n, respectively. Constraints of type (36)) and (37) all together make sure that each state is assigned a label.

Due to constraints of type (36), if a state v is greater than a label $j + 1$, then it must also be greater than label j. By constraint of type (37) it is not possible for a state to be greater than a label j and at the same time to be smaller than label $j + 1$. Constraints of type (39) prevent incompatible states from being assigned a single label. Those constraints that involve α_{vw} and β_{vw} play the same role as in the previous two ILP formulation. The relation induced by a solution to this 0-1 model is as follows:

$$R = \{(v, w) \mid \exists j \text{ s.t. } g_{j,v} = l_{v,j} = g_{j,w} = l_{v,j} = 0\}. \tag{46}$$

5 Experimental Results

Next, we present some experimental results evaluating the performance of these three formulations. The implementation is in Java, using Cplex to solve ILPs, executed on an Ubuntu 16.04 computer with a 3.6GHz processor.

5.1 Experimental filters

We conducted tests using three kinds of filters. For comparison purposes, we choose two of them to be ones on which previous work performed experiments.

The first kind consists of naïve filters for a family of the *single-agent-donut* problem (Figure 3) by varying the number n of regions. This family of filters was originally studied in [20].

The first part of this figure shows n beam sensors, numbered $1, \ldots, n$, that partition a donut-shaped environment into n regions. In this environment, an agent moves in an unpredictable but continuous path. When the agent passes a beam sensor, the system can identify which one of the n beam sensors it was, but cannot detect if the crossing was clockwise or otherwise. The task is to make an alarm when it is completely sure that the agent is in region 1. For this problem, the system can use a naïve filter, each state of which indicates a set of possible regions in which the agent can be. Accordingly, the initial state is the set of all regions $1, 2, \ldots, n$. For each state, we can draw an outgoing edge whose label is a beam the system senses, and that edge goes to another state whose set of regions are obtained by filtering in the source state, the set of regions the robot can be.

For any $n \geq 3$, the naïve filter that solves an instance of the single-agent-donut problem with n regions has $2n + 1$ states. To illustrate, see Figure 3b, which shows the naïve filter for the case of $n = 5$. For any $n \geq 3$, the optimal filter has only 5 states, regardless of n. Figure 3c shows this optimal filter.

We also consider a family of *two-agent-donut* problems, which are similar to single-agent-donut problems, but instead of a single agent, there are two agents in the environment. This kind of problem was introduced by Tovar *et al.* [26]. The goal is to determine, at any time, if the two agents are in the same region or not. When an agent crosses a beam, the system can only detect which beam sensors it was, but it can detect neither the direction of crossing nor the identity of that agent.

