Approximating the Minimal Sensor Selection for Supervisory Control

Kurt R. Rohloff
Coordinated Science Laboratory
The University of Illinois at Urbana-Champaign
1308 West Main St.
Urbana, IL 61801
USA
krohloff@control.csl.uiuc.edu
black.csl.uiuc.edu/∼krohloff

Samir Khuller
Department of Computer Science
University of Maryland
College Park, MD 20742
USA
samir@cs.umd.edu
www.cs.umd.edu/users/samir

Guy Kortsarz
Computer Science Department
Business and Science Building
Rutgers University
Camden, NJ
USA
guyk@camden.rutgers.edu

April 21, 2005

Abstract

This paper discusses the problem of selecting a set of sensors of minimum cost that can be used for the synthesis of a supervisory controller. It is shown how this sensor selection problem is related to a type of directed graph *st*-cut problem that has not been previously discussed in the literature. Approximation algorithms to solve the sensor selection problem can be used to solve the graph cutting problem and vice-versa. Polynomial time algorithms to find good approximate solutions to either problem most likely do not exist (under certain complexity assumptions), but a time efficient approximation algorithm is shown that solves a special case of these problems. It is also shown how to convert the sensor selection problem into an integer programming problem.

1 Introduction

When a controller operates on a system so that the behavior of the controlled system matches some specification, the controller may not need sensors to observe all behavior in the system. That is, there may be several sets of sensors that could be selected for the controller to use that would be sufficient for the controller to match the specification. Therefore, if there is a cost associated with allowing a controller to use a sensor, then for reasons of economy or simplicity it may be desired that the controller use a set of sensors with the lowest cost possible. However, the lowest cost sensor selection may not always be obvious when designing a controller. This paper discusses this sensor cost minimization problem using the framework of supervisory control theory and discrete-event systems introduced in the seminal works [12, 13].

In the framework of this paper the behavior of the systems and specifications are modeled as finite state automata. Controllers may have sufficient actuation to disable some events but not others as in [12, 13]. Similarly, as alluded to above, a controller may not be able to observe all system events. The framework presented in [10] is used to model the observation behavior of a set of sensors where each sensor is assigned to observe all occurrences of exactly one event and the sensors are deterministic such that they report all occurrences of the events they are designated to sense. Therefore, a given sensor selection for a controller effectively partitions system events into a set events whose occurrences are always observed by the controller and a set of events that are never observed by the controller. Each sensor is assumed to have a non-negative and possibly non-uniform cost of installation to observe its assigned event such that once a sensor is installed there is no extra cost associated with the use of that sensor.

Unfortunately, for the specialized case of uniform sensor cost, the optimal sensor selection problem outlined above is NP-complete ([21]). This means that there is most likely no algorithm that runs in polynomial time and always calculates the minimal cost sensor selection. Fortunately, effective polynomial time approximation algorithms exist for many real-world NP-complete optimization problems ([2]). With this in mind, an approximation of the minimal sensor selection may commonly be sufficient and acceptable for practical use. Therefore, an interesting compromise to designing algorithms to find the minimum cost sensor selection would be to develop methods to approximate the minimal cost sensor selection. Hopefully some bounds could be placed on the closeness of the approximations found this way as not all NP-complete problems have equally effective polynomial-time approximation methods. However, there has been little investigation into the calculation of approximate solutions to many computationally difficult supervisory control problems. Therefore, this paper explores the problem of approximating solutions to the sensor cost minimization problem.

Variations of the sensor selection problem using frameworks similar to the one used in this paper have been investigated in [4, 6, 7, 21, 22]. The problem of designing an observation function that is as coarse as possible is discussed in [4]. A projection mapping is assumed in [4] that is different from the natural projection operation used as the observation function in this paper, and optimization and approximation methods are not discussed in [4]. The optimization of the observable event set is discussed in [6] for achieving both observability and normality for a problem setting very similar to that discussed here. An exponential-time algorithm is shown in [6] for giving an optimal observable set. An algorithm is given in [22] for optimizing the sensor selection set in exponential time along with a polynomial time algorithm for finding exactly one locally minimum sensor selection. An optimal sensor selection problem is also discussed in [7], except the observation function is different from the one assumed in this paper. Preliminary versions of the results presented in this paper are shown in [8].

In the next section the necessary background information from supervisory control and computer science is given. The problem statement of the minimal cost sensor selection problem is formulated in Section 3. The sensor selection problem is related to a type of directed graph *st*-cut problem in Section 4, and some inapproximability results for the sensor selection and graph cutting problems are shown in Section 5. A polynomial time approximation algorithm for a special case of the sensor selection and graph cutting problems is shown in Section 6. Section 7 shows how the minimal cost sensor selection problem can be converted into an integer programming problem. The paper closes with a brief discussion of the results in Section 8.

2 Notational Review

To aid the reader, this section gives a review of necessary concepts of supervisory control and theory of computation. First, the supervisory control models of [10, 12, 13] are presented in Subsection 2.1, and Subsection 2.2 discusses some fundamental results related to the theory of computation. A more indepth review of supervisory control is given in [3] and the theory of computation is more deeply discussed in [2, 5, 20].

2.1 Supervisory Control

In the supervisory control framework systems and specifications are respectively modeled as the automata $G = (X^G, x_0^G, \Sigma, \delta^G, X_m^G)$ and $H = (X^H, x_0^H, \Sigma, \delta^H, X_m^H)$ where X^G and X^H are sets of states, x_o^G and x_o^H are initial states, Σ is the common event set of the automata, $\delta^G : X^G \times \Sigma \rightarrow X^G$ and $\delta^H : X^H \times \Sigma \rightarrow X^H$ are the (possibly partial) state transition functions, and X_m^G and X_m^H are sets of marked states.

Deterministic system and specification automata are exclusively used in this paper. The state transition function can be extended in the usual manner to be defined over strings of events. The notation $x \overset{s}{\mapsto}_B y$ is also sometimes used in this paper to denote that according to the transition rules of a possibly nondeterministic automaton B, there is a path of transitions from x to y labeled by the string s.

The language *generated* by an automata G is the set of strings

$$\mathcal{L}(G) = \{s \in \Sigma^{G*} | \delta^G(x_0^G, s)!\}$$

that are defined in G from the initial state. Note that the unary operator ! for $f(\alpha)!$ returns true if $f(\cdot)$ is defined for input α, false otherwise. The language *marked* by an automata G is the set of strings

$$\mathcal{L}_m(G) = \{s \in \Sigma^{G*} | \delta^G(x_0^G, s) \in X_m^G\}$$

that lead to a marked state from the initial state. The language generated ($\mathcal{L}(G)$) is a prefix-closed language, i.e., it contains all the prefixes of all its strings, but the marked language ($\mathcal{L}_m(G)$) is not prefix-closed in general. For a language K, the language $\overline{K}$ denotes the set of all the prefixes of all the strings in K. An automaton that marks a prefix-closed language is called a *prefix-closed automaton*. An automaton G is said to be *nonblocking* if the prefix-closure of its marked language is equal to its generated language, i.e., $\overline{\mathcal{L}_m(G)} = \mathcal{L}(G)$. Therefore, G is nonblocking if for any string of behavior $s \in \mathcal{L}(G)$ there always exists another string t such that $st \in \mathcal{L}_m(G)$.

Following the modeling formalisms of [10, 12, 13], systems are modeled as finite state automata with supervisory controllers. Controllers may have a set of sensors to observe a set of system events $\Sigma_o \subseteq \Sigma$ with each sensor assigned to deterministically observe all occurrences of exactly one event. Furthermore, on the occurrence of observable events, controllers may be given sufficient actuation to selectively disable a subset of the controllable events $\Sigma_c \subseteq \Sigma$. Controllers can be realized as finite state automata that observe some events and control a potentially different set of events. Controllers should not be able to disable uncontrollable events and control actions should not update on the occurrence of unobservable events.

Given a controller S and a system G, the composed system of S controlling G is denoted as the controlled system S/G. The generated behavior of the controlled system S/G is said to match the generated behavior of a specification H if $\mathcal{L}(S/G) = \mathcal{L}(H)$, and the marked behavior of the controlled system S/G is said to match the marked behavior of a specification H if $\mathcal{L}_m(S/G) = \mathcal{L}_m(H)$. Controller S is said to be nonblocking for system G if S/G is nonblocking, i.e., if $\overline{\mathcal{L}_m(S/G)} = \mathcal{L}(S/G)$. Also, let $\Sigma_{uc} = \Sigma \setminus \Sigma_c$ denote the set of uncontrollable events.

For a given set of observable events $\Sigma_o \subseteq \Sigma$, a natural projection operation $P : \Sigma \rightarrow \Sigma_o$ is used to model a controller's observations of system behavior. For the empty event ϵ, $P(\epsilon) = \epsilon$, and for a string of events s and an event σ,

$$P(s\sigma) = \begin{cases} P(s)\sigma & \text{if } \sigma \in \Sigma_o \\ P(s) & \text{otherwise} \end{cases} .$$

The inverse function $P^{-1} : \Sigma_o^* \rightarrow 2^{\Sigma^*}$ is defined such that

$$P^{-1}(s) = \{t | P(t) = s\}$$

is the set of strings with s as their common projection.

As system behavior progresses and a string of events s is generated by the system, a controller would observe $P(s)$. The controller would then use the observation projection $P(s)$ to estimate the current system state and determine its control action. A controller is said to be *admissible* if it only attempts to disable controllable events and updates its control action only on the occurrence of observable events. See Figure 1 for a schematic of a system G that is controlled by the controller S to match a specification H. In this figure, a string of behavior s is generated by G and $P(s)$ is observed by the controller. After observing $P(s)$, the controller enforces control action $S(P(s))$ on the behavior of G.

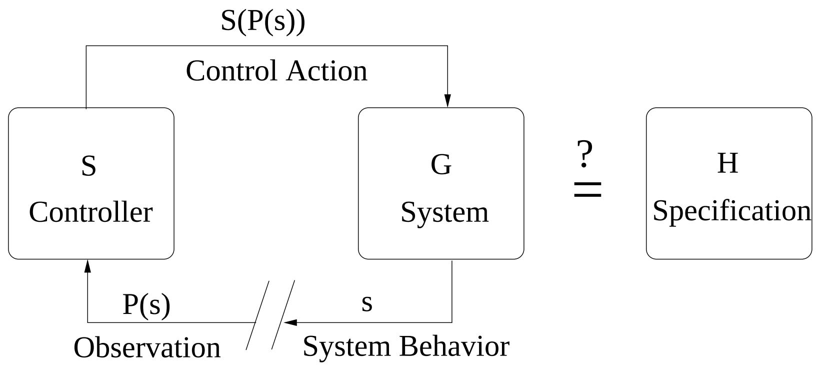

Figure 1: Schematic of a supervisory control system

Three important properties related to controller existence are controllability, observability and $\mathcal{L}_m(G)$-closure.

Definition 1 *[12] Consider the languages K and M such that $M = \overline{M}$ and the set of uncontrollable events Σ_{uc}. The language K is* controllable *with respect to M and Σ_{uc} if*

$$\overline{K}\Sigma_{uc} \cap M \subseteq \overline{K}. \tag{1}$$

The concept of controllability implies that for a set of generated system behaviors M, a set of marked specification behaviors K and for every possible occurrence of illegal behavior, a controller has sufficient actuation to prevent that behavior from occurring. That is, if there is an event σ such that $\overline{K} \subset \overline{K}\sigma \cap M$, then $\sigma \in \Sigma_c$.

Definition 2 *[10] Consider the languages K and M such that $M = \overline{M}$ and the sets of controllable, Σ_c, and observable Σ_o events. The language K is* observable *with respect to M, $P(\cdot)$ and Σ_c if for all $t \in \overline{K}$ and for all $\sigma \in \Sigma_c$,*

$$\left[\left(t\sigma \notin \overline{K}\right) \wedge (t\sigma \in M)\right] \Rightarrow \left(P^{-1}\left[P(t)\right]\sigma \cap \overline{K} = \emptyset\right). \tag{2}$$

The concept of observability captures the notion that for a set of generated system behaviors M and a set of marked specification behaviors K, that for every possible string of behavior $t\sigma \in M$ such that σ is controllable, t is legal, but $t\sigma$ is not, then there must be no control conflict associated with a controller's estimate of disabling σ. That is, $P^{-1}\left[P(t)\right]\sigma$ must not contain a string $t'\sigma$ that is legal but indistinguishable from $t\sigma$ with respect to the sensor selection Σ_o.

