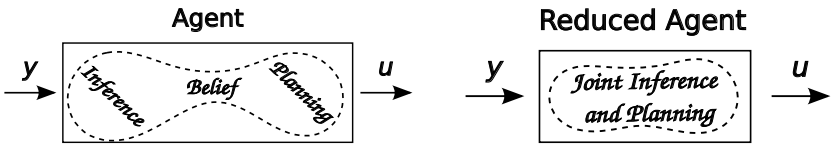


Figure 1: *Representation Reduction* Canonical belief spaces can become a serious informational bottleneck between inference and planning modules. An agent's understanding of its environment need not be any richer than necessary to support the task at hand.

1 Introduction

We are interested in obtaining agents that perform optimally on a task involving uncertainty while being as simple as possible, as measured by the size of their internal representation. While this is a problem relevant in many fields, our motivation comes from the fields of robotics and computer vision. Embodied agents have sensors providing very high-dimensional data; the worlds state is similarly high dimensional. Representing the agents belief about the world is complicated enough, and often inference is intractable. However, it has been observed that, for certain tasks, it is not necessary to represent and plan using the entire belief. Therefore, it is interesting to understand whether it is possible to solve a planning problem using a smaller representation than the entire belief. We will show that it is indeed possible and we will provide a constructive algorithm to obtain representation-minimal agents.

1.1 Previous Work

The topic of reduction of logical functions has been extensively studied. The formalism of finite state machines was developed in the 1950s and 60s, and by 1971, Hopcroft **?** published an $(n \log n)$-time algorithm for reducing completely-specified FSMs. Incompletely-Specified FSMs were studied in turn, and various heuristics were developed to quickly approximate minimum[1] reductions. **???**. Representation reduction in artificial intelligence has been dealt with in various guises: Dimensionality reduction, belief space compression etc. – most heuristics can be considered as implicit hard-coded representation reductions.

1.2 Contributions

This chapter re-casts problem of representation reduction in terms of well-studied computational constructs, and finds absolute minimum reductions in the case of discrete time, discrete input and output. It evaluates three algorithms for approximating minimum reductions, and clarifies their strengths and weaknesses. This work is the result of a collaboration of Andrea Censi, who introduced the formalism of representation reduction in the context of POMDP solvers and proposed the bit-at-a-time algorithm

[1]Here, we distinguish "minimal" from "minimum" reductions. A minimal reduction is one that cannot be further reduced by combining any of its states, whereas a minimum reduction is a minimal reduction with the fewest possible states

2 Formalization

Definition 2.1 (Policies)**.** Given a set $\mathcal{Y}$ of observations, recursively construct

$$\mathcal{C}_0 = \{\emptyset\} \quad \text{and} \quad \mathcal{C}_{i+1} = \{(c, y_{i+1}) : c \in \mathcal{C}_i,\ y_{i+1} \in \mathcal{Y}_c\}, \tag{1}$$

where $\mathcal{Y}_c \subseteq \mathcal{Y}$ are the observations that may be seen in context c. Let $\mathcal{C} = \bigcup_{i=0}^{\infty} \mathcal{C}_i$. A **policy** P is then a tuple $\langle \mathcal{C}, \mathcal{U}, \mathcal{T}, \mathcal{Y} \rangle$, where $\mathcal{U}$ is some decision set and $\mathcal{T} : \mathcal{C} \setminus \{\emptyset\} \to \mathcal{U}$.

Definition 2.2 (Completely-Determined Policies)**.** If $\mathcal{Y} = \bigcup_{\mathcal{C}} \mathcal{Y}_c$ and $\mathcal{C} = \mathcal{Y}^{\leq n}$ for some $n \in \mathbb{N}$, then $P = \langle \mathcal{C}, \mathcal{U}, \mathcal{T}, \mathcal{Y} \rangle$ is **completely determined**. A **completion** of P is a policy $P' = \langle \mathcal{C}', \mathcal{U}', \mathcal{T}', \mathcal{Y} \rangle$ such that

$$\mathcal{Y} \subseteq \mathcal{Y}', \quad \mathcal{C}' = \bigcup_{i=0}^{\infty} (\mathcal{Y}')^i, \quad \mathcal{U} \subseteq \mathcal{U}', \quad \text{and} \quad \mathcal{T}'|_{\mathcal{C}} = \mathcal{T}. \tag{2}$$

Let $\mathrm{Comp}(P)$ be the set of completions of the policy P.