The third family consists of *randomly generated filters.*

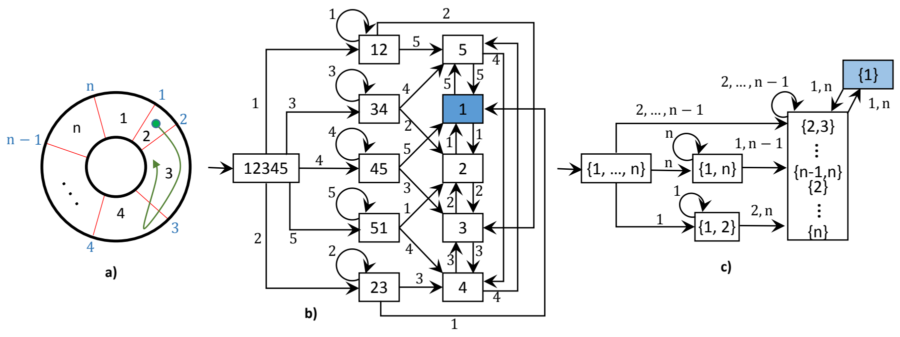


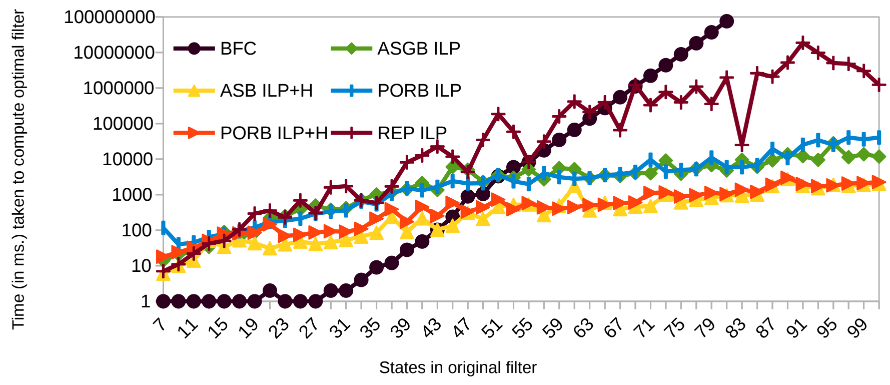
Fig. 3 **(a)** A donut-shaped environment, in which an agent moves within n regions separated by n beam sensors. When the robot crosses a beam, the system knows which sensor it was, but it does not know the direction of that crossing. The task of the system is to determine at any time whether the agent is definitely in region 1 or not. **(b)** A naïve filter which is used to accomplish the task for an instance problem with $n = 5$ regions. **(c)** The smallest filter the system can use to accomplish its task.

5.2 Algorithms to be compared

We compare 6 algorithms. Notice that in the assignment-based ILP and the partial-ordering based ILP we set $n = |V|$. In those two ILPs, the value of $|V|$ is an upper bound for the number of labels used, or more precisely, for the number of states in the reduced filter. Clearly, that upper bound works for any filter. However, for input filters for which we know an upper bound $h < |V|$ on the number of states in the optimally reduced filter, then we can use that upper bound and set $n = h$. By so doing, the number of variables and constraints in the model may be considerably reduced. One way to obtain this sort of h is to compute a feasible solution for the FPM problem using a heuristic, efficient algorithm and then set h to the number of states of the reduced filter. Clearly, the number of states of an optimally reduced filter will be less than or equal to the number of states of a reduced filter computed by that heuristic algorithm. In our experiments, we consider whether applying this kind of bound helps or not.

We considered the following algorithms:

1. BFC: the heuristic algorithm of O'Kane and Shell [21] where conflict graphs are colored by a brute force algorithm.
2. ASGB ILP: the assignment-based ILP (with $n = |V|$).
3. ASGB ILP+H: the assignment-based ILP with $n = h$, where h is a value obtained by the heuristic algorithm of O'Kane and Shell, with conflicts colored greedily.
4. PORB ILP: the partial-ordering based ILP (with $n = |V|$).
5. PORB ILP+H: a variation on PORB ILP using the $n = h$ bound, analogous to ASGB ILP+H.
6. REP ILP: The representative ILP.



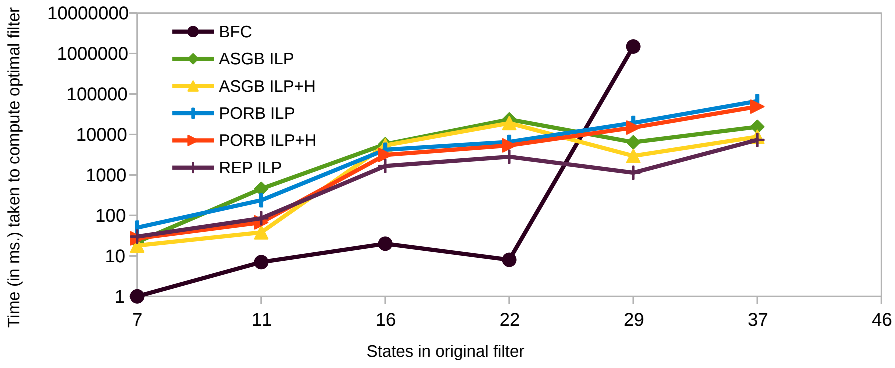


Fig. 4 The performance of the algorithms in computing optimal filters for: **(top)** naïve filters of single-agent-donut instances, **(bottom)** naïve filters of two-agent-donut instances. Notice that the vertical axis in both charts is in logarithmic scale. For each filter size, the experiment was performed for only one filter of that size.