Definition 3 *Consider the sets of languages K and M. The set K is* M-closed *if*

$$K = \overline{K} \cap M. \tag{3}$$

The concept of M-closure implies that the generated behavior $\overline{K}$ in a specification is marked if the behavior is marked in the system behavior M.

The above definitions of controllability, M-closure and observability are central to the following controller existence theorem called the controllability and observability theorem.

Theorem 1 *[10] For a finite state automaton system G, a finite state automaton specification H such that $\mathcal{L}_m(H) \subseteq \mathcal{L}_m(G)$, a set of controllable events Σ_c and a set of observable events Σ_o, there exists a partial observation controller S such that $\mathcal{L}_m(S/G) = \mathcal{L}_m(H)$ and $\mathcal{L}(S/G) = \overline{\mathcal{L}_m(H)}$ if and only if the following three conditions hold:*

1. *$\mathcal{L}_m(H)$ is controllable with respect to $\mathcal{L}(G)$ and Σ_{uc}.*

2. *$\mathcal{L}_m(H)$ is observable with respect to $\mathcal{L}(G)$, Σ_o and Σ_c.*

3. *$\mathcal{L}_m(H)$ is $\mathcal{L}_m(G)$-closed.*

For languages generated by deterministic automata, controllability and $\mathcal{L}_m(G)$-closure can be decided in polynomial time using standard automata manipulation operations. There is a construction presented in [19] for deciding the observability of languages generated by deterministic automata in polynomial time. The essence of this method is that a machine $\mathcal{M}$ is constructed from a system G, a specification automata H, the controllable events Σ_c and the observable events Σ_o, such that $\mathcal{L}_m(\mathcal{M}) = \emptyset$ if and only if the $\mathcal{L}_m(H)$ is observable with respect to $\mathcal{L}(G)$, Σ_o and Σ_c.

A less restrictive version of Theorem 1 also holds for the case of generated language specifications (and hence prefix-closed marked language specifications) where the $\mathcal{L}_m(G)$-closure condition is disregarded.

2.2 Theory of Computation

In the field of computation theory a problem instance is said to be a "decision problem" if all problem instances are mapped to be either "true" or "false". In the set of decision problems, a problem is said to be in class P if it can be decided in polynomial time using a *deterministic* computation device and it is said to be in NP if it can be decided in polynomial time using a *nondeterministic* computation device. Although it is not known for sure, it is generally believed that the class NP is distinct from the class P. Therefore, it is believed that not all decision problems can be solved efficiently in time.

In addition to the above classes of decision problems, $DTIME(f(n))$ is the set of all problems that can be solved by *deterministic* algorithms with time complexity in $O(f(n))$. Using standard computer science notation, n is the size of the encoding of the problem instance. Of interest is the class $DTIME(n^{\text{polylog } n})$. It is believed that $NP \nsubseteq DTIME(n^{\text{polylog } n})$, but this has not been proved [1].

Similar to decision problems, there is a set of problems called optimization problems. Each optimization problem has a set of problem instances $\mathcal{P}$, a set of feasible solutions $\mathcal{F}_p$ for a problem instance $p \in \mathcal{P}$ and a cost function $cost_p : \mathcal{F}_p \rightarrow \Re$ that maps the set of feasible solutions of a problem instance to a real value that is a measure of the desirability of that solution. The solution to an instance of an optimization problem is the minimal cost solution for that problem instance, i.e., $F(p) \in \mathcal{F}_p$ such that $\forall f \in \mathcal{F}_p, cost_p(F(p)) \leq cost_p(f)$.

The set of feasible solutions to the optimization problem may be finite, countably infinite or a subset of the real numbers. Similar to decision problems, there are the PO and NPO optimization problem classes where an optimization problem is said to be in PO if an optimal solution can be calculated in polynomial time using a *deterministic* computation device and it is said to be in NPO if an optimal solution can be computed in polynomial time using a *nondeterministic* computation device.

There are several important optimization problems in NPO that are believed to not be in PO. Please see the compendium in [2] for an extensive listing of these problems. However, even though some optimization problems can most likely not be solved in polynomial time, it may still be possible to reasonably calculate approximate solutions to these problems in polynomial time with reasonably good bounds on the performance of the approximation. Naturally, the solutions to some problems may be more difficult to approximate than the solutions to other problems. The concept of an *r-approximation*, shown in Definition 4 below captures this property.

Definition 4 *[2] For a problem instance $p \in \mathcal{P}$ of an optimization problem, let $\mathcal{A}$ be an algorithm that such that when given p as input, $\mathcal{A}$ returns an approximate solution $\mathcal{A}(p) \in \mathcal{F}_p$ for that problem instance. The approximation algorithm $\mathcal{A}(p)$ is* r-approximate *if*

$$\forall p \in \mathcal{P} \left(\frac{cost(\mathcal{A}(p))}{cost(F(p))} \leq r \right). \tag{4}$$

Conceptually, the r in an r-approximation is the maximum known ratio between the cost of the optimal solution $cost(F(p))$ and the approximation found by an algorithm, $\mathcal{A}(p)$. The ratio r may be some function on the size of the problem instance.

3 The Sensor Selection Problem

Before the sensor selection problems are introduced some important concepts are defined.

Definition 5 *A set $\Sigma_o \subseteq \Sigma$ is called a* sufficient sensor selection *with respect to G, H and Σ_c if $\mathcal{L}(H)$ is observable with respect to $\mathcal{L}(G)$, Σ_o and Σ_c.*

Note that with Definition 5 and Theorem 1, if Σ_o is a sufficient sensor selection and $\mathcal{L}(H)$ is controllable with respect to $\mathcal{L}(G)$ and Σ_c, then there exists an admissible controller S such that $\mathcal{L}(S/G) = \mathcal{L}(H)$. It is generally assumed in this paper that $\mathcal{L}(H)$ is always controllable with respect to $\mathcal{L}(G)$ and Σ_c.

It may be possible that there is a positive cost function $cost : \Sigma \to \Re^+ \cup \{0\}$ such that $cost(\sigma)$ represents the cost associated with a controller being able to observe occurrences of the event σ in the system. This cost function can be extended in the usual manner to be defined over sets of events. Formally, for a set of events $\Sigma_o \subseteq \Sigma$ define

$$cost(\Sigma_o) = \sum_{\sigma \in \Sigma_o} cost(\sigma). \tag{5}$$

When given a system G, a specification H and a set of controllable events Σ_c, it may be desired to find the lowest cost sufficient sensor selection Σ_o in order to ensure that there exists a controller S such that $\mathcal{L}(S/G) = \mathcal{L}(H)$. This prompts the formal definition of the minimal cost sufficient sensor selection problem.

Problem 1 Minimal Cost Sensor Selection: *Given G, H, $\Sigma_c \subseteq \Sigma$ and a cost function cost* $: \Sigma \rightarrow \Re^+ \cup \{0\}$*, find a sufficient sensor selection* Σ_o^{min} *such that for any other sufficient sensor selection* Σ_o, $cost(\Sigma_o^{min}) \leq cost(\Sigma_o)$.

A simple example of the sensor selection problem is now shown.

Example 1 *Consider the system and specification seen in Figure 2. Suppose that* $\Sigma_c = \{\alpha\}$*. There are several sufficient sensor selections for this specification with respect to the given system:* $\{\alpha\}, \{\beta,\gamma\}, \{\gamma,\lambda\}, \{\beta,\lambda\}$.

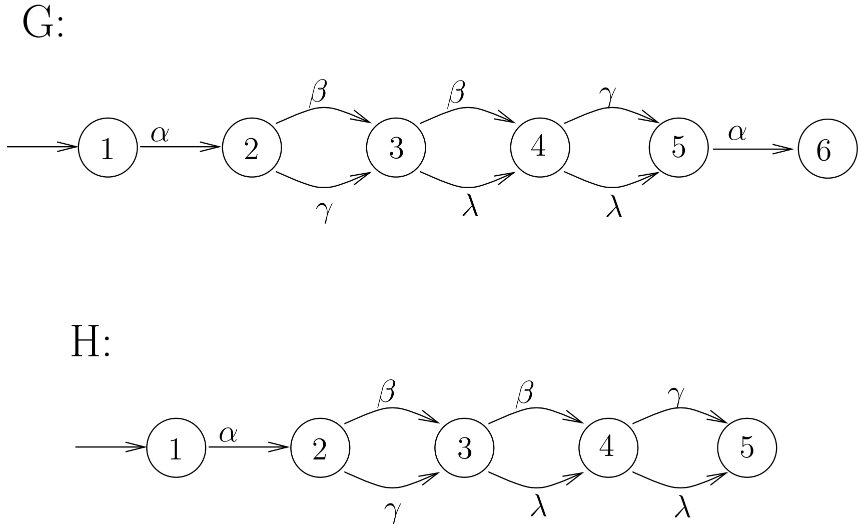

Figure 2: The system G and specification H of Example 1.

Now suppose that the cost of using the sensors is non-uniform, such that

$$cost(\alpha) = 7, cost(\beta) = 4, cost(\gamma) = 5, cost(\lambda) = 2.$$

With these sensor costs, the minimal cost sensor selection is $\{\beta,\lambda\}$.

There is a special case of Problem 1 where the sensors have uniform cost and the cost minimization problem becomes a cardinality minimization problem.

Problem 2 Minimal Cardinality Sensor Selection: *Given G, H and* $\Sigma_c \subseteq \Sigma$*, find a sufficient sensor selection* Σ_o^{min} *such that for any other sufficient sensor selection* Σ_o, $|\Sigma_o^{min}| \leq |\Sigma_o|$.

A simple example of Problem 2 is now shown.

Example 2 *Consider the system and specification seen in Figure 2. Suppose that $\Sigma_c = \{\alpha\}$. There are several sufficient sensor selections for this specification with respect to the given system: $\{\alpha\}, \{\beta, \gamma\}, \{\gamma, \lambda\}, \{\beta, \lambda\}$. However, $\{\alpha\}$ is the minimal cardinality sufficient sensor selection.*

Because of the NP-completeness of Problem 2, the minimal cardinality sensor selection cannot always be found in a computationally efficient manner [21]. This result therefore shows that Problem 1 is similarly computationally difficult.

However, despite the computational difficulties of the sensor selection problems, a sufficient sensor selection Σ_o may still need to be found reasonably efficiently such that the cost of this sensor selection ($cost(\Sigma_o)$) is as close to the minimal cost sensor selection ($cost(\Sigma_o^{min})$) as possible. Fortunately, as mentioned above, some NP-complete minimization problems have fairly accurate polynomial time approximation algorithms [2, 20]. This means sufficient and approximate solutions can be found for many computationally difficult problems in a reasonable amount of time. However, to the best of the authors' knowledge, there has been no investigation into the approximation difficulty of sensor selection problems.

4 The Graph Cutting Problem

It was mentioned above that the observability of languages marked by finite state automata can be tested using a nondeterministic automata construction introduced in [19]. This construction can be used to convert sensor selection problems into a special type of graph cutting problem called an "edge colored directed graph *st*-cut problem".

For the edge colored directed graph *st*-cut problem, assume an edge colored directed graph $D = (V, A, C)$ is given where V is a set of vertices, $A \subseteq V \times V$ are directed edges and $C = \{c_1, \ldots, c_p\}$ is a set of colors. Each edge is assigned a color in C. The directed graph in Figure 3 is an example of an edge colored directed graph where the edges are assigned colors $\{\alpha, \beta, \gamma, \lambda\}$.

Let A_i be the set of edges having color c_i. Given $I \subseteq C$, let $A_I = \cup_{c_i \in I} A_i$. For two nodes $s, t \in V$ such that there is a path of directed edges from s to t, I is a colored *st*-cut if $(V, (A \setminus A_I), C)$ has no path from s to t. As seen in Figure 4, $I = \{\beta, \gamma\}$ is a colored *st*-cut for the graph in Figure 3.