Definition 2.3 (FSM Representations)**.** An **FSM representation** (or just **representation**) is a tuple $\langle \mathcal{C}, \mathcal{R}, \mathcal{U}, \mathcal{S}, \mathcal{T}, \mathcal{Y} \rangle$ (abbreviated to $\langle \mathcal{R}, \mathcal{S} \rangle$ when $P = \langle \mathcal{C}, \mathcal{U}, \mathcal{T}, \mathcal{Y} \rangle$ is given), with "states" $\mathcal{S} \subseteq \mathbb{N}$ and state assignments $\mathcal{R} : \mathcal{C} \to \mathcal{S}$, such that

$$\mathcal{R}(c) = \mathcal{R}(c') \quad \text{and} \quad y \in \mathcal{Y}_c \cap \mathcal{Y}_{c'} \implies \mathcal{T}(c, y) = \mathcal{T}(c', y). \tag{3}$$

Let $\mathrm{Rep}(P)$ be the set of representations of the policy P.

Definition 2.4 (Minimal Representations)**.** The **size** of an FSM representation is the cardinality of its state set. A representation $\langle \mathcal{R}, \mathcal{S} \rangle$ of P is **minimal** if $|\mathcal{S}| = \min\{|\mathcal{S}'| : \langle \mathcal{R}', \mathcal{S}' \rangle \in \mathrm{Rep}(P)\}$. A representation $\langle \mathcal{R}', \mathcal{S}' \rangle$ is a **reduction** of the representation $\langle \mathcal{R}, \mathcal{S} \rangle$ if there is a surjection $\phi : \mathcal{S} \to \mathcal{S}'$ such that $\mathcal{R}' = \phi(\mathcal{R})$.

Example 2.1. If $\mathcal{C} = \{c_1, c_2, \ldots\}$, then we have a canonical representation $\langle \mathcal{R}, \mathcal{S} \rangle$, where

$$\mathcal{S} = \{1, \ldots, |\mathcal{C}|\} \quad \text{and} \quad \mathcal{R} : c_k \mapsto k. \tag{4}$$

It can be shown that the size of a minimal representation of a policy P is equal to the minimum size of the minimal representations of its completions, i.e.

$$\min\{|\mathcal{S}'| : \langle \mathcal{R}', \mathcal{S}' \rangle \in \mathrm{Rep}(P)\} = \min\{|\mathcal{S}'| : \langle \mathcal{R}', \mathcal{S}' \rangle \in \mathrm{Rep}(P'),\ P' \in \mathrm{Comp}(P)\} \tag{5}$$

Incompletely-determined policies allow more freedom in representation reduction, as shown in the next example.

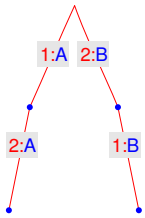
Example 2.2 (Incompletely-Determined Policies)**.** Let $\mathcal{C} = \{\emptyset, (1), (2), (1,2), (2,1)\}$, $\mathcal{U} = \{A, B\}$, and

$$\mathcal{T}(c, y) = \begin{cases} A & c \in \{(1), (1,2)\} \\ B & c \in \{(2), (2,1)\} \end{cases}.$$

Observe that the minimal policy is the same as that of the completely-determined policy in Example **??**.