5.3 Computing optimal solutions

In this section, we compare the performance of the six algorithms in computing optimally reduced filters.

5.3.1 Naïve filters of single-agent-donut instances

The chart in Figure 4-top depicts the result of this experiment, which was performed on the naïve filters of instances of single-agent-donut problem where the number of regions varied from 3 to 50 regions. BFC outperforms the ILP based algorithms in computing optimal filters for very small filters, but is rapidly outperformed by the ILP-based algorithms as the original filter sizes grow. We also observed in this experiment that the use of a value obtained by a heuristic algorithm as an upper bound helped ASGB ILP+H and PORB ILP+H to perform better than their corresponding versions without seeds. REP ILP did not perform well for this kind of filters.

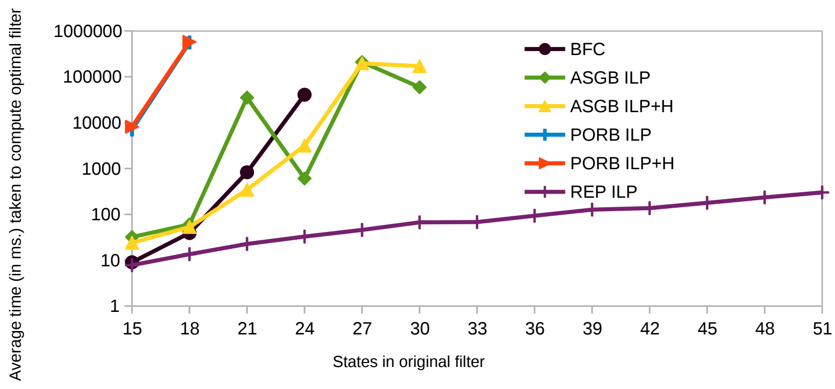


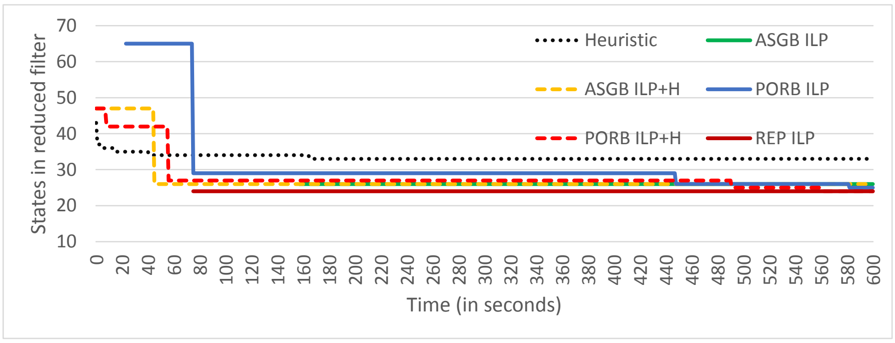
Fig. 5 The performance of the algorithms in computing optimal filters for random filters for which the number of compatible pairs was linear to the number of states of the filter. For each of the filter sizes, the plotted time is the average time to minimize the filter over 10 randomly generated filters of that size. If an algorithm could not find an optimal solution for a filter on a given size in half an hour, then it was considered as a failure, and as a result, the average time is not plotted. Note again that the vertical axis of the chart is in logarithmic scale.

5.3.2 Naïve filters of two-agents-donut instances

Figure 4-bottom shows the results of this experiment. In a time limit of eight hours, BFC could compute optimal filters only for instances with up to seven regions. In the same time limit, ILP based algorithms could make optimal filters for one more instance—an instance with eight regions. Consider that although ILP based algorithm could solve only one more instance, the naïve filter for the instance with 8 regions (which has 37 states) has 8 states more than the naïve filter for 7 regions (which has 29 states).