It is also possible that the various edge colors might have a non-uniform cost function $cost : C \to \Re^+ \cup \{0\}$ associated with using that color in a

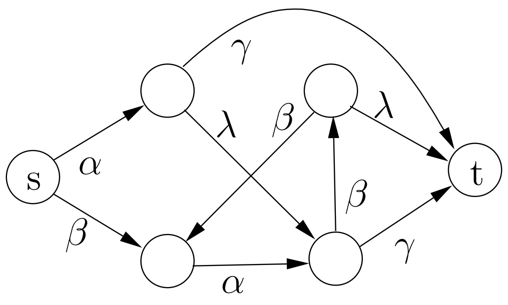

Figure 3: An example of an edge colored directed graph.

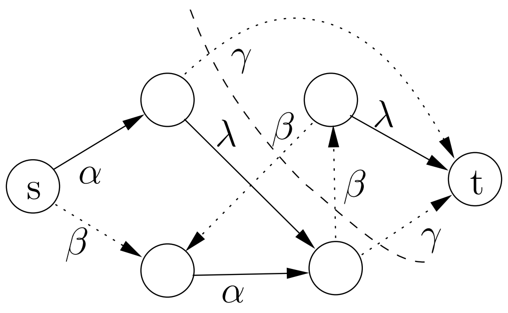

Figure 4: The colored cut $I = \{\beta, \gamma\}$ for the graph in Figure 3.

colored cut. The cost function can be extended in the usual manner to be defined over sets of colors. That is, for $C_1 \subseteq C$, $cost(C_1) = \sum_{c \in C_1} cost(c)$. This cost function induces the minimal cost colored cut problem seen below.

Problem 3 Minimal Cost Colored Cut: *For an edge colored directed graph $D = (V, A, C)$, two vertices $s, t \in V$, and a cost function $cost : C \rightarrow \Re^+ \cup \{0\}$, find a colored st-cut $I^{min} \subseteq C$ such that for any other colored st-cut $I \subseteq C$, $cost(I^{min}) \leq cost(I)$.*

Similar to the Minimal Cost Sensor Selection Problem, the Minimal Cost Colored Cut problem has a special case where $cost(\cdot)$ is a constant function.

Problem 4 Minimal Cardinality Colored Cut: *For an edge colored directed graph $D = (V, A, C)$, two vertices $s, t \in V$, find a colored st-cut $I^{min} \subseteq C$ such that for any other colored st-cut $I \subseteq C$, $|I^{min}| \leq |I|$.*

In the following subsections it is shown how to convert colored cut problems into sensor selection problems and vice-versa.

4.1 Converting Colored Cut Problems into Sensor Selection Problems

It is now shown how to convert an instance of a minimal cost colored cut problem into an instance of a minimal cost sensor selection problem. Suppose an edge colored directed graph $D = (V, A, C)$, two vertices $s, t \in V$ and a cost function $cost : C \rightarrow \Re^+ \cup \{0\}$ are given. A system G, a specification H and a controllable event set Σ_c are now constructed from D. For the colors $C = \{c_1, \dots, c_p\}$, let the event set Σ include a corresponding set of events $\{\sigma_1, \dots, \sigma_p\}$ such that every color $c_i \in C$ is paired with event σ_i. Let γ be another event and define $\Sigma = \{\sigma_1, \dots, \sigma_p, \gamma\}$. Also define $X^G = V \cup \{s', s'', t'\}$ where s', s'', t' are states not in V. Let $x_0^G = s$. To define the state transition function, let v_1, v_2 be any vertices except s. If $(v_1, v_2) \in A_i$, then $v_1 \overset{\sigma_i}{\mapsto}_G v_2$. If $(s, v_2) \in A_i$, then $s \overset{\sigma_i}{\mapsto}_G v_2$ and $s'' \overset{\sigma_i}{\mapsto}_G v_2$. If $(v_1, s) \in A_i$, then $v_1 \overset{\sigma_i}{\mapsto}_G s''$. For simplicity it is assumed that $(s, s) \notin A$. Also, transitions are added such that $s \overset{\gamma}{\mapsto}_G s'$ and $t \overset{\gamma}{\mapsto}_G t'$. Let H be a copy of G except that $\delta^H(t, \gamma)$ is undefined. Let $\Sigma_c = \{\gamma\}$. For the definition of the cost function assign

$$\forall i \in \{1, \dots, p\} cost(\sigma_i) = cost(c_i).$$

An example of such a system construction for converting a directed graph D to a system and specification G and H is given in Figure 5.

The following theorem demonstrates the correctness of the construction.

Theorem 2 *Consider a edge colored directed graph $D = (V, A, C)$ with a system G and a specification H constructed from it as outlined above. A set of colors $I = \{c_a, \dots, c_z\}$ is a colored st-cut for D if and only if selecting the observable events $\Sigma_o = \{\sigma_a, \dots, \sigma_z\}$ corresponding to I makes the system G observable with respect to H.*

Proof: This proof is demonstrated in two parts. First, suppose that $I = \{c_a, \dots, c_z\}$ is a colored *st*-cut for D. Then, for any path from s to t in D, there is an edge colored by one of the colors in I. Therefore, due to the construction of G and H, for any path from s to t in G, there is a state transition labeled by an event in $\{\sigma_a, \dots, \sigma_z\}$ and any path from s to t in G and H does not return to s after initialization. According to H, the event γ should be enabled at initialization and should be disabled at state t. Due

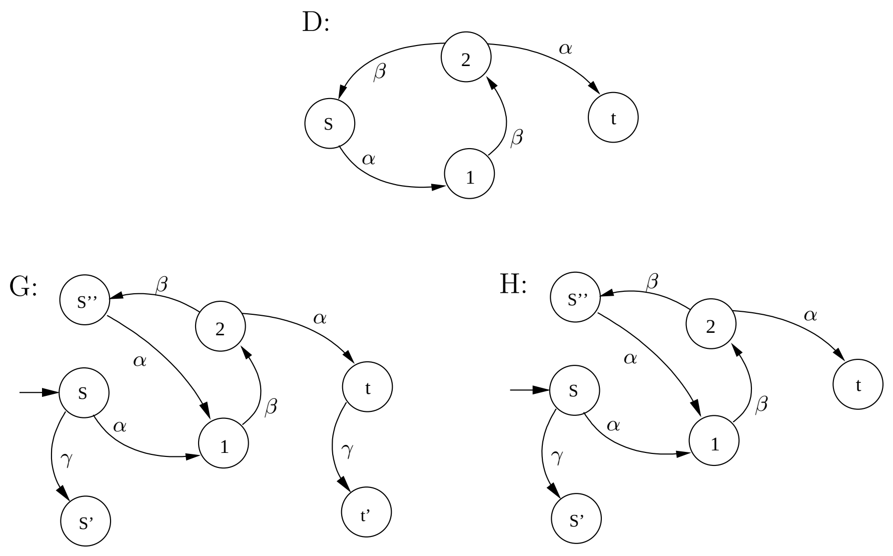

Figure 5: A directed graph D and the systems G and H constructed from it.

to the fact that for every path from s to t in G there is a state transition labeled by an event in $\{\sigma_a, \dots, \sigma_z\}$ and any path from s to t in G and H does not return to s after initialization. Therefore, if events in $\{\sigma_a, \dots, \sigma_z\}$ are observable, then there is no control conflict when γ needs to be disabled. Hence, selecting the observable events $\Sigma_o = \{\sigma_a, \dots, \sigma_z\}$ corresponding to I makes the system G observable with respect to H.

Now suppose that $I = \{c_a, \dots, c_z\}$ is not a colored st-cut for D. Then, there is a path from s to t in D with no edge colored with a color in I. Therefore, due to the construction of G and H, there is a path from s to t in G with no state transitions labeled by events in $\{\sigma_a, \dots, \sigma_z\}$. Hence, if only the events in $\{\sigma_a, \dots, \sigma_z\}$ are observable, then there is a path from s to t in G such that when γ should be disabled at t, a controller would not know for sure that γ should be disabled. Consequently, selecting the observable events $\Sigma_o = \{\sigma_a, \dots, \sigma_z\}$ corresponding to I does not make the system G observable with respect to H. ■

$\Sigma_o = \{\sigma_a, \dots, \sigma_z\}$ corresponding

4.2 Converting Sensor Selection Problems into Colored Cut Problems

The construction is now shown to convert an instance of Problem 1 into an instance of Problem 3. Suppose $H = (X^H, x_0^H, \Sigma, \delta^H)$, $G = (X^G, x_0^G, \Sigma, \delta^G)$, Σ_o and Σ_c are given and it is desired to test if $\mathcal{L}(H)$ is observable with respect to $\mathcal{L}(G)$, Σ_o and Σ_c. This is done by constructing an automaton $\mathcal{M}_{\Sigma_o} = (X^{\mathcal{M}_{\Sigma_o}}, x_0^{\mathcal{M}_{\Sigma_o}}, \Sigma^{\mathcal{M}_{\Sigma_o}}, \delta^{\mathcal{M}_{\Sigma_o}})$ that is a modification of $\mathcal{M}$ automaton method for testing observability and co-observability in [18, 19]. The $\mathcal{M}_{\Sigma_o}$ automaton is effectively a nondeterministic simulation of estimates an observer may make of unobservable system behavior with respect to a specification based on imperfect predictions of occurrences of unobservable events $(\Sigma \setminus \Sigma_o)$ in the system.

Let Σ' be a copy of the event set Σ where for every event $\sigma \in \Sigma$, there is a corresponding event $\sigma' \in \Sigma'$. The following are then defined:

$$\begin{array}{lcl} X^{\mathcal{M}_{\Sigma_o}} & := & X^H \times X^H \times X^G \cup \{d\}, \\ x_0^{\mathcal{M}_{\Sigma_o}} & := & (x_0^H, x_0^H, x_0^G), \\ \Sigma^{\mathcal{M}_{\Sigma_o}} & := & \Sigma \cup \Sigma'. \end{array}$$

Suppose a string of events s has been simulated to occur in the system G by $\mathcal{M}_{\Sigma_o}$ and the simulation is at state $(x_1, x_2, x_3) \in X^{\mathcal{M}_{\Sigma_o}}$. State x_3 represents the true state of the system G and x_2 represents the corresponding state of the specification H after s has occurred. States x_2 and x_3 always update simultaneously. However, as was stated above, the observer attempts to predict the occurrence of system events and the state x_1 represents a possible observer estimate of the specification state based on imperfect predictions of the simulated system behavior due to the observation of $P(s)$.

At state (x_1, x_2, x_3) of the simulation, if an event σ is correctly predicted by the observer in the simulation, there is a transition from (x_1, x_2, x_3) labeled by σ where all of the component states of (x_1, x_2, x_3) update on the occurrence of σ according to the transition rules of H, H and G respectively. A correct prediction may occur for either observable or unobservable events.

However, if an event σ occurs in the system that is not predicted correctly by the observer in the simulation, there is a transition from (x_1, x_2, x_3) labeled by σ' where the x_2, x_3 component states of (x_1, x_2, x_3) update on the occurrence of σ according to the transition rules of H and G respectively. Similarly, if an event σ does not occur in the system but is incorrectly predicted to have occurred by the observer in the simulation, then there is a transition from (x_1, x_2, x_3) labeled by σ' where the x_1 component state of (x_1, x_2, x_3) updates on the occurrence of σ according to the transition rules of H. Therefore, because unobservable event occurrences can not be guaranteed to be perfectly predicted, the $\mathcal{M}_{\Sigma_o}$ simulation is nondeterministic.