$\mathcal{T} : \mathcal{C} \setminus \{\emptyset\} \to \mathcal{U}$	
(1)	$\mapsto A$
(2)	$\mapsto B$
$(1,2)$	$\mapsto A$
$(2,1)$	$\mapsto B$

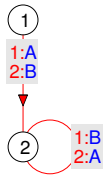
(a) Decision Table



(b) Decision Tree

$\mathcal{R} : \mathcal{C} \to \mathcal{S}$	
$\emptyset$	$\mapsto 1$
(1)	$\mapsto 2$
(2)	$\mapsto 2$
$(1,2)$	$\mapsto 2$
$(2,1)$	$\mapsto 2$

(c) Representation



(d) Minimal Policy

2.1 FSM Reduction

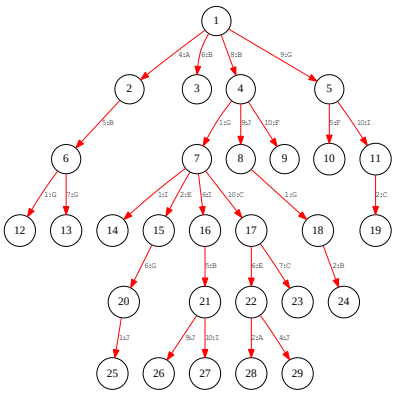
Given a decision policy (or an ISFSM) how do we find an obedient (or equivalent) ISFSM with the smallest possible state set?

for completely-specified FSM, this can be done in $n \log n$ time by Hopcroft's algorithm (Alg. 3.1).

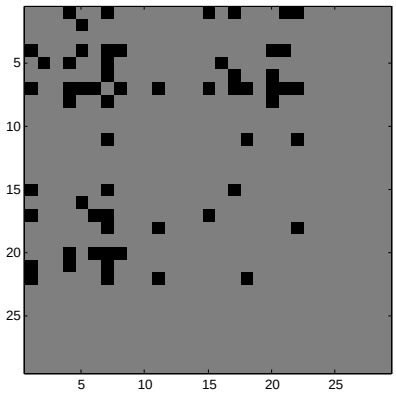
To find a minimum representation of a given policy, we first compute a graph of reducibility relations, then compute a minimal clique-covering.

Cliques on the equivalence graph identify sets of states that can be collapsed into a single state. The minimal clique-covering, that is the smallest collection of disjoint cliques that covers the equivalence graph, correponds to a minimal reduction of the FSM.

3 Representation Reduction Strategies



(a) Canonical Representation

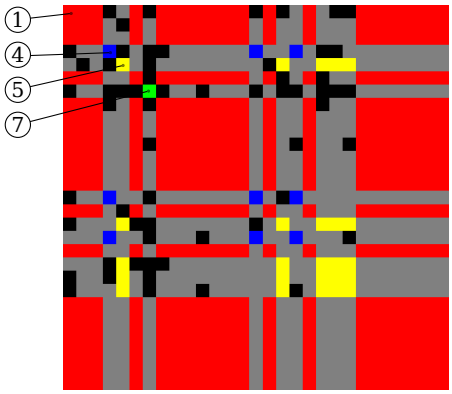


(b) Compatibility Matrix

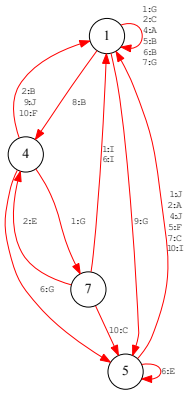
For practical computation of reducibility, we'll start with the weaker condition of compatibility.

3.1 Reducibility Relations

Definition 3.1 (Reducibility)**.** For a given policy $P = \langle \mathcal{C}, \mathcal{U}, \mathcal{T}, \mathcal{Y} \rangle$, two contexts $c_1, c_2 \in \mathcal{C}$ are **reducible** (write $c_1 \sim c_2$) if there exists a representation $\langle \mathcal{R}, \mathcal{S} \rangle$ of P such that $\mathcal{R}(c_1) = \mathcal{R}(c_2)$. Likewise, for a given representation $R = \langle \mathcal{R}, \mathcal{S} \rangle$, two states $s_1, s_2 \in \mathcal{S}$ are **reducible** if there exists a reduction $(\phi, \langle \mathcal{R}', \mathcal{S}' \rangle)$ of R such that $\phi(s_1) = \phi(s_2)$.