5.3.3 Random filters with linear number of compatible pairs

In this experiment, we generated filters for which the number of distinct compatible pairs was less than $3|V|$, and the compatibility enforcement graph had both number of vertices and edges around $|V|$. The observation space and color space upon which the filters are constructed has sizes of five and three, respectively. For each state v and an observation y, on the basis of a probability of $1/2$ it was chosen if v must have an outgoing edge labeled y or not, and if yes, then the destination of that outgoing edge was chosen randomly among the states of the filter. Figure 5 compares the performance of the algorithms for this kind of filters. One notable result about this experiment is that the representative ILP considerably outperformed the other algorithms. This is because the REP ILP introduces a variables x_{uv} only when u and v are compatible. More precisely, the number of variables in the REP ILP model was linear to the number of states of the filter, while the number of variables of the other ILPs was quadratic to the number of states of the filter.



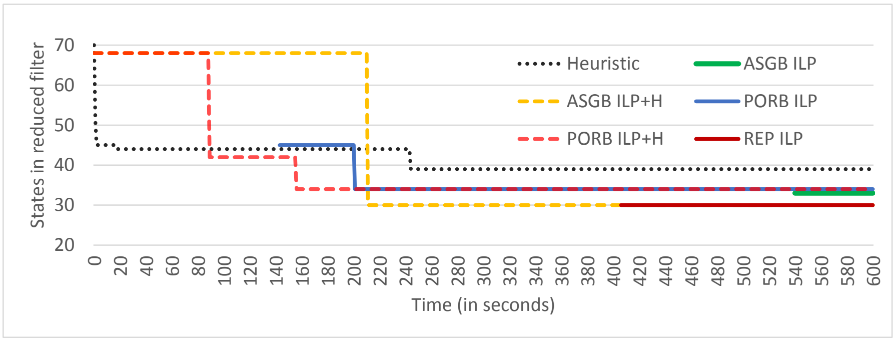


Fig. 6 The trade-off chart of running time vs quality of solution of reduced filters found by the algorithms on **(top)** the naïve filter of two-agents-donut problem with 13 regions **(bottom)** the naïve filter of two-agents-donut problem with 15 regions. The former filter has 92 states, and the later has 121 states.

5.4 Computing smaller filters

This section considers experiments that use ILP models for finding smaller (rather than optimal) filters where optimal filters cannot be computed due to lack of time. Especially, we are interested in trade-off between the time a solver spends and the quality of a solution returned. Figure 6 presents the results of our experiments on naïve filters of two instances of two-agents-donut problem, which are considered difficult to minimize. Notice that each algorithm had 10 minutes to find best solutions it could. The heuristic algorithm with greedy-random coloring was performed as many times as it could during 10 minutes. As an example, this algorithm was performed 34749 rounds to find its best solution on the naïve filter of two-agent-donut with 15 regions. Results shows that although the heuristic algorithm can find good feasible solutions very quickly, we can wait a fairly small amount of time for the ILPs to obtain better solutions than those obtained by the heuristic algorithm.

5.5 Discussion

In this section, we discuss a few observations we made. Our experiments show that none of the three proposed ILP formulations is fully superior to one another and that each one is useful for certain kind of filters. In particular, we observed,

1. from the experiment on random filters (Section 5.3.3 and Figure 5), that the representative ILP formulation is superior to the assignment-based ILP and partial-ordering ILP for filters whose union of all compatibility relations is not dense;
2. from the experiment on optimal filters for single-agent-donut problems (Section 5.3.1 and Figure 4), that the assignment-based ILP and partial-ordering ILP are superior to the representative ILP formulation for filters whose union of all compatibility relations is dense;
3. from the experiment on random filters (Section 5.3.3 and Figure 5), that the assignment-based ILP is superior to the partial-ordering ILP for filters whose union of all compatibility relations is not dense; and
4. from the experiment on computing feasible solutions (Section 5.4 and Figure 6), that for filters that are hard to minimize, the partial-ordering ILP finds feasible solutions faster than the assignment-based ILP.

The first two observations are supported by the fact that the number of variables in the representative ILP model is linear to the number of compatible pairs of states, while the number of variables in the other two ILP models are always quadratic to the number of states of the filter, regardless of the number of compatible pairs of states.

To summarize, our results suggest several general rules of thumb for selecting the most appropriate ILP model.