If the $\mathcal{M}_{\Sigma_o}$ simulation ever reaches a composed state where the observer believes the occurrence of a controllable event is allowed by the specification due to the properties of state x_1, but in reality it is not due to specification state x_2, yet still possible due to system state x_3, then there is a control conflict as the observer would believe a controllable event is illegal when in reality it is not. This possibility is captured by the $(*)$ condition such that if the simulation could reach a state (x_1, x_2, x_3) where $(*)$ holds, then illegal controllable behavior could occur in the system without an observer being able to resolve the control conflict.

$$\left.\begin{array}{ll} \delta^H(x_1,\sigma) & \text{is defined if } \sigma \in \Sigma_c \\ \delta^H(x_2,\sigma) & \text{is not defined} \\ \delta^G(x_3,\sigma) & \text{is defined} \end{array}\right\} \qquad (*)$$

The nondeterministic transition relation $\delta^{\mathcal{M}_{\Sigma_o}}$ is now more formally defined as follows.
For $\sigma' \in \Sigma'$ such that for the corresponding $\sigma \in \Sigma$, $\sigma \notin \Sigma_o$:

$$\delta^{\mathcal{M}_{\Sigma_o}}((x_1,x_2,x_3),\sigma') = \left\{ \begin{array}{ll} (\delta^H(x_1,\sigma), x_2, x_3) & \text{if } \delta^H(x_1,\sigma)! \\ (x_1, \delta^H(x_2,\sigma), \delta^G(x_3,\sigma)) & \text{if } (\delta^H(x_2,\sigma)! \wedge \delta^G(x_3,\sigma)!) \end{array} \right\}.$$

For $\sigma \in \Sigma$,

$$\delta^{\mathcal{M}_{\Sigma_o}}((x_1,x_2,x_3),\sigma) = \left\{ \begin{array}{ll} (\delta^H(x_1,\sigma), \delta^H(x_2,\sigma), \delta^G(x_3,\sigma)) & \text{if } \left(\begin{array}{c} \delta^H(x_1,\sigma)! \wedge \\ \delta^H(x_2,\sigma)! \wedge \\ \delta^G(x_3,\sigma)! \end{array} \right) \\ d & \text{if } (*) \end{array} \right\}.$$

No other transitions are defined in $\mathcal{M}_{\Sigma_o}$. The notation is used such that $\delta^H(x,\sigma)!$ is true if $\delta^H(x,\sigma)$ is defined and false otherwise. A similar definition holds for $\delta^G(x,\sigma)!$. Note that because $\mathcal{M}_{\Sigma_o}$ is nondeterministic, $\delta^{\mathcal{M}_{\Sigma_o}}(\cdot,\cdot)$ possibly returns a set of states.

The $\mathcal{M}_{\Sigma_o}$ automaton here is modified from the original in [19] in that Σ' transitions replace some Σ transitions. The $\sigma' \in \Sigma'$ transitions correspond to transitions that would not exist if $\sigma \in \Sigma \setminus \Sigma_o$ were made observable. The $\mathcal{M}_{\Sigma_o}$ automaton construction prompts the following proposition which follows from the results in [19].

Proposition 1 *[19] The state d is not reachable in $\mathcal{M}_{\Sigma_o}$ if and only if $\mathcal{L}(H)$ is observable with respect to $\mathcal{L}(G)$, Σ_o and Σ_c.*

An example is now given of an $\mathcal{M}_{\Sigma_o}$ automaton construction.

Example 3 *Recall the system and specification shown in Example 1. The $\mathcal{M}_{\emptyset}$ automaton constructed for this system and specification with $\Sigma_c = \{\alpha\}$ can be seen in Figure 6.*

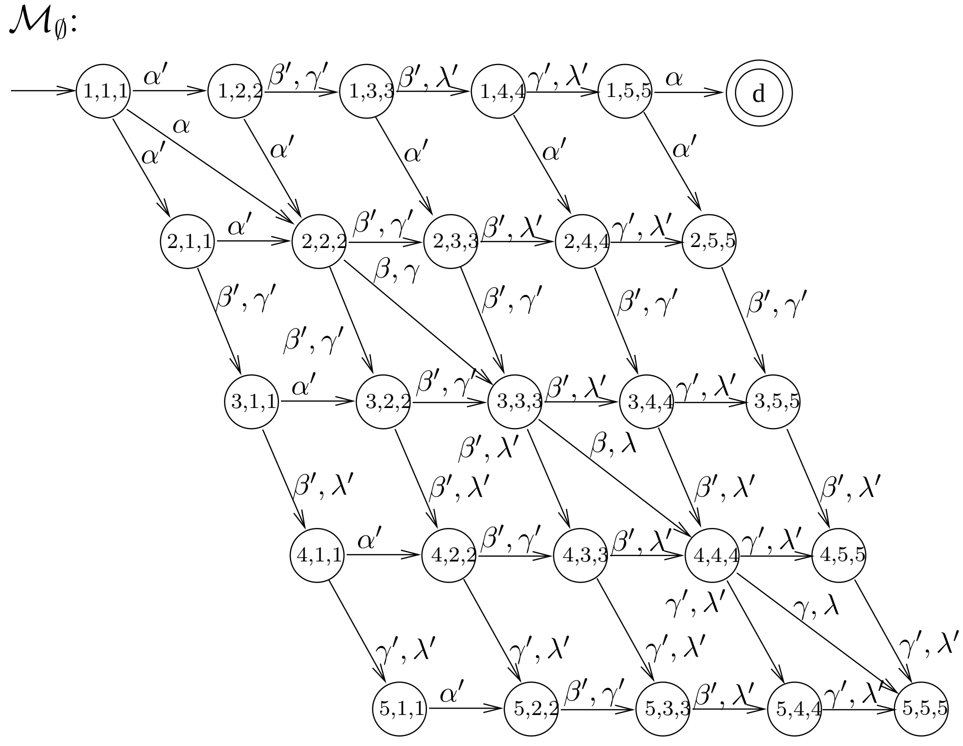

Figure 6: The $\mathcal{M}_{\emptyset}$ machine constructed from G and H of Example 1.

The behavior of $\mathcal{M}_{\Sigma_o}$ nondeterministically simulates an observer's estimate of a system's behavior with respect to a specification. As was mentioned above, a Σ transition occurs in $\mathcal{M}_{\Sigma_o}$ if an event occurrence in G is correctly predicted by the observer, and a Σ' transition occurs if the prediction is not correct. Therefore, if an event is observed, it is predicted correctly and σ' transitions in $\mathcal{M}_{\Sigma_o}$ would be removed if σ were made observable. This implies that $\mathcal{M}_{\Sigma_o \cup \{\sigma\}}$ can be constructed from $\mathcal{M}_{\Sigma_o}$ by cutting all σ' transitions, and conversely, cutting all occurrences of σ' transitions in $\mathcal{M}_{\Sigma_o}$ corresponds to adding σ to Σ_o. This realization prompts the following lemma.

Lemma 1 *The automaton $\mathcal{M}_{\Sigma_o}$ can be constructed from $\mathcal{M}_\emptyset$ by cutting Σ'_o labeled transitions in $\mathcal{M}_\emptyset$.*

Proof: This lemma is shown by a proof by induction on the cardinality of Σ_o.

Base: Suppose $\Sigma_o = \emptyset$. This case is trivial as $\mathcal{M}_{\Sigma_o} = \mathcal{M}_\emptyset$.

Induction hypothesis: For $|\Sigma_o| = n$, the $\mathcal{M}_{\Sigma_o}$ automaton can be constructed from $\mathcal{M}_\emptyset$ by cutting Σ'_o labeled transitions in $\mathcal{M}_\emptyset$.

Induction step: Let $|\Sigma_o| = n$. From the induction hypothesis it is known that the $\mathcal{M}_{\Sigma_o}$ automaton can be constructed from $\mathcal{M}_\emptyset$ by cutting Σ'_o labeled transitions in $\mathcal{M}_\emptyset$.

Let σ be some event in $\Sigma \setminus \Sigma_o$. From the construction of $\mathcal{M}_{\Sigma_o}$ and $\mathcal{M}_{(\Sigma_o \cup \{\sigma\})}$, the only difference in the transition structure of these two automata is that transitions labeled by σ' are absent in $\mathcal{M}_{(\Sigma_o \cup \{\sigma\})}$. Therefore the $\mathcal{M}_{(\Sigma_o \cup \{\sigma\})}$ automaton can be constructed from $\mathcal{M}_{\Sigma_o}$ by cutting all σ' labeled transitions in $\mathcal{M}_{\Sigma_o}$. Hence, the $\mathcal{M}_{(\Sigma_o \cup \{\sigma\})}$ automaton can be constructed from $\mathcal{M}_\emptyset$ by cutting $\Sigma'_o \cup \{\sigma'\}$ labeled transitions in $\mathcal{M}_\emptyset$. ■

Lemma 1 shows that the sensor selection problem is really a type of colored cut problem. Suppose the automaton $\mathcal{M}_\emptyset$ is considered to be an edge colored directed graph as introduced above such that the transition labels are defined to be edge colors and the cost of cutting all edges associated with a color is defined to be the cost of making the corresponding event observable. A colored $x_0^{\mathcal{M}_\emptyset}d$-cut for $\mathcal{M}_\emptyset$ where only Σ' transitions are cut corresponds to a sufficient sensor selection for observability to hold. This prompts the following theorem.

Theorem 3 *$\mathcal{L}(H)$ is observable with respect to $\mathcal{L}(G)$, Σ_o and Σ_c if and only if $\Sigma'_o \subseteq \Sigma'$ is a colored $x_0^{\mathcal{M}_\emptyset}d$-cut for $\mathcal{M}_\emptyset$.*

Proof: This proof is demonstrated in two parts. Suppose that $\mathcal{L}(H)$ is observable with respect to $\mathcal{L}(G)$, Σ_o and Σ_c. Therefore d is not reachable in $\mathcal{M}_{\Sigma_o}$. From Lemma 1 the automaton $\mathcal{M}_{\Sigma_o}$ can be constructed from $\mathcal{M}_\emptyset$ by cutting Σ'_o labeled transitions in $\mathcal{M}_\emptyset$. Hence, Σ'_o is a colored $x_0^{\mathcal{M}_\emptyset}d$-cut in $\mathcal{M}_\emptyset$.

Now suppose that $\mathcal{L}(H)$ is not observable with respect to $\mathcal{L}(G)$, Σ_o and Σ_c. Therefore d is reachable in $\mathcal{M}_{\Sigma_o}$. From Lemma 1 the automaton $\mathcal{M}_{\Sigma_o}$

can be constructed from $\mathcal{M}_\emptyset$ by cutting Σ_o' labeled transitions in $\mathcal{M}_\emptyset$. Hence, Σ_o' is a not a colored $x_0^{\mathcal{M}_\emptyset}d$-cut in $\mathcal{M}_\emptyset$. ■

The $\mathcal{M}_\emptyset$ cut problem is not in the same form as in Problem 3 or Problem 4 as Σ labeled transitions can never be cut in the $\mathcal{M}_\emptyset$ automaton of Theorem 3 by making events observable. To counter this difference the $\tilde{\mathcal{M}}_{\Sigma_o}$ construction below is used which is a copy of $\mathcal{M}_{\Sigma_o}$ except that some states are combined to hide Σ transitions in $\mathcal{M}_\emptyset$.

To start, construct $\mathcal{M}_{\Sigma_o}$ from H, G, Σ_c and Σ_o. Define:

$$X_x^{\mathcal{M}_{\Sigma_o}} = \left\{ y^{\mathcal{M}_{\Sigma_o}} | \exists t \in \Sigma^* \text{ such that } x^{\mathcal{M}_{\Sigma_o}} \overset{t}{\mapsto}_{\mathcal{M}_{\Sigma_o}} y^{\mathcal{M}_{\Sigma_o}} \right\}.$$

Notice the x subscript on $X_x^{\mathcal{M}_{\Sigma_o}}$. The set $X_x^{\mathcal{M}_{\Sigma_o}}$ represents all states that could be reached from $x^{\mathcal{M}_{\Sigma_o}}$ in $\mathcal{M}_{\Sigma_o}$ if only Σ transitions were allowed. These are the same transitions in $\mathcal{M}_{\Sigma_o}$ that could not be cut by making more events observable. Due to this, the states in $X_x^{\mathcal{M}_{\Sigma_o}}$ would be reachable from $x^{\mathcal{M}_{\Sigma_o}}$ according to the transition rules of $\mathcal{M}_{\Sigma_o}$ no matter what events are made observable.