(c) Greedy Clique Covering of 3(b)



(d) Reduced Representation

Figure 3: *Greedy Reduction Algorithm* A "running clique" is kept, to which new states are added until all remaining states are incompatible with at least one state in the clique. Then, a new clique is begun. Here, the cliques are $\{1, 2, 3, 6, 8, 9, 10, 11, 12, 13, 14, 16, 19, 22, 26, 27, 28, 29, 30, 31, 32\}$ (red), $\{4, 15, 18\}$ (blue), $\{5, 17, 21, 22, 23\}$ (yellow), and $\{7\}$ (green). (d) shows the resulting policy graph once

Observe that for any representation $\langle \mathcal{C}, \mathcal{R}, \mathcal{U}, \mathcal{S}, \mathcal{T}, \mathcal{Y} \rangle$, the contexts $c_1, c_2 \in \mathcal{C}$ are reducible if and only if the states $\mathcal{R}(c_1)$ and $\mathcal{R}(c_2)$ are reducible. Observe also that for incompletely-determined policies, reducibility is a symmetric but not-necessarily-transitive relation

Example 3.1 (Non-Transitive Reducibility)**.** Suppose $\mathcal{Y} = \{1, 2, 3\}$, $\mathcal{C} = \{\emptyset, (1), (2), (1, 3), (2, 3)\}$, $\mathcal{U} = \{A, B\}$, and

$$\mathcal{T}(c) = \begin{cases} A & c \in \{(1), (1, 3)\} \\ B & c \in \{(2), (2, 3)\} \end{cases}. \tag{6}$$

Observe that, under this policy, $\emptyset \sim (1)$ and $\emptyset \sim (2)$, but $(1) \not\sim (2)$, since $\mathcal{T}(1, 3) \neq \mathcal{T}(2, 3)$.

However, it can be shown that, under a completely-determined policy, reducibility induces an equivalence relation. In either case, we compute reducibility using the following criterion:

Lemma 3.1. Two contexts $c_1, c_2 \in \mathcal{C}$ are reducible iff

$$\mathcal{T}(c_1, s) = \mathcal{T}(c_2, s) \quad \text{for all} \quad s \in \mathcal{Y}^* \quad \text{such that} \quad (c_1, s), (c_2, s) \in \mathcal{C} \tag{7}$$

This informs the following algorithm

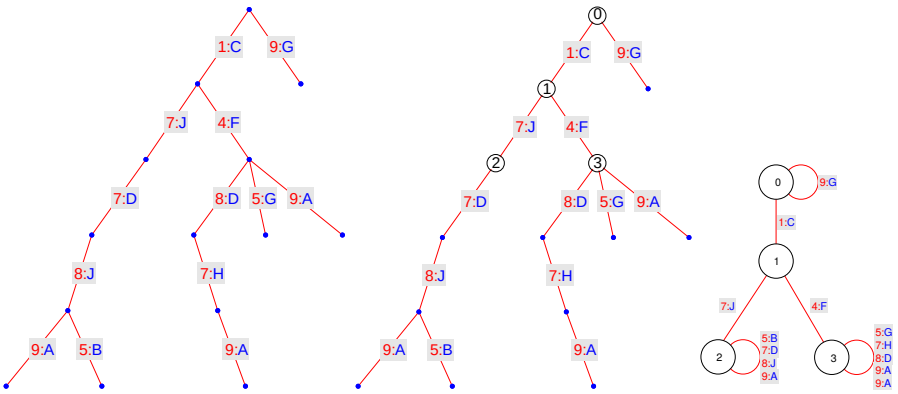
3.2 Bit-at-a-Time

The Bit-at-a-Time reduction, proposed by Andrea Censi, generates a set of states one at a time, separating ambiguous contexts recursively. An ambiguous context is an (input, state) pair for which more than one output is defined. Ambiguities which arise in shorter sequences are separated first.