– For filters that have a large number of colors relative to the number of states, the representative ILP model is likely to outperform the other two ILP models. Such filters might arise, for example, in state estimate settings, where the filter should produce different outputs for each of its internal states. Likewise, if the number of observations is large relative to the number of states, the representative ILP model also seems to outperform the other models. This would be the case, for example, in robotic perception applications in which large amounts of partially redundant data are available.
– For filters whose number of colors and observations are relatively small compared to the number of states of the filter —notably, in cases where the system is using sparse information to reason about long-term dynamics of its environment, such as in some tracking problems— the assignment-based and partial-ordering ILP models tend to fare better. In such cases, our results suggest to use the assignment-based ILP if the number of states of the filters is small, otherwise use the partial-ordering ILP.

6 Conclusions

In this paper, we proposed the use of integer linear programming for automatic reduction of combinatorial filters.

The filter partitioning minimization problem is known to be a difficult problem to compute exact solutions for. Our experiments show that by using the ILP technique, we can compute exact solutions for large filters for which the brute force algorithm is unable to compute exact solutions. Even for large filters for which we cannot compute an exact solution in a reasonable amount time, we can still resort to the ILP technique to compute feasible solutions that are smaller than those computed by the heuristic algorithm of O'Kane and Shell [20]. Those smaller feasible solutions are unlikely to be computed by randomized versions of the algorithm of O'Kane and Shell even if it is executed thousands of times. This is because forming the equivalence class must 'globally' find the 'mergeable' states while the algorithm of O'Kane and Shell finds them 'locally.' In fact, their algorithm iteratively colors a sequence of conflict graphs, at which step of which it is decided with which states, a state must not merged, and that decision forces certain decisions to be made at later steps.

Future work can consider designing metrics to measure the level of difficulty of minimizing a given filter, which can be used for deciding an optimal stopping time and deciding which one of the three ILP formulations is more appropriate to use for a given filter. It can also consider computing strong lower bounds, similar to a recent work by van Hoeve [10], who uses an idea based on decision diagrams for computing lower bounds for the graph coloring problem. Being able to efficiently compute strong lower bounds not only assists accelerate proving optimally, but it also helps decide stopping time for filters that are hard to minimize. Another consideration would be using a 'better' relation than the union of all compatibility relations in the ILP formulations. One possibility would be the *mergeability relation*, a subset of the union of all compatibility relations, which consists of only those pairs of compatible states that are related by at least a compatibility equivalence relation. The mergeability relation will serve a better mean to design a heuristic algorithm on for estimating lower bounds, but experiments are required to see if it will affect the quality of solutions or the performance of the ILPs. Another direction would be to apply machine learning techniques to optimize the use of the current work, especially for computing lower and upper bounds and for learning an appropriate time at which the solver must stop in computing solutions for difficult instances of filters.