With this in mind, the following nondeterministic automaton $\tilde{\mathcal{M}}_{\Sigma_o}$ is constructed from $\mathcal{M}_{\Sigma_o}$ such that if there are two states $x^{\mathcal{M}_{\Sigma_o}}, y^{\mathcal{M}_{\Sigma_o}}$ and some string of transitions labeled by $s\sigma' \in \Sigma^*\Sigma'$ such that according to the transition rules of $\mathcal{M}_{\Sigma_o}$, $x^{\mathcal{M}_{\Sigma_o}} \overset{s\sigma'}{\mapsto}_{\mathcal{M}_{\Sigma_o}} y^{\mathcal{M}_{\Sigma_o}}$, then according to the transition rules of $\tilde{\mathcal{M}}_{\Sigma_o}$, $x^{\mathcal{M}_{\Sigma_o}} \overset{\sigma}{\mapsto}_{\tilde{\mathcal{M}}_{\Sigma_o}} y^{\mathcal{M}_{\Sigma_o}}$ where $x^{\mathcal{M}_{\Sigma_o}}$ and $x^{\mathcal{M}_{\Sigma_o}}$ are states in both $\mathcal{M}_{\Sigma_o}$ and $\tilde{\mathcal{M}}_{\Sigma_o}$, but with different outgoing state transitions in the two automata. This construction effectively condenses all $\mathcal{M}_{\Sigma_o}$ states reachable by Σ transitions abd replaces the remaining Σ' labels with the corresponding Σ labels. However, it is assumed that $d \notin X_{x_0}^{\mathcal{M}_{\Sigma_o}}$ because if $d \in X_{x_0}^{\mathcal{M}_{\Sigma_o}}$, then even if a controller could observe the occurrences of all events ($\Sigma_o = \Sigma$), the system could not be made observable in any case.

Let $\tilde{\mathcal{M}}_{\Sigma_o} = (X^{\tilde{\mathcal{M}}_{\Sigma_o}}, x_0^{\tilde{\mathcal{M}}_{\Sigma_o}}, \Sigma^{\tilde{\mathcal{M}}_{\Sigma_o}}, \delta^{\tilde{\mathcal{M}}_{\Sigma_o}})$ where

$$\begin{aligned} X^{\tilde{\mathcal{M}}_{\Sigma_o}} &:= X^H \times X^H \times X^G \cup \{d\}, \\ x_0^{\tilde{\mathcal{M}}_{\Sigma_o}} &:= (x_0^H, x_0^H, x_0^G), \\ \Sigma^{\tilde{\mathcal{M}}_{\Sigma_o}} &:= \Sigma. \end{aligned}$$

The transition relation $\delta^{\tilde{\mathcal{M}}_{\Sigma_o}}$ is defined as follows.

Suppose there exists three states $x^{\mathcal{M}_{\Sigma_o}}, y^{\mathcal{M}_{\Sigma_o}}, z^{\mathcal{M}_{\Sigma_o}} \in X^{\mathcal{M}_{\Sigma_o}}$ and an event $\sigma \in \Sigma$ such that $z^{\mathcal{M}_{\Sigma_o}} \in X_x^{\mathcal{M}_{\Sigma_o}}$ and $z^{\mathcal{M}_{\Sigma_o}} \overset{\sigma'}{\mapsto}_{\mathcal{M}_{\Sigma_o}} y^{\mathcal{M}_{\Sigma_o}}$. Then,

$$\delta^{\tilde{\mathcal{M}}_{\Sigma_o}}(x^{\mathcal{M}_{\Sigma_o}}, \sigma) = \left\{ \begin{array}{ll} y^{\mathcal{M}_{\Sigma_o}} & \text{if } d \notin X_y^{\mathcal{M}_{\Sigma_o}} \\ d & \text{if } d \in X_y^{\mathcal{M}_{\Sigma_o}} \end{array} \right\}.$$

An example is now given of an $\tilde{\mathcal{M}}_{\Sigma_o}$ automaton construction.

Example 4 *Recall the system and specification shown in Example 1 and the resulting $\mathcal{M}_{\Sigma_o}$ automaton seen in Figure 6. The corresponding $\tilde{\mathcal{M}}_\emptyset$ automaton constructed for this system and specification with $\Sigma_c = \{\alpha\}$ can be seen in Figure 7.*

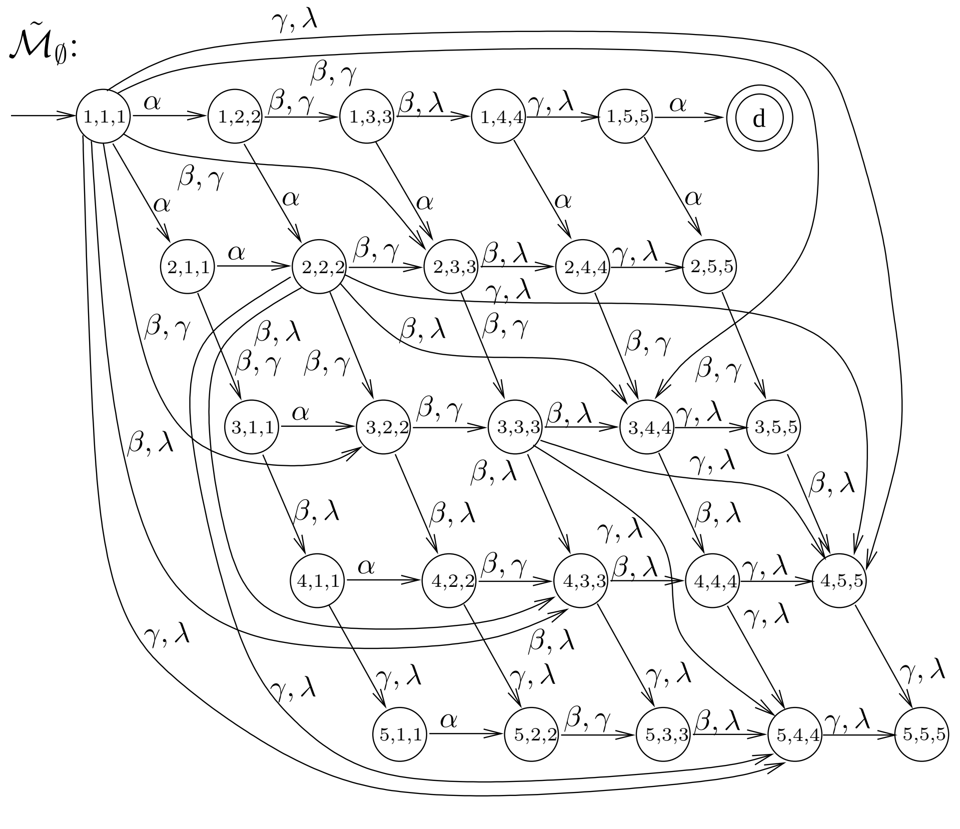

Figure 7: The $\tilde{\mathcal{M}}_\emptyset$ machine constructed from G and H of Example 1.

The $\tilde{\mathcal{M}}_{\Sigma_o}$ automaton is really a colored directed graph where states are vertices, transitions are directed edges and the transition labels are the colors. This prompts one of the main results of this chapter.

Theorem 4 *Given an $\tilde{\mathcal{M}}_\emptyset$ automaton constructed from H, G, Σ_c and $\emptyset$ as the set of observable events, $\mathcal{L}(H)$ is observable with respect to $\mathcal{L}(G)$, Σ_o and Σ_c if and only if Σ_o is a colored $x_0^{\tilde{\mathcal{M}}_\emptyset}d$-cut in the colored directed graph $\tilde{\mathcal{M}}_\emptyset$.*

Proof: It has already been shown that $\mathcal{L}(H)$ is observable with respect to $\mathcal{L}(G)$, Σ_o and Σ_c if and only if Σ'_o is a colored $x_0^{\mathcal{M}_\emptyset}d$-cut in the colored directed graph $\mathcal{M}_\emptyset$. Therefore it is sufficient to show that Σ_o is a colored $x_0^{\tilde{\mathcal{M}}_\emptyset}d$-cut in the colored directed graph $\tilde{\mathcal{M}}_\emptyset$ if and only if Σ'_o is a colored $x_0^{\mathcal{M}_\emptyset}d$-cut in the colored directed graph $\mathcal{M}_\emptyset$.

Define a natural projection operation $P' : \Sigma \cup \Sigma' \to \Sigma'$. Also define the translation operator $\tilde{\Psi} : \Sigma' \to \Sigma$ such that $\tilde{\Psi}(\sigma') = \sigma$. Both of these functions are extended in the usual manner to be defined over strings. Also define the function $\tilde{P} : \Sigma \cup \Sigma' \to \Sigma$ that is the composition of $P'(\cdot)$ and $\tilde{\Psi}(\cdot)$, i.e., $\tilde{P}(\sigma) = \tilde{\Psi}(P'(\sigma))$. These functions also have inverse operations defined in the usual manner.

First, suppose that Σ'_o is not a colored $x_0^{\mathcal{M}_\emptyset}d$-cut in the colored directed graph $\mathcal{M}_\emptyset$. Then there exists a string of transitions labeled by $s \in (\Sigma \cup \Sigma')^*$ such that $x_0^{\mathcal{M}_\emptyset} \overset{s}{\mapsto}_{\mathcal{M}_\emptyset} d$. Due to the construction of $\tilde{\mathcal{M}}_\emptyset$, $x_0^{\tilde{\mathcal{M}}_\emptyset} \overset{\tilde{P}(s)}{\mapsto}_{\tilde{\mathcal{M}}_\emptyset} d$.

Now suppose that Σ'_o is not a colored $x_0^{\tilde{\mathcal{M}}_\emptyset}d$-cut in the colored directed graph $\tilde{\mathcal{M}}_\emptyset$. Then there exists a string of transitions labeled by $s \in \Sigma^*$ such that $x_0^{\tilde{\mathcal{M}}_\emptyset} \overset{s}{\mapsto}_{\tilde{\mathcal{M}}_\emptyset} d$. Due to the construction of $\tilde{\mathcal{M}}_\emptyset$, there exists some string $t \in \tilde{P}^{-1}(s)$ such that $x_0^{\mathcal{M}_\emptyset} \overset{t}{\mapsto}_{\mathcal{M}_\emptyset} d$. ■

By the definition of NP-completeness there exists polynomial-time many-one reductions between the graph cutting and sensor selection problems. However, one of the main reasons the reductions between these problems are demonstrated is to show that approximation algorithms developed for one problem can be easily and intuitively used to develop approximation algorithms for the other.

Due the construction of the $\mathcal{M}_{\Sigma_o}$ automaton, it should be apparent that $|X^{\mathcal{M}_{\Sigma_o}}| \leq |X^G| * |X^H|^2 + 1$. Furthermore, at each state $x^{\mathcal{M}}_{\Sigma_o} \in X^{\mathcal{M}}_{\Sigma_o}$, the number of outgoing state transitions is at most three times the maximum number of outgoing state transitions in any state of G or H. If E^G is the set of state transitions in G and E^H is the number of state transitions in H, let $e = \max\{|E^G|, |E^H|\}$. Therefore $\mathcal{M}_{\Sigma_o}$ can be constructed in time and space in $O(e * |X^G| * |X^H|^2)$ using standard breadth-first digraph construction algorithms. Therefore, because reachability can be tested in polynomial time, the observability of $\mathcal{L}(H)$ with respect to $\mathcal{L}(G)$, Σ_o and Σ_c can be tested in polynomial time [19].

5 Inapproximability Results

To the knowledge of the authors, the egde colored directed graph cutting problem has not been explored in the standard literature (from graph theory or computer science). Unfortunately, although many other types of graph cutting problems are computationally simple, it is shown here that solutions to Problem 4 are most likely difficult to approximate. Because of the above results, solutions to the sensor selection problem are similarly difficult to approximate.

Consider a bipartite graph $K = (V_1, V_2, E)$ where the sets V_1 and V_2 are partitioned into a disjoint union of q sets, $V_1 = \cup_{i=1}^{q} Y_i$ and $V_2 = \cup_{j=1}^{q} Z_i$. The sets $\{Y_1, Z_1, \ldots , Y_q, Z_q\}$ all have size N. E is the set of edges in K. The bipartite graph and the partitions of V_1 and V_2 induce a super-graph $\mathcal{H}$ where the vertices of $\mathcal{H}$ are the sets $\{Y_1, Z_1, \ldots , Y_q, Z_q\}$. Y_i and Z_j are connected by a (super)edge in $\mathcal{H}$ if and only if there exists $(a, b) \in Y_i \times Z_j$ that are adjacent in K. For the purpose of this paper, it is convenient to assume that $\mathcal{H}$ is d-regular. Namely, every Y_i, Z_j has exactly d neighbors in the super-graph.