Algorithm 1: (Hopcroft) Compute Reducibility Relations

Input: A representation $\langle \mathcal{C}, \mathcal{R}, \mathcal{U}, \mathcal{S}, \mathcal{T}, \mathcal{Y} \rangle$
Output: A reducibility matrix $A : \mathcal{S} \times \mathcal{S} \to \{\textbf{true}, \textbf{false}\}$.

$A(s_1, s_2) \Leftarrow \textbf{true}$ for all $s_1, s_2 \in \mathcal{S}$.
repeat
  $isChanged \Leftarrow \textbf{false}$
  for $s_1 < s_2 \in \mathcal{S}$ **do**
    if $A(s_1, s_2) = \textbf{true}$ **then**
      for $c_1 \in \mathcal{R}^{-1}(s_1)$, $c_2 \in \mathcal{R}^{-1}(s_2)$ **do**
        for $y \in \mathcal{Y}_{c_1} \cap \mathcal{Y}_{c_2}$ **do**
          if $\mathcal{T}(c_1, y) \neq \mathcal{T}(c_2, y)$ or $\sim A(\mathcal{R}(c_1, y), \mathcal{R}(c_2, y))$ **then**
            $A(s_1, s_2) \Leftarrow \textbf{false}$.
            $isChanged \Leftarrow \textbf{true}$.
          end if
        end for
      end for
    end if
  end for
until $\sim isChanged$



(a) Decision Tree (b) Partitioned Tree (c) Reduced FSM

Figure 4: *Censi's Bit-at-a-time Algorithm*

3.3 Greedy Covering

Although reducibility is not an equivalence relation, any reduction $\phi : \mathcal{S} \to \mathcal{S}'$ induces an equivalence relation, partitioning $\mathcal{S}$ into cliques of mutually-reducible states, i.e.

$$\mathcal{S} = \bigsqcup_{s' \in \mathcal{S}'} \phi^{-1}(s'), \quad \text{where} \quad \phi(s_1) = \phi(s_2) \implies A(s_1, s_2) \tag{8}$$

Thus, a minimum reduction induces a minimum clique partition of the reducible states of a representation.

3.4 Assembling Cliques

Notation 1 (Arrow notation)**.** For a policy $P = \langle \mathcal{C}, \mathcal{U}, \mathcal{T}, \mathcal{Y} \rangle$, write $c \to c'$ if $c = (c_1, \ldots, c_i) \in \mathcal{C}_i \subseteq \mathcal{C}$ and $c' = (c_1, \ldots, c_i, y) \in \mathcal{C}_{i+1} \subseteq \mathcal{C}$, for some i. For a representation $\langle \mathcal{R}, \mathcal{S} \rangle$ of P, write $s_1 \to s_2$ if there are $c_1 \in f^{-1}(s_1)$ and $c_2 \in f^{-1}(s_2)$ such that $c_1 \to c_2$.

We propose the following, greedy, approximate algorithm In order find the

Algorithm 2: Greedy Clique Covering

Input: A representation $\langle \mathcal{C}, \mathcal{R}, \mathcal{U}, \mathcal{S}, \mathcal{T}, \mathcal{Y} \rangle$ with $s_1 < s_2$ only if $s_2 \not\to s_1$.
Input: A reducibility matrix $A : \mathcal{S} \times \mathcal{S} \to \{\textbf{true}, \textbf{false}\}$ as computed by Algorithm 1.
Output: A partition function $\phi : \mathcal{S} \to \mathcal{S}'$ with $\phi(s_1) = \phi(s_2)$ only if $A(s_1, s_2)$.

```
S' ⇐ S
φ ⇐ id_S
unused ⇐ S
while |unused| > 0 do
  s_1 ⇐ min(unused)
  unused ⇐ unused ∖ {s_1}
  for s_2 ∈ unused do
    if A(s_1, s_2) then
      φ(s_2) ⇐ s_1
      unused ⇐ unused ∖ {s_2}
    end if
  end for
end while
```

