2008 American Control Conference
Westin Seattle Hotel, Seattle, Washington, USA
June 11-13, 2008

Optimal Parametric Discrete Event Control: Problem and Solution

Christopher Griffin
Oak Ridge National Laboratory
P.O. Box 2008 MS 6418
Oak Ridge, TN 37831
E-mail: cgriffin229@yahoo.com

***Abstract*— We present a novel optimization problem for discrete event control, similar in spirit to the optimal parametric control problem common in statistical process control. In our problem, we assume a known finite state machine plant model G defined over an event alphabet Σ so that the plant model language $L = \mathcal{L}_{\mathrm{M}}(G)$ is prefix closed. We further assume the existence of a *base control structure* M_K, which may be either a finite state machine or a deterministic pushdown machine. If $K = \mathcal{L}_{\mathrm{M}}(M_K)$, we assume K is prefix closed and that $K \subseteq L$.**

We associate each controllable transition of M_K with a binary variable $X_1, \ldots, X_n$ indicating whether the transition is enabled or not. This leads to a function $M_K(X_1, \ldots, X_n)$, that returns a new control specification depending upon the values of $X_1, \ldots, X_n$. We exhibit a branch-and-bound algorithm to solve the optimization problem $\min_{X_1,\ldots,X_n} \max_{w \in K} C(w)$ such that $M_K(X_1, \ldots, X_n) \models \Pi$ and $\mathcal{L}_{\mathrm{M}}(M_K(X_1, \ldots, X_n)) \in \mathcal{C}(L)$. Here Π is a set of logical assertions on the structure of $M_K(X_1, \ldots, X_n)$, and $M_K(X_1, \ldots, X_n) \models \Pi$ indicates that $M_K(X_1, \ldots, X_n)$ satisfies the logical assertions; and, $\mathcal{C}(L)$ is the set of controllable sublanguages of L[1].

I. Introduction

The supervisory control theory (SCT) for discrete event dynamic systems (DEDS) was introduced by Ramadge and Wonham [1]. The initial work on regular languages was extended in many subsequent publications. Examples of these extensions can be found in [2]–[14]. In SCT a plant model is given by a finite state machine $G = \langle Q, \Sigma, \delta, q_0, Q_f \rangle$. Here, Q is a finite set of states; Σ is the finite event alphabet with $\Sigma = \Sigma_c \cup \Sigma_u$; q_0 is the start state; Q_f are the final states and $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation. The events in Σ_c may be disabled by a supervisor and hence are called controllable. A second machine, M_K may be run in parallel with the plant G, thus controlling the stream of symbols generated by G. A complete description of SCT may be found in [1] or [15].

[12] extends the SCT to the case when the controller machine M_K is given by a deterministic pushdown machine (DPDM). Informally, a pushdown machine (PDM) is a finite state machine augmented with an infinite stack memory. Formally, A pushdown machine is a tuple $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, Q_f \rangle$, where Γ is a stack alphabet, Z_0 is the initial symbol on the stack and $\delta \subseteq Q \times \Sigma \cup \{\epsilon\} \times \Gamma \times Q \times \Gamma^*$ is the transition relation in the PDM. A PDM is deterministic if the following hold:

1) If $(q_1, a, Z, q_2, \gamma) \in \delta$ and $(q_1, a, Z, q_2', \gamma') \in \delta$, then $q_2 = q_2'$ and $\gamma = \gamma'$.
2) If $(q_1, \epsilon, Z, q_2, \gamma) \in \delta$, then for all $a \in \Sigma$, $\delta(q_1, a, Z) = \emptyset$.

A complete description of the use of DPDM in control can be found in [12] and [14].

By $\mathcal{L}_{\mathrm{M}}(G)$ we mean the set of strings (language) that cause G to transition from its start state (and stack configuration, if applicable) to a final state–this is valid for both FSM and DPDM. Let $L = \mathcal{L}_{\mathrm{M}}(G)$ we denote the prefix closure of L by $\overline{L}$ and say that a language is prefix closed if $L = \overline{L}$. Details of these terms may be found in [1], [15] and [12]. Let M_K be a controller machine and let G be a plant model if $K = \mathcal{L}_{\mathrm{M}}(M_K)$ and $L = \mathcal{L}_{\mathrm{M}}(G)$, we say that K is controllable with respect to G if $\overline{K}\Sigma_u^* \cap \overline{L} = \overline{K}$ [15]. The concept of controllability is central to the study of the SCT.