Given $X \subseteq V_1 \cup V_2$, it is said that the super-edge (Y_i, Z_j) in $\mathcal{H}$ is *covered* by X if there exists two nodes $a \in X \cap Y_i$ and $b \in X \cap Z_j$ such that $(a, b) \in E$. The MIN-REP problem can now be formally introduced.

Problem 5 MIN-REP*: Given a bipartite graph K as introduced above, find a subset X of minimum size covering all super-edges of $\mathcal{H}$.*

The subset $X \cap Y_i$ is referred to as the *representatives* of Y_i in X (similarly for Z_j). The set X is said to be a set of *unique representatives* if $|X \cap Y_i| = 1$ and $|X \cap Z_j| = 1$ for all i, j.

In addition the following result from [9] is needed that shows the approximation difficulty of Problem 5. Consider the following theorem based on the well-known satisfiability problem (SAT) from computer science.

Theorem 5 *[14]. Let T be an instance of SAT. For any $0 < \epsilon < 1$, there exists a (quasi-polynomial) reduction of T to K, an instance of MIN-REP with n vertices so that if T is satisfiable, then there exists a set of unique representatives which cover all super-edges. If T is not satisfiable, then the size of any MIN-REP solution has at least $2q2^{(\log n)^{1-\epsilon}}$ vertices.*

In the reduction in Theorem 5, if T is not satisfiable, then the average number of representatives needed per super-node is $\Omega(2^{(\log n)^{1-\epsilon}})$. Also, n is quasi-polynomial in the size of the SAT formula. This is used in [9] to show the approximation difficulty of the MIN-REP problem.

Theorem 6 *[9]. The MIN-REP problem admits no $2^{(\log n)^{1-\epsilon}}$ approximation for any $\epsilon > 0$ unless $NP \subseteq DTIME(n^{polylog\ n})$.*

A conversion of an instance of the MIN-REP problem, $K = (V_1, V_2, E)$, to an instance of the minimal cardinality colored cut problem, $D = (V, A, C)$, can now be shown that establishes a similar connection between SAT and the colored cut problem and ultimately the sensor selection problem. For simplicity, it is assumed that D may be a multi-graph, but by using a standard trick of subdividing edges, this assumption can be removed. The directed graph D is constructed in three parts.

Let $n = |V_1| + |V_2|$ be the number of vertices in the MIN-REP instance. First construct the vertices V of D. Add two nodes s and t to V. The rest of the vertices are associated with super-edges. Go over all super-edges $\tilde{e} = (Y_i, Z_j)$. For this super-edge the following vertices are added. Let the edges between Y_i, Z_j be $\{e^1_{ij}, \dots, e^b_{ij}\}$ where $e^k_{ij} = (y^k, z^k)$, $y^k \in Y_i$ and $z^k \in Z_j$. Also add the vertices $x^k_{i,j}$, $1 \le k \le b+1$.

Now construct the edges A. For every i, j, add two parallel edges from x^1_{ij} to x^2_{ij}, two parallel edges from x^2_{ij} to x^3_{ij} and so on until x^{b+1}_{ij} is reached. This construction of edges from $x^1_{i,j}$ to $x^{b+1}_{i,j}$ is called the *chain* of super-edge e_{ij}. Observe that there is one chain associated with Z_j for every Y_i adjacent to it.

Also, add n^3 parallel edges $e^1_{i,j}, \dots, e^{n^3}_{i,j}$ from s into $x^1_{i,j}$ (the first vertex in the chain). This is done for all i, j. Similarly, for the end of the chain, add n^3 parallel edges $d^1_{i,j}, \dots, d^{n^3}_{i,j}$ from $x^{b+1}_{i,j}$ into t.

Finally the edges of D are colored. For each $y \in Y_i$, $z \in Z_j$ let there be colors c_y and c_z. For the paired edges $e^k_{ij} = (y^k, z^k)$ used to construct the chains, color one of the edges (x^k_{ij}, x^{k+1}_{ij}) in the chain c_{y^k} and the other c_{z^k}. The edges touching s and t are colored new distinct colors. An example of the construction can be seen in Figure 8 for the case of $q = 1$.

Now that the construction has been shown, it is discussed why this demonstrates that the minimal cardinality colored cut problem is difficult to approximate. Consider a super-edge $e_{ij} = (Y_i, Z_j)$ and its chain. Let I be any feasible colored st-cut of D.

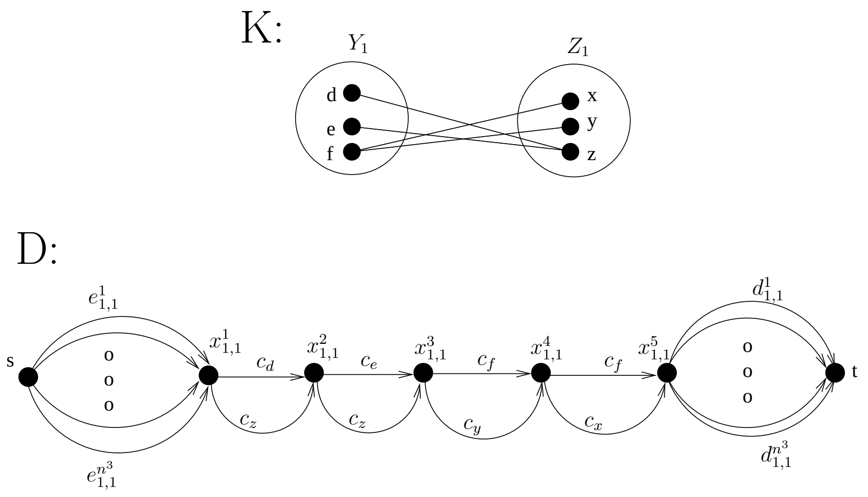

Figure 8: Conversion of a MIN-REP problem to a graph cutting problem when $q = 1$.

Lemma 2 *There exists at least one parallel pair of edges in the chain corresponding to e_{ij} so that I contains both colors corresponding to this parallel pair of edges.*

Proof: Suppose that for each pair of parallel edges, at most one color is in I. This implies that even with I colored edges, there is a path from all $x^k_{i,j}$, $1 \le k \le b+1$ vertices to $x^{b+1}_{i,j}$. Because s has n^3 parallel edges to $x^1_{i,j}$, all having different colors, it is too costly to disconnect $x^1_{i,j}$ from s. (Observe that the solution picking all the c_z, c_y colors for all y, z is feasible and has size n^2, hence an approximation ratio of better than n can not disconnect $x^1_{i,j}$ from s.) Similarly, it is too costly to disconnect t from $x^{b+1}_{i,j}$. Thus, if there is not a pair of parallel edges both cut in every chain, then I is not an st-cut because a directed path from s to t exists. ■

Let I be a colored st-cut for the minimal cardinality colored cut problem constructed above.

Corollary 1 *Let X be the vertices corresponding to the colored cut I. X is a feasible solution to the MIN-REP problem and $|X| = |I|$.*

Proof: Consider a super-edge (Y_i, Z_j). There is a parallel pair of edges such that its colors are in I by Claim 2. The color of each edge corresponds to some vertex $y \in Y_i$ and $z \in Z_j$ such that $(y, z) \in E$. Hence, I defines in a natural way a collection $X \subseteq V$ that is a solution for MIN-REP. Therefore, the X collection covers all the super-edges. Furthermore, the number of colors equals the number of vertices from X. ■

Lemma 3 *Given any MIN-REP solution X for K, a subset I of $|X|$ colors can be found that solves the minimal cardinality colored cut problem constructed from K.*

Proof: Given that $x \in X$, this defines a color c_x corresponding to x in the construction. Let I be the collection of all these colors. It is shown that I is a feasible solution for D.

Let $e_{ij} = (Y_i, Z_j)$. Consider the chain of e_{ij}. As e_{ij} is covered by X, there are $y \in X \cap Y_i$ and $z \in Z_j$ so that $(y, z) \in E$. By definition, there is a parallel pair of edges on this chain from x_{ij}^k corresponding to (y, z) and its colors are in I. The cutting of edges colored c_y and c_z cuts both of these parallel edges from D. It follows that there cannot be a path through the chain of e_{ij} to t. Since all super-edges are covered by X, it follows that s cannot reach t after the edges colored by I are removed. Furthermore, $|I| = |X|$. ■

The following is thus immediate.

Theorem 7 *Let T be an instance of SAT. For any $0 < \epsilon < 1$, there exists a (quasi-polynomial) reduction of T to D, an instance of colored cut with n vertices so that if T is satisfiable, then there exists a set of $2q$ colors forming a feasible solution to the minimal cardinality colored cut problem. If T is not satisfiable, then any colored cut solution has at least $2q2^{\log^{(1-\epsilon)} n}$ colors.*

Corollary 2 *The minimal cardinality colored cut problem admits no $2^{(\log n)^{1-\epsilon}}$-approximation, for any $\epsilon > 0$ unless $NP \subseteq DTIME(n^{polylog\ n})$.*

Corollary 3 *The minimal cardinality sensor selection problem admits no $2^{(\log n)^{1-\epsilon}}$-approximation for any $\epsilon > 0$ unless $NP \subseteq DTIME(n^{polylog\ n})$.*

It follows that if solutions to any of the colored cut or sensor selection problems discussed in this paper can be approximated with better than a $2^{(\log n)^{1-\epsilon}}$-approximation, then a method for solving NP-complete problems in quasipolynomial time has been found. This lower bound is generally considered to be a very poor lower bound in the computer science community. Indeed, as ϵ approaches 0, then $2^{(\log n)^{1-\epsilon}}$ approaches n. So far, for all problems admitting this bound, the best existing approximation known is n^ϵ for some constant $\epsilon > 0$. We show such an approximation for a special case.

6 Algorithm for a Special Case

Now that it has been shown that colored cuts and sensor selections are in a sense difficult to approximate, an approximation method for a special case with a reasonable approximation bound is shown. Suppose an instance of the minimal cardinality colored cut problem $D = (V, A, C)$ is given with vertices $s, t \in V$ such that there are no parallel edges in the same direction. K is a prespecified value which is optimized in the following proof.

Algorithm 1
Input: $D = (V, A, C), s, t \in V$ *and* K*, an integer.*
Output: A colored st-cut I *for* D*.*
Initialize: $I := \emptyset$*, a set of colors.*

· *As long as there is a st path of length at most* K*, add the colors associated with this path to* I *and remove all edges in* D *with colors in* I*.*
· *After all paths of length at most* K *have been removed, let* k *be the distance from* s *to* t*.*
· *Perform a depth-first-search to convert* D *to a layered directed graph* D' *of depth* k*.*
· *Let* V_i *denote the set of vertices at distance* i *from* s *in* D' *and let* A_i *denote the set of edges from* V_i *to* V_{i+1}*.*
· *Select the set of edges* A_j *that uses the least number of colors and add those colors to* I*.*

Return: I*.*

Theorem 8 *Algorithm 1 gives a* $2|V|^{2/3}$ *approximation for Problem 4 where* D *contains no parallel edges in the same direction.*

Proof: Let I^{min} represent the minimal colored cut. Suppose in the first phase of the algorithm, paths $\{P_1, \dots, P_l\}$ are removed. At most lK colors are chosen during this step because there are at most K colors per path. The colors used by the paths are disjoint, but the optimal solution needs to remove at least one color for each path. Hence, $l \leq |I^{min}|$. If $\{I_1, \dots, I_l\}$ are the colors removed by each path, then $\Sigma_{i=1}^{l} |I_i| \leq K|I^{min}|$.

Let I_{A_j} represent the colors cut in the last part of the algorithm. Also, let $|A_p| = \min_i |A_i|$. It is known that $\Sigma_{i=1}^{k} |V_i| < |V|$ where k is the distance from s to t, $k \geq K$. Also, $A_i \leq |V_i||V_{i+1}|$. Therefore, $|I_{A_j}| \leq |I_{A_p}| \leq |A_p| \leq \left(\frac{|V|}{K}\right)^2$.