absolute minimum representation of a given policy, it suffices to run the greedy algorithm on all possible orderings of its states: Given a minimum clique covering $S = \{s_{c_{11}}, s_{c_{12}}, \ldots, s_{c_{1N_1}}\} \cup \{s_{c_{21}}, s_{c_{22}}, \ldots, s_{c_{2N_2}}\} \cup \cdots \cup \{s_{c_{K1}}, s_{c_{K2}}, \ldots, s_{c_{KN_K}}\}$, feed states to the greedy algorithm in the order in which they are written. Failure to add states to a running clique will occur only once per clique in the minimal covering (exactly K times) [2], so the greedy algorithm will produce a minimum covering. Of course, exhaustively checking every ordering will end up taking exponential time. In fact, it has been shown that the minimum clique cover problem is NP-hard **?**.

[2] If more than K times, then some running clique

3.5 Maximal Anticlique

Alberto and Simão **?** propose a heuristic to increase the likelihood that a greedy algorithm produces a minimum clique covering – First, a maximum anticlique is found in the compatibility graph (This is also an NP-hard problem **?**, but the size of the maximum anticlique generally grows more slowly than the graph itself). Now, each state in the maximum anticlique must belong to a different clique in the minimum clique covering. Also, every remaining state must be compatible with at least one of the states in the anticlique (or else violate the maximality of the anticlique). The greedy algorithm then proceeds, taking first the states in the maximum anticlique, then the remaining states It can easily be shown that an ordering of this type will produce a minimum reduction. The number of such orderings still grows exponentially with the number of states, but in practice it grows significantly more slowly.

3.6 Comparisons

Two types of random FSMs were generated to test the correctness and numerical efficiency of the algorithms described above.

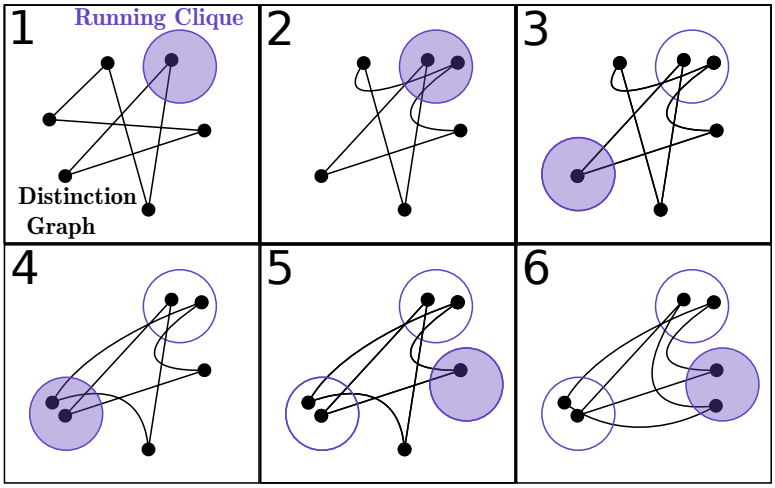
3.6.1 Poisson Random Tree

A Poisson random tree is an decision policy generated by recursively adding $X \sim \text{Poisson}(\lambda)$ children to each new node of an existing policy. We conditioned this result on the outcome that the process not terminate before the tree reaches depth H (contains a sequence of length H or greater). These decision policies are well-studied in decision theory, as they model birth/death processes where individuals continuously produce offspring at a rate of λ per lifetime.