The remainder of this paper is organized as follows: In Section II we lay out the optimal control problem we are attempting to solve. In Section III we provide a branch-and-bound algorithm for solving this problem. In Section IV we provide an example of our algorithm. In Section V we discuss how our problem relates to previous optimal control problems considered in the literature. Finally, we provide conclusions and future directions in Section VI.

II. Problem Statement

Assume we are given a finite state machine plant model G defined over an event alphabet Σ so that the plant model language $L = \mathcal{L}_{\mathrm{M}}(G)$ is prefix closed. Further, suppose we have a function $M_K(X_1, \ldots, X_n)$ that returns a collection of machines (either DPDM or FSM) so that $\mathcal{L}_{\mathrm{M}}(M_K(X_1, \ldots, X_n)) \subseteq L$ is prefix closed as well. We study the following problem:

$$\begin{aligned} \min_{X_1,\ldots,X_n} & \max_{w \in K} C(w) \\ \text{s.t. } & M_K(X_1, \ldots, X_n) \models \Pi \\ & K = \mathcal{L}_{\mathrm{M}}(M_K(X_1, \ldots, X_n)) \\ & K \in \mathcal{C}(L) \\ & (X_1, \ldots, X_n) \in \mathbb{B} \end{aligned} \tag{1}$$

In the following sections we define the objective function, the function $M_K(X_1, \ldots, X_n)$ and the constraints $M_K(X_1, \ldots, X_n) \models \Pi$ and $K \in \mathcal{C}(L)$.

[1]A portion of this work was created as a Eugene P. Wigner Fellow and staff member at the Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U.S. Department of Energy under Contract DE-AC05-00OR22725.

A. Objective Function

In [8] Sangupta and LaFortune proposed the minimax objective function for a regular language L. For each event, the authors assumed an enabling cost $\kappa_\alpha(a)$ and an execution cost for each event $a \in \Sigma$, $\kappa_\rho(a)$. Given a string w accepted by a finite state machine G and a supervisor φ, the cost of the string w could be computed by inductively determining which events were enabled during string execution and which events in w had actually been executed. A detailed account of this cost function is described in [8]. We will not reiterate the details since they are expertly defined in this reference.

It is clear that a similar definition extends readily to deterministic pushdown machines. In order to guarantee a convergent optimization theory, Sangupta and LaFortune assumed that all closed loops in the given plant model G had zero net cost. (Otherwise, when considering arbitrarily many loops, the controller could easily produce an infinite cost.) We do not make this restriction.

Instead, we apply a depreciation constant β to each string. Let $w = a_1 \cdots a_n$ be a string and M_K be a FSM/DPDM. Define

$$\kappa_\alpha(M_K|w(i-1)) = \sum_{a\in\Sigma, w(i-1)a\in\mathcal{L}_{\mathrm{M}}(M_K)} \kappa_\alpha(a)$$

where $w(i-1)$ is the string $a_1 a_2 \cdots a_{i-1}$ and we define $w(0) = \epsilon$. Then the cost of string w is:

$$C(w) = \sum_{i=1}^{n} \beta^{i-1}(\kappa_\alpha(M_K|w(i-1)) + \kappa_\rho(a_i)) \quad (2)$$

Trivially for $\beta \in [0, 1)$, there is a supremal value C^* such that for all $w \in L$, $c(w) \leq C^*$. Hence we need not concern ourselves with questions of convergence.

Remark 1: One *negative* point about abandoning Sangupta and LaFortune's approach is the effect a short term time horizon has on the overall objective function. We appreciate that this violates the principle of infinite operation embedded in the controllability predicate, however we counter that many real world optimal control problems often contain either a finite time horizon, (in which case strict controllability is not needed) or they use a depreciation constant in an infinite time horizon. Therefore, we do *not* feel that this is in anyway detrimental to our theory. Further, we believe it opens a new avenue of research for those interested in pursuing an extension of the results we present here with $\beta = 1$.

B. The Function $M_K(X_1, \ldots, X_n)$

In our formulation, let M_K be a machine providing a prefix closed sublanguage of $\mathcal{L}_{\mathrm{M}}(G)$, for the fixed plant model G. Suppose that M_K has controllable transitions $\tau_1, \ldots, \tau_n$. For binary variable X_i, we have $X_i = 1$ if and only if τ_i is enabled in M_K. Hence, we may define $M_K(X_1, \ldots, X_n)$ to be the machine obtained from M_K by enabling or disabling the appropriate transitions. Clearly, $M_K = M_K(1, 1, \ldots, 1)$.