The optimal solution has at least one color from this cut that is different from the colors that have been added to I. $|I| = \Sigma_{i=1}^{l}|I_i| + |I_{A_j}| \leq K|I^{min}| + \left(\frac{|V|}{K}\right)^2$. By choosing $K = |V|^{2/3}$ it is easy to show with some mathematical manipulation that $\frac{|I|}{|I^{min}|} \leq 2|V|^{2/3}$ and hence a $2|V|^{2/3}$ approximation is obtained. ■

When a sensor selection problem is converted to a colored cut problem using the conversion outlined above, methods developed to solved the colored cut can be used to solve the sensor selection problem. In Algorithm 1, the restriction that D contains no parallel edges in the same direction is generally fairly restrictive for solving many interesting sensor selection problems. However, this method is still interesting and relevant from a graph theoretic and computer science point of view. The authors hope that by presenting this algorithm further research might be spurred into approximation algorithms for other special cases of the colored cut and sensor selection problems. The above conversion of the sensor selection problem to the colored cut problem is helpful for analysis of the sensor selection problem as graph cutting problems are much more intuitive and this would helpfully aid further research on these problems.

7 Integer Programming

It is now shown that another approach to approximating the minimal cost sensor selection is to use integer programming based methods. The integer programming problem is a general optimization problem from the field of the combinatorial optimization that has been well explored in the literature [11]. This section discusses how to convert the minimal cost colored cut problem to an integer programming problem. Therefore, using the reduction methods discussed above to convert the sensor selection problem into a colored cut problem, integer programming methods can also be used for the sensor selection problem. First the integer programming problem is introduced.

Problem 6 The Integer Programming Problem: *Given a z element row vector $\vec{C}$, a $y \times z$ matrix $\vec{A}$ and a y element column vector $\vec{B}$, find a z element column vector $\vec{x} \in \{0,1\}^z$ that minimizes $\vec{C}\vec{x}$ subject to $\vec{A}\vec{x} \geq \vec{B}$.*

The integer programming problem is known to be NP-complete, but there is a vast literature on calculating approximate solutions to this problem as outlined in [11, 20]. Unfortunately, the integer programming problem is known to be NPO-complete [2] which means that it is in the most difficult class of NP-complete optimization problems. However, because of the problem conversion methods discussed in this section, already developed and mature methods for the well understood integer programming can be used to find solutions to the sensor selection problem.

7.1 Problem Conversion

It is now shown how to convert the minimal cost colored cut problem into an integer programming problem. Suppose an edge-colored directed graph $D = (V, A, C)$ is given with a cost function $cost : C \to \Re^+ \cup \{0\}$. Suppose $V = \{v_1, v_2, \ldots, v_{n_V}\}$, $A = \{a_1, a_2, \ldots, a_{n_A}\}$ and $C = \{c_1, c_2, \ldots, c_{n_C}\}$. Without loss of generality assume that for the graph cutting problem the task is to find the minimal cost colored $v_1 v_{n_V}$-cut $I \subseteq C$.

For the colors C, let there be a set of boolean variables $\{b_{c_1}, b_{c_2}, \ldots, b_{c_{n_C}}\}$ and for the set of vertices V, let there be another set of boolean variables $\{b_{v_1}, b_{v_2}, \ldots, b_{v_{n_V}}\}$. For a given cut I, values can be assigned to $\{b_{c_1}, b_{c_2}, \ldots, b_{c_{n_C}}\}$ such that $b_{c_i} = 1$ if and only if $c_i \in I$. Note that by definition, $\sum_{i=1}^{n_C} b_{c_i} cost(c_i)$ is the cost of the colored cut I. For a cut I, values are assigned to to $\{b_{v_1}, b_{v_2}, \ldots, b_{v_{n_V}}\}$ such that $b_{v_1} = 1$, and for all $i, j \in \{1, \ldots, n_V\}$ and $k \in \{1, \ldots, n_C\}$ such that if $(v_i, v_j) \in A_{c_k}$, then $(b_{v_i} = 1) \wedge (b_{c_k} = 0) \Rightarrow (b_{v_j} = 1)$. These constraints on the assignment of values to $\{b_{v_1}, b_{v_2}, \ldots, b_{v_{n_V}}\}$ can be thought of as a form of a reachability condition. That is, for a vertex v_i, if $b_{v_i} = 1$, then for all vertices v_j that are reachable from v_i along edges in $A \setminus A_I$, it must hold that $b_{v_j} = 1$. Therefore, because $b_{v_1} = 1$, these constraints imply that if v_j is reachable from v_1 along edges in $A \setminus A_I$, then $b_{v_j} = 1$.

Note that if it is necessary to assign $b_{v_{n_V}} = 1$ with the above constraints, then I is not a $v_1 v_{n_V}$-cut in D. However, if it is possible to assign $b_{v_{n_V}} = 0$ with the above constraints, then I is a $v_1 v_{n_V}$-cut in D. This is demonstrated in the following lemma. If the constraint is added that $b_{v_{n_V}} = 0$, then all vertices v_k which can reach v_{n_V} along edges in $A \setminus A_I$, it must hold that $b_{v_k} = 0$. Note that for any vertex v_j not reachable from v_1 with the colored cut I, or which cannot reach v_{n_V} with the colored cut I, the corresponding boolean variable b_{v_j} can be assigned arbitrarily.

Lemma 4 *Suppose an edge colored directed graph $D = (V, A, C)$ is given along with a set of colors $I \subseteq C$. The set I is a $v_1 v_{n_V}$-cut in D if and only if there exists boolean variables $\{b_{c_1}, b_{c_2}, \ldots, b_{c_{n_C}}\}$ subject to $b_{c_i} = 1$ if and only if $c_i \in I$ and boolean vertex variables $\{b_{v_1}, b_{v_2}, \ldots, b_{v_{n_V}}\}$ can* *be assigned values such that* $b_{v_1} = 1$*,* $b_{v_{n_V}} = 0$ *and for all* $(v_i, v_j) \in A$*,* $((v_i, v_j) \in A_{c_k}) \wedge (b_{v_i} = 1) \wedge (b_{c_k} = 0) \Rightarrow (b_{v_j} = 1)$*.*

Proof: First suppose that I is a $v_1 v_{n_V}$-cut in D. Assign values to the boolean variables $\{b_{c_1}, b_{c_2}, \ldots, b_{c_{n_C}}\}$ as described above. Also assign $b_{v_1} = 1$, and if a vertex v_i is reachable from v_1 in $(V, A \setminus A_I, C)$, then assign 1 to b_{v_i}. If a vertex v_j is not reachable from v_1 in $(V, A \setminus A_I, C)$, then assign 0 to b_{v_j}. This assignment satisfies the set of conditions that $b_{v_1} = 1$ and for all $(v_i, v_j) \in A$, if $(v_i, v_j) \in A_{c_k}$, $b_{v_i} = 1$ and $b_{c_k} = 0$, then $b_{v_j} = 1$. Also, because I is a $v_1 v_{n_V}$-cut in D, then v_{n_V} is not reachable from v_1 in $(V, A \setminus A_I, C)$, so $b_{v_{n_V}} = 0$ is satisfied.

Now suppose that I is not a $v_1 v_{n_V}$-cut in D. Assign values to the boolean variables $\{b_{c_1}, b_{c_2}, \ldots, b_{c_{n_C}}\}$ as described above. Also assign $b_{v_1} = 1$, and if a vertex v_i is reachable from v_1 in $(V, A \setminus A_I, C)$, then assign 1 to b_{v_i}. This assignment is necessary and sufficient to satisfy the set of conditions that $b_{v_1} = 1$ and for all $(v_i, v_j) \in A$, $((v_i, v_j) \in A_{c_k}) \wedge (b_{v_i} = 1) \wedge (b_{c_k} = 0) \Rightarrow (b_{v_j} = 1)$. However, because I is not a $v_1 v_{n_V}$-cut in D, then v_{n_V} is reachable from v_1 in $(V, A \setminus A_I, C)$, so the constraint that $b_{v_{n_V}} = 0$ cannot be satisfied if $b_{v_1} = 1$ and for all $(v_i, v_j) \in A$, $((v_i, v_j) \in A_{c_k}) \wedge (b_{v_i} = 1) \wedge (b_{c_k} = 0) \Rightarrow (b_{v_j} = 1)$. ■

Note that because $\{b_{c_1}, b_{c_2}, \ldots, b_{c_{n_C}}\}$ and $\{b_{v_1}, b_{v_2}, \ldots, b_{v_{n_V}}\}$ are boolean variables, $(b_{v_i} = 1) \wedge (b_{c_k} = 0) \Rightarrow (b_{v_j} = 1)$ holds if and only if $-b_{v_i} + b_{v_j} + b_{c_k} \geq 0$, $b_{v_1} = 1$ holds if and only if $b_{v_1} \geq 1$ and $b_{v_{n_V}} = 0$ holds if and only if $-b_{v_{n_V}} \geq 0$. It is now shown how to construct the matrices $\vec{A}$ and $\vec{B}$ and a vector $\vec{x}$ such that $\vec{A}\vec{x} \geq \vec{B}$ if and only if the linear inequalities $b_{v_1} \geq 1$, $-b_{v_{n_V}} \geq 0$ and for all $(v_i, v_j) \in A$, if $(v_i, v_j) \in A_{c_k}$, then $b_{c_k} - b_{v_i} + b_{v_j} \geq 0$ are satisfied.

First, let $\vec{x}$ be the boolean $(n_C + n_V)$-element column vector defined as follows:

$$\vec{x} = \begin{bmatrix} b_{c_1} \\ b_{c_2} \\ \vdots \\ b_{c_{n_C}} \\ b_{v_1} \\ b_{v_2} \\ \vdots \\ b_{v_{n_V}} \end{bmatrix}.$$

To encode the constraints that for all $(v_i, v_j) \in A$, if $(v_i, v_j) \in A_{c_k}$, then $b_{c_k} - b_{v_i} + b_{v_j} \geq 0$, suppose that the edges in A are given an arbitrary ordering and suppose without loss of generality that the mth edge corresponds to the constraint that $b_{c_{k_m}} - b_{v_{i_m}} + b_{v_{j_m}} \geq 0$. Note that the k_mth entry in $\vec{x}$ is $b_{c_{k_m}}$, the $(n_C + i_m)$th entry in $\vec{x}$ is $b_{v_{i_m}}$ and the $(n_C + j_m)$th entry in $\vec{x}$ is $b_{v_{j_m}}$. Construct the $(n_C + n_V)$-element row vector $\vec{A}_m$ such that the k_mth entry in $\vec{A}_m$ is 1, the $(n_C + i_m)$th entry in $\vec{A}_m$ is -1, the $(n_C + j_m)$th entry in $\vec{A}_m$ is 1 and all other entries in $\vec{A}_m$ are 0. Also define the variable B_m to be 0. Note that due to construction of $\vec{A}_m$ and B_m, $\vec{A}_m\vec{x} \geq B_m$ if and only if $b_{c_k} - b_{v_i} + b_{v_j} \geq 0$.

To encode the constraint that $b_{v_1} \geq 1$, note that the $(n_C + 1)$th entry in $\vec{x}$ is b_{v_1}. Therefore, construct the $(n_C + n_V)$-element row vector $\vec{A}_{v_1}$ such that the $(n_C + 1)$th entry in $\vec{A}_{v_1}$ is 1 and all other entries are 0. Also define the variable B_{v_1} to be 1. Therefore $\vec{A}_{v_1}\vec{x} \geq B_{v_1}$ if and only if $b_{v_1} \geq 1$.

Finally, to encode the constraint that $-b_{v_{n_V}} \geq 0$, note that the $(n_C + n_V)$th entry in $\vec{x}$ is $b_{v_{n_V}}$. Therefore, construct the $(n_C + n_V)$-element row vector $\vec{A}_{v_{n_V}}$ such that the $(n_C + n_V)$th entry in $\vec{A}_{v_{n_V}}$ is -1 and all other entries are 0. Also define the variable $B_{v_{n_V}}$ to be 0. Therefore $\vec{A}_{v_{n_V}}\vec{x} \geq B_{v_{n_V}}$ if and only if $-b_{v_{n_V}} \geq 0$.