Acknowledgements

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

References

1. Alam, T., Bobadilla, L., Shell, D.A.: Space-efficient filters for mobile robot localization from discrete limit cycles. IEEE Robotics and Automation Letters **3**(1), 257–264 (2018)
2. Bobadilla, L., Sanchez, O., Czarnowski, J., LaValle, S.M.: Minimalist multiple target tracking using directional sensor beams. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3101–3107. IEEE (2011)
3. Campêlo, M., Campos, V.A., Corrêa, R.C.: On the asymmetric representatives formulation for the vertex coloring problem. Discrete Applied Mathematics **156**(7), 1097–1111 (2008)
4. Campêlo, M., Corrêa, R., Frota, Y.: Cliques, holes and the vertex coloring polytope. Information Processing Letters **89**(4), 159–164 (2004)
5. Chen, Z., et al.: Bayesian filtering: From Kalman filters to particle filters, and beyond. Statistics **182**(1), 1–69 (2003)
6. Choset, H., Burdick, J.: Sensor based planning. I. the generalized voronoi graph. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 2, pp. 1649–1655. IEEE (1995)
7. Erdmann, M.A., Mason, M.T.: An exploration of sensorless manipulation. IEEE Journal on Robotics and Automation **4**(4), 369–379 (1988)
8. Goldberg, K.Y.: Orienting polygonal parts without sensors. Algorithmica **10**(2), 201–225 (1993)
9. Ho, Y., Lee, R.: A Bayesian approach to problems in stochastic estimation and control. IEEE Transactions on Automatic Control **9**(4), 333–339 (1964)
10. van Hoeve, W.J.: Graph coloring lower bounds from decision diagrams. In: Proceedings of the International Conference on Integer Programming and Combinatorial Optimization, pp. 405–418. Springer (2020)
11. Jabrayilov, A., Mutzel, P.: New integer linear programming models for the vertex coloring problem. In: Proceedings of the Latin American Symposium on Theoretical Informatics, pp. 640–652. Springer (2018)
12. Kalman, R.E.: A new approach to linear filtering and prediction problems. Transaction of the ASME, Journal of Basic Engineering **82**, 34–45 (1960)
13. Kristek, S.M., Shell, D.A.: In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 973–979 (2012)
14. Laguna, G., Murrieta-Cid, R., Becerra, H.M., Lopez-Padilla, R., LaValle, S.M.: Exploration of an unknown environment with a differential drive disc robot. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2527–2533 (2014)
15. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge, U.K. (2006). Available at http://planning.cs.uiuc.edu/
16. LaValle, S.M., et al.: Sensing and filtering: A fresh perspective based on preimages and information spaces. Foundations and Trends® in Robotics **1**(4), 253–372 (2012)
17. Lopez-Padilla, R., Murrieta-Cid, R., LaValle, S.M.: Optimal gap navigation for a disc robot. In: Proceedings of the International Workshop on the Algorithmic Foundations of Robotics, pp. 123–138. Springer (2013)
18. Masreliez, C., Martin, R.: Robust Bayesian estimation for the linear model and robustifying the Kalman filter. IEEE Transactions on Automatic Control **22**(3), 361–371 (1977)
19. Méndez-Díaz, I., Zabala, P.: A cutting plane algorithm for graph coloring. Discrete Applied Mathematics **156**(2), 159–179 (2008)
20. O'Kane, J.M., Shell, D.A.: Automatic reduction of combinatorial filters. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 4082–4089. IEEE (2013)
21. O'Kane, J.M., Shell, D.A.: Concise planning and filtering: hardness and algorithms. IEEE Transactions on Automation Science and Engineering **14**(4), 1666–1681 (2017)
22. Rahmani, H., O'Kane, J.M.: On the relationship between bisimulation and combinatorial filter reduction. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 7314–7321 (2018)
23. Saberifar, F.Z., Mohades, A., Razzazi, M., O'Kane, J.M.: Combinatorial filter reduction: Special cases, approximation, and fixed-parameter tractability. Journal of Computer and System Sciences **85**, 74–92 (2017)

24. Takahashi, O., Schilling, R.J.: Motion planning in a plane using generalized voronoi diagrams. IEEE Transactions on Robotics and Automation **5**(2), 143–150 (1989)
25. Tovar, B.: Minimalist models and methods for visibility-based tasks. University of Illinois at Urbana-Champaign (2009)
26. Tovar, B., Cohen, F., Bobadilla, L., Czarnowski, J., LaValle, S.M.: Combinatorial filters: Sensor beams, obstacles, and possible paths. ACM Transactions on Sensor Networks **10**(3), 1–32 (2014)
27. Tovar, B., Murrieta-Cid, R., LaValle, S.M.: Distance-optimal navigation in an unknown environment without sensing distances. IEEE Transactions on Robotics **23**(3), 506–518 (2007)
28. Yu, J., LaValle, S.M.: Story validation and approximate path inference with a sparse network of heterogeneous sensors. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 4980–4985 (2011)
29. Yu, J., LaValle, S.M.: Shadow information spaces: Combinatorial filters for tracking targets. IEEE Transactions on Robotics **28**(2), 440–456 (2012)
30. Zhang, Q., Rekleitis, I., Dudek, G.: Uncertainty reduction via heuristic search planning on hybrid metric/topological map. In: Proceedings of the 12th Conference on Computer and Robot Vision, pp. 222–229 (2015)
31. Zhang, Y., Shell, D.A.: Cover combinatorial filters and their minimization problem. In: Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (2020)