C. Constraint: $M_K(X_1, \ldots, X_n) \models \Pi$

For the remainder of this paper, we will assume that the sentences of Π are formed from the logical language [16]:

$$\langle \{q_0, \ldots, q_n, \sigma_1, \ldots, \sigma_m, Z_0, \ldots, Z_k\} \cup \{\delta, Q_f\}, \emptyset, \in \rangle \quad (3)$$

That is to say, logical sentences may use constants (nouns) from $Q = \{q_0, \ldots, q_n\}$, $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$, $\Gamma = \{Z_0, \ldots, Z_k\}$ and the sets Q_f and δ and they may relate these two elements together using the $\in$ relation. The empty set symbol indicates that there are no functions in this logic. An example sentence,

$$\exists p((q_0, \sigma_1, Z_0, p, Z_0) \in \delta \wedge (q_0, \sigma_2, Z_0, p, Z_0) \notin \delta)$$

is a simple sentence saying there is a transition from the start state q_0 on input σ_1 with top stack symbol Z_0 to state p, but no such transition on input symbol σ_2.

Clearly, if we restrict our attention to this class of sentences Π, then for all functions $M_K(X_1, \ldots, X_n)$ whose range consists of FSM/DPDM, $M_K(X_1, \ldots, X_n) \models \Pi$ is decidable for fixed $X_1, \ldots, X_n$.

Let $\psi_1, \ldots, \psi_m$ be a set of sentences in the logic given in Equation 3. We will say that $M_K(X_1, \ldots, X_n) \models \Pi$ for fixed values of $X_1, \ldots, X_n$ if and only if each sentence in Π is true in the automaton $M_K(X_1, \ldots, X_n)$. That is, the graph structure of M_K satisfies sentences in Π [17].

Suppose that ψ is a sentence Π and fix values for $X_1, \ldots, X_n$ Suppose that $M_K(X_1, \ldots, X_n) \not\models \psi$; that is, ψ is not true in $M_K(X_1, \ldots, X_n)$. We will say that ψ is *reachable* from $M_K(X_1, \ldots, X_n)$ if there is a finite set of variables $X_{i_1} = \cdots = X_{i_k} = 1$ so that when we set $X_{i_1} = \cdots = X_{i_k} = 0$, then $M_K(X_1, \ldots, X_n) \models \psi$. Alternatively, ψ is reachable if there is a finite set of controllable transitions $\tau_1, \ldots, \tau_k$ in $M_K(X_1, \ldots, X_n)$ that can be disabled and thus make the new structure satisfy ψ. The concept of reachability will be important in our branch-and-bound optimization algorithm.

D. Constraint: $K \in \mathcal{C}(L)$

By $K \in \mathcal{C}(L)$ we simply require that the resulting language of the supervised system be controllable with respect to the plant language $L = \mathcal{L}_{\mathrm{M}}(G)$.

III. Branch-and-Bound Algorithm

We present a branch-and-bound algorithm to solve Problem (1). A driving subroutine of our algorithm is Kumar's supremal controllable sublanguage identification algorithm [4] or Griffin's variation of Kumar's algorithm for DPDM controllers.

A. Supremal Controllable Sublanguage Computation

In [2] Ramadge and Wonham introduced the notion of the supremal controllable sublanguage. Let $K = \mathcal{L}_{\mathrm{M}}(M_K)$ and let $L = \mathcal{L}_{\mathrm{M}}(G)$. If K is not controllable with respect to L, then the supremal controllable sublanguage of K with respect to L is the language $\sup_{\mathrm{C}}(K) \subseteq K$ such that (i) $\sup_{\mathrm{C}}(K)$ is controllable with respect to L and (ii) if $K' \subseteq$

K and K' is controllable with respect to L, then $K' \subseteq \sup_{\mathrm{C}}(K)$.

[1] showed that when K and L are both generated by finite state machines, then so is $\sup_{\mathrm{C}}(K)$. In [4], Kumar et al. provided an elegant algorithm to modify the structure of M_K to generate $\sup_{\mathrm{C}}(K)$ when M_K and G are both finite state machines and K and L are prefix closed. [14] showed how to extend Kumar's algorithm to the case when K is generated by a DPDM and L is generated by a finite state machine and again, K and L are prefix closed. Note, if $\sup_{\mathrm{C}}(K) = \emptyset$, we will say that these two algorithms return $\emptyset$. Otherwise, they return a new machine M'_K that generates $\sup_{\mathrm{C}}(K)$.