Note that $\vec{A}_{v_1}\vec{x} \geq B_{v_1}$, $\vec{A}_{v_{n_V}}\vec{x} \geq B_{v_{n_V}}$ and for all $m \in \{1, \ldots, |A|\}$, $\vec{A}_m\vec{x} \geq B_m$ if and only if $b_{v_1} \geq 1$, $-b_{v_{n_V}} \geq 0$ and for all $(v_i, v_j) \in A$, if $(v_i, v_j) \in A_{c_k}$, then $b_{c_k} - b_{v_i} + b_{v_j} \geq 0$. With this in mind, define $\vec{A}$ and $\vec{B}$ as follows:

$$\vec{A} = \begin{bmatrix} \vec{A}_1 \\ \vec{A}_2 \\ \vdots \\ \vec{A}_{|A|} \\ \vec{A}_{v_1} \\ \vec{A}_{v_{n_V}} \end{bmatrix}$$

$$\vec{B} = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_{|A|} \\ B_{v_1} \\ B_{v_{n_V}} \end{bmatrix}.$$

Due to the construction of $\vec{A}$ and $\vec{B}$, $\vec{A}_{v_1}\vec{x} \geq B_{v_1}$, $\vec{A}_{v_{n_V}}\vec{x} \geq B_{v_{n_V}}$ and for all $m \in \{1, \ldots, |A|\}$, $\vec{A}_m\vec{x} \geq B_m$ if and only if $\vec{A}\vec{x} \geq \vec{B}$. This implies the following theorem.

Theorem 9 *Suppose an edge-colored directed graph $D = (V, A, C)$ is used to construct $\vec{A}$ and $\vec{B}$ as described above. A set of colors $I \subseteq C$ is a colored $v_1v_{n_V}$-cut in D if and only if $\vec{A}\vec{x} \geq \vec{B}$ where $b_{c_i} = 1$ if and only if $c_i \in I$ and*

$$\vec{x} = \begin{bmatrix} b_{c_1} \\ b_{c_2} \\ \vdots \\ b_{c_{n_C}} \\ b_{v_1} \\ b_{v_2} \\ \vdots \\ b_{v_{n_V}} \end{bmatrix}.$$

Proof: Due to the construction of $\vec{A}$ and $\vec{B}$, $\vec{A}_{v_1}\vec{x} \geq B_{v_1}$, $\vec{A}_{v_{n_V}}\vec{x} \geq B_{v_{n_V}}$ and for all $m \in \{1, \ldots, |A|\}$, $\vec{A}_m\vec{x} \geq B_m$ if and only if $\vec{A}\vec{x} \geq \vec{B}$. Due to the constructions of $\vec{A}_{v_1}$, B_{v_1}, $\vec{A}_{v_{n_V}}$, $B_{v_{n_V}}$, and for all $m \in \{1, \ldots, |A|\}$, $\vec{A}_m$ and B_m, $\vec{A}_{v_1}\vec{x} \geq B_{v_1}$, $\vec{A}_{v_{n_V}}\vec{x} \geq B_{v_{n_V}}$ and for all $m \in \{1, \ldots, |A|\}$, $\vec{A}_m\vec{x} \geq B_m$ if and only if $b_{v_1} \geq 1$, $-b_{v_{n_V}} \geq 0$ and for all $(v_i, v_j) \in A$, if $(v_i, v_j) \in A_{c_k}$, then $b_{c_k} - b_{v_i} + b_{v_j} \geq 0$. Because of Lemma 4, I is a $v_1v_{n_V}$-cut in D if and only if the boolean variables in $\vec{x}$ are assigned values such that $b_{c_i} = 1$ if and only if $c_i \in I$, $b_{v_1} = 1$, $b_{v_{n_V}} = 0$ and for all $(v_i, v_j) \in A$, $((v_i, v_j) \in A_{c_k}) \wedge (b_{v_i} = 1) \wedge (b_{c_k} = 0) \Rightarrow (b_{v_j} = 1)$. ■

Because of Theorem 9, if the set of colors $I \subseteq C$ corresponds to a set of boolean variables $\{b_{c_1}, b_{c_2}, \ldots, b_{c_{n_C}}\}$ such that $b_{c_i} = 1$ if and only if $c_i \in I$, then the minimal cost colored cut problem is to find the the set $I \subseteq C$ that minimizes $\sum_{i=1}^{n_C} b_{c_i} cost(c_i)$ subject to $\vec{A}\vec{x} \geq \vec{B}$.

Now define the $(n_C + n_V)$-element row vector $\vec{C}$ as follows:

$$\vec{C} = \begin{bmatrix} cost(c_1) & cost(c_1) & \cdots & cost(c_{n_C}) & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}.$$

Note that $\vec{C}\vec{x} = \sum_{i=1}^{n_C} b_{c_i} cost(c_i)$. The above definitions imply the following corollary of Theorem 9.

Corollary 4 *For the constructions of $\vec{A}$, $\vec{B}$ and $\vec{C}$ from $D = (V, A, C)$ and some constant k, a boolean vector $\vec{x}$ subject to the constraint that $\vec{A}\vec{x} \geq \vec{B}$*

exists such that $\vec{C}\vec{x} = k$ if and only if there is a colored cut $I \subseteq C$ is a $v_1 v_{n_V}$-cut in D subject to $\sum_{i=1}^{n_C} b_{c_i} cost(c_i) = k$ where in $\vec{x}$, $(b_{c_i} = 1) \iff (c_i \in I)$.

An example of the construction of the integer programming matrices from an instance of the minimal cost colored cut problem is now given.

Example 5 *Consider the edge-colored directed graph seen in Figure 9.*

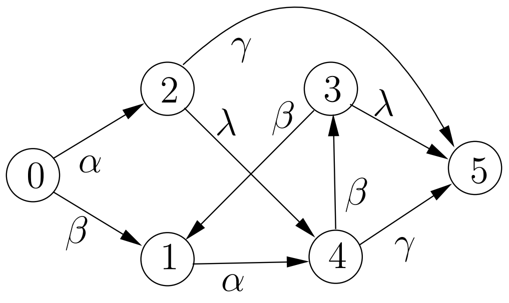

Figure 9: An example of a colored graph.

With the directed graph in Figure 9, let:

$$\vec{x} = \begin{bmatrix} c_\alpha \\ c_\beta \\ c_\gamma \\ c_\lambda \\ r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{bmatrix}.$$

Suppose the edges in the directed graph in Figure 9 are given the arbitrary ordering $(0,1),(3,1),(0,2),(4,3),(1,4),(2,4),(2,5),(3,5),(4,5)$. Also note that $A_\alpha = \{(0,2),(1,4)\}$, $A_\beta = \{(0,1),(3,1),(4,3)\}$, $A_\gamma = \{(2,5),(4,5)\}$ and $A_\lambda = \{(2,4),(3,5)\}$.

With the above ordering on the edges and the construction methods given above, the following assignments are made to the integer programing matri-

ces:

$$\vec{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\vec{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\vec{C} = \begin{bmatrix} cost(\alpha) & cost(\beta) & cost(\gamma) & cost(\lambda) & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

8 Discussion

This paper has discussed the approximation properties of a computationally difficult sensor selection problem in supervisory control. This sensor selection problem is shown to be related to a type of edge colored directed graph *st*-cut problem not previously discussed in computer science. Solutions to both the sensor selection and graph cutting problems are difficult to approximate, but a $|V|^{2/3}$ approximation algorithm is shown for a special case of these problems. It is also shown how to convert the sensor selection and graph cutting problems into an integer programming problem.

Note that the methods shown here to deal with the sensor selection problem allows the control designer to force some events to be observable or unobservable. First, the cost function could appropriately adjusted to force an event to be cost free to observe, or, conversely, to make an event prohibitively expensive to observe. Secondly, during the construction of $\mathcal{M}_{\Sigma_o}$, an event σ could be forced to be made unobservable by converting all instances of transitions labeled by σ' to transitions labeled by σ. Then, during the conversion of this modified $\mathcal{M}_{\Sigma_o}$ automaton to $\tilde{\mathcal{M}}_{\Sigma_o}$, no transitions corresponding to σ could be cut in $\tilde{\mathcal{M}}_{\Sigma_o}$ to simulate making σ observable.

Although this paper shows how the theory of approximation algorithms can be applied in a formal manner for the analysis of problems in supervisory control, this paper has not discussed heuristic approaches for approximations as in [15, 16]. Heuristic approaches may perform horribly for a few particular problem instances, but work very well for most other "average case" problems. However, the use of heuristic methods for supervisory control problems such as the ones discussed in this paper is still being explored and research is ongoing.

Note that if there is a cost associated with controlling an event, the problem of finding the minimal cost set of controllable events to make the system controllable with respect to a specification language is computable in polynomial time. This is because for a given system and specification there is a unique infimal set of controllable events that make the system controllable [16]. However, the dual of the sensor selection problem for finding the minimal cost set of controllable events to make the system controllable with respect to a sublanguage of a specification language is computationally equivalent to the sensor selection problem discussed in this paper [16]. This is because it is not known which sublanguage of the specification language is proper to calculate the set of controllable events with respect to. There are also extensions of the sensor selection problem to the case of decentralized control [17].

9 Acknowledgment

This paper reports on work commenced while the first author was under the direction of Jan H. van Schuppen at CWI in Amsterdam. The first author would also like to acknowledge the helpful comments and discussions from Stéphane Lafortune of The University of Michigan. Financial support in part for the investigation was made available by the European Commission through the project Control and Computation (IST-2001-33520) of the Information Society Technologies Program and by NSF grants CCR-0082784 and CCR-0325571.

References

[1] S. Arora and C. Lund. Hardness of approximations. In D.S. Hochbaum, editor, *Algorithms for NP-Hard Problems*, pages 46–93. PWS Publishing, 1997.

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kahn, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation Combinatorial Optimization Problems and Their Approximability Properties.* Springer-Verlag, Berlin Heidelberg, 1999.

[3] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kluwer Academic Publishers, Boston, MA, 1999.

[4] H. Cho. Designing observation functions in supervisory control. In *Proc. Korean Automatic Control Conference*, pages 523–528, 1990.

[5] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, 1979.

[6] A. Haji-Valizadeh and K.A. Loparo. Minimizing the cardinality of an events set for supervisors of discrete-event dynamical systems. *IEEE Trans. Auto. Contr.*, 41(11):1579–1593, November 1996.

[7] S. Jiang, R. Kumar, and H.E. Garcia. Optimal sensor selection for discrete-event systems with partial observation. *IEEE Trans. Auto. Contr.*, 48(3):369–381, 2003.

[8] S. Khuller, G. Kortsarz, and K. Rohloff. Approximating the minimal sensor selection for supervisory control. In *Proc. 7th Workshop on Discrete Event Systems*, Reims, France, September 2004. To appear.

[9] G. Kortsarz. On the hardness of approximating spanners. In *The First International Workshop on Approximation*, pages 135–146, 1998.

[10] F. Lin and W. M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44:173–198, 1988.

[11] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982.

[12] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete-event processes. *SIAM Journal of Control Optimization*, 25(1):206–230, 1987.

[13] P.J. Ramadge and W.M. Wonham. The control of discrete-event systems. *Proc. IEEE*, 77(1):81–98, 1989.

[14] R. Raz. A parallel repetition theorem. *SIAM Journal of Computing*, 27(3):763–803, 1998.

[15] K. Rohloff. *Computations on Distributed Discrete-Event Systems.* PhD thesis, The University of Michigan, Ann Arbor, 2004. Available at `http://www.eecs.umich.edu/umdes`.

[16] K. Rohloff and J. H. van Schuppen. Approximating the minimal-cost sensor-selection for discrete-event systems. Technical Report MAS-R0404, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, the Netherlands, December 2004.

[17] K. Rohloff and J. H. van Schuppen. Approximating minimal communicated event sets for decentralized supervisory control. In *Proc. of the 16th IFAC World Congress*, 2005.

[18] K. Rudie and J.C. Willems. The computational complexity of decentralized discrete-event control problems. *IEEE Trans. Auto. Contr.*, 40(7):1313–1318, 1995.

[19] J. Tsitsiklis. On the control of discrete-event dynamical systems. *Mathematics of Control, Signals and Systems*, 2:95–107, 1989.

[20] V. Vazirani. *Approximation Algorithms.* Springer-Verlag, Berlin Heidelberg, 2001.

[21] T.-S. Yoo and S. Lafortune. NP-completeness of sensor selection problems arising in partially-observed discrete-event systems. *IEEE Trans. Auto. Contr.*, 47(9):1495–1499, 2002.

[22] S. Young and V. K. Garg. Optimal sensor and actuator choices for discrete event systems. In *Proc. 31st Allerton Conference on Communication, Control, and Computing*, October 1994.