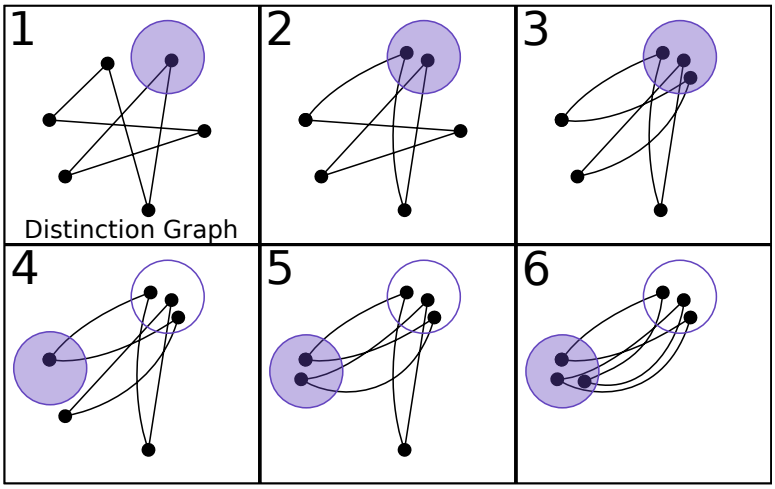
3.6.2 Pathological Tree

The "pathological" tree is a policy of depth 2 with $6n+1$ contexts. was designed to frustrate the algorithm of Alberto and Simão. Each of its states at depth 1 is incompatible with exactly two others. The resulting distinction graph consists of disjoint rings. The order in which states are added to cliques is critical. Half of all orderings result in a three-state FSM, whereas the minimum number of states is two.

4 Results

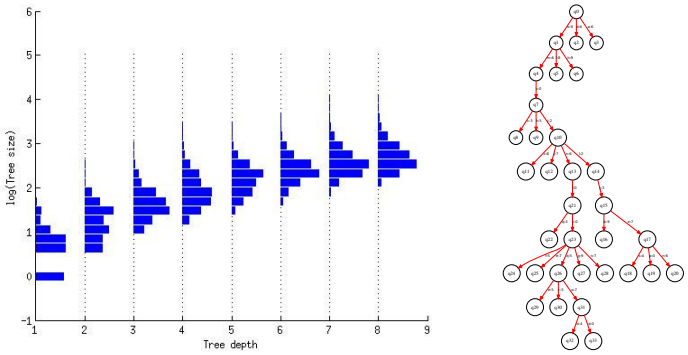


(a) Bad Ordering



(b) Good Ordering

Figure 5: *Greedy Algorithm on Pathological Trees* Here we illustrate the dependence of greedy algorithms on the order of their inputs. In the first set of figures (a), we proceed randomly, greedily adding random states to a running clique until none can be added (no two states sharing an edge in the distinction graph can be part of the same clique). This results in one more clique than is necessary - proceeding counterclockwise, we cover the set with only two cliques. Alberto and Simão's maximum anticlique algorithm fares no better, as the maximal anticlique has size 2.



(a) Poisson Tree Growth Trend (b) Tree, Depth 10

Figure 6: *Poisson Random Tree*

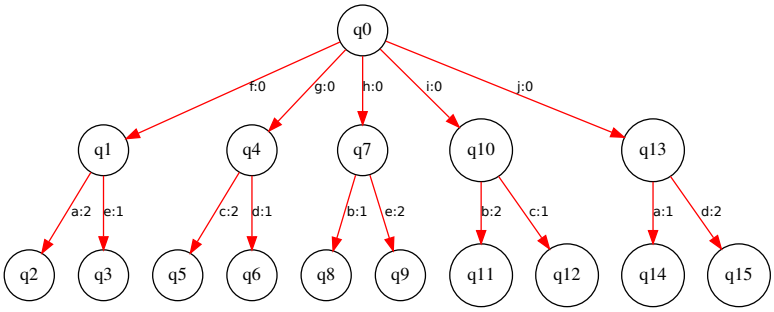
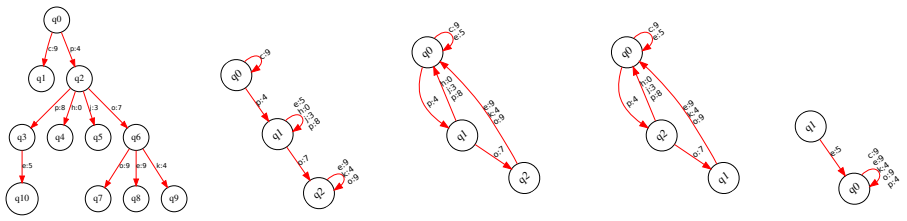
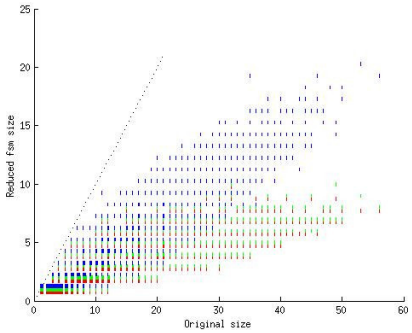


Figure 7: *Pathological Tree*

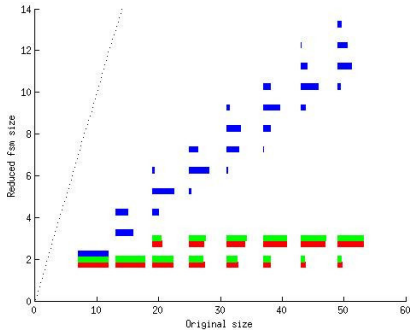


(a) Original (b) Bit-at-a-Time (c) Greedy (d) Alberto-Simão (e) Exact

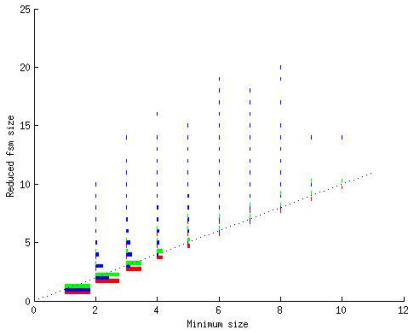
Figure 8: *Typical Reductions* Here we contrast the results of the various reduction algorithms introduced above. The Bit-at-a-Time method (b) produces a minimal sub-tree of the canonical policy. The last three methods are equivalent up to a reordering of states, so reductions (c) and (d) are practically identical.



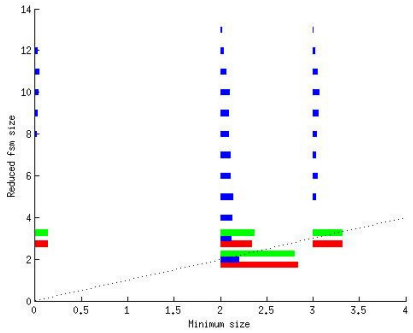
(a) Poisson Trees Reduced vs. Orig. Size



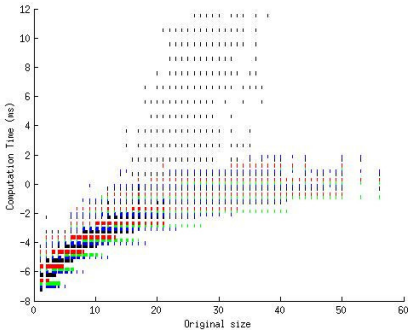
(b) Patho. Trees Reduced vs. Orig. Size



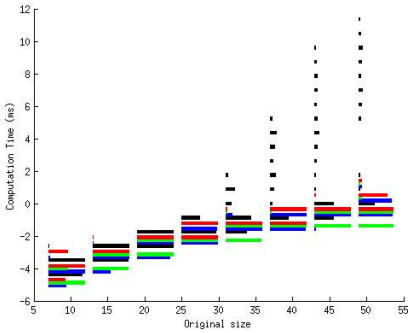
(c) Poisson Trees Reduced vs. Min. Size



(d) Patho. Trees Reduced vs. Min. Size



(e) Algo. Runtimes on Poison Trees



(f) Algo. Runtimes on Patho. Trees

Figure 9: Red bars pertain to Alberto and Simão's method, blue bars pertain to the bit-at-a time method, green bars pertain to the greedy clique completion method, and black bars pertain to an exhaustive search for a minimal reduction.