We will make use of one of these two algorithms (depending upon whether M_K is a FSM or DPDM) in the branch-and-bound algorithm presented in the next section. We refer to this algorithm as the Kumar/Griffin Algorithm and assume the reader will know to use Griffin's variation when M_K is a DPDM.

B. The Algorithm

We provide a depth first search branch-and-bound algorithm below. At each node of the branch-and-bound tree, we will have a set of fixed variable indexes, $\mathcal{I}$ (for immutable). These are variables that are fixed in value and cannot be modified.

The algorithm we provide operates as follows. Initially, all variables are initialized to 1. This means that every controllable transition in M_K is enabled. We test to see whether the parametric control problem is solvable with $M_K(X_1, \ldots, X_n)$. If not, then we stop. We also test to see if Π is reachable; if not we stop. In each case, we stop because we've determined we can never find a substructure of $M_K(X_1, \ldots, X_n)$ that is controllable with respect to the plant model and that satisfies Π. We use Algorithm $\mathfrak{D}$ to solve the parametric control problem. If Algorithm $\mathfrak{D}$ disables some transitions in $M_K(X_1, \ldots, X_n)$, then we identify the new values for $X_1, \ldots, X_n$. If possible, we determine an incumbent solution. At this point, we are ready to descend. We choose a (non-zero) variable (say X_1) and set it to zero. This *disables* a transition in the base structure M_K. We add 1 to the immutable set $\mathcal{I}$. With 1 in $\mathcal{I}$ this means we cannot change the value of X_1 at any node *below* level 1 of the tree.

We descend down the tree repeating the testing process above and also checking for feasible solutions that yield an objective function value that is smaller than the current incumbent solution. When we cannot set any more variables to 0, we climb back up the tree and change the X_i variables that were set to 0 to 1; we then descend with these values fixed at 1. Ultimately, we will return to the root node and X_1 will be changed from 0 to 1. The process then repeats with $X_1 = 1$ (and immutable). A completely expanded search tree is shown in Figure 1

In the main body of the algorithm, there are a few caveats. First, if Algorithm $\mathfrak{D}$ attempts to disable a transition corresponding to a variable with index in the immutable set, then search along this branch is discontinued. This is because we can be certain we have already explored this structure–otherwise, the index would not be both in the immutable set and have its corresponding variable set to 1. Second, we have devised a cost bound based on the fact we are solving a minimax problem that allows us to stop exploring a branch when we are certain no improved solution can be obtained.

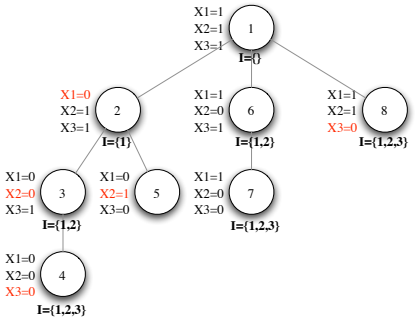

Fig. 1. A fully expanded tree; nodes are labeled in order they will be investigated. The immutable set is shown below each node.

In short, the algorithm shown in Algorithm III.1. is an exhaustive search algorithm that implicitly enumerates some of the nodes in the search when it is clear they will not yield an improved solution. This fact is shown in Theorem 3.

C. Proof of Convergence

Lemma 2: Let N_1 and N_2 be two nodes in the branch-and-bound tree produced by Algorithm III.1 and suppose that M_K^1 and M_K^2 are the controllable substructures of M_K generated after Step 2 for these two nodes respectively. Finally, suppose that N_1 is the parent of N_2. Then:

1) $\min_{w \in \mathcal{L}_{\mathrm{M}}(M_K^1)} C(w) \leq \min_{w \in \mathcal{L}_{\mathrm{M}}(M_K^2)} C(w)$ and
2) $\max_{w \in \mathcal{L}_{\mathrm{M}}(M_K^1)} C(w) \geq \max_{w \in \mathcal{L}_{\mathrm{M}}(M_K^2)} C(w)$.

Proof: In obtaining M_K^2 from M_K^1, we remove a transition. This effectively removes a collection of strings from $\mathcal{L}_{\mathrm{M}}(M_K^1)$ to obtain $\mathcal{L}_{\mathrm{M}}(M_K^2)$. If the strings producing a maximal cost were removed, then it follows that $\max_{w \in \mathcal{L}_{\mathrm{M}}(M_K^1)} C(w) > \max_{w \in \mathcal{L}_{\mathrm{M}}(M_K^1)} C(w)$; otherwise $\max_{w \in \mathcal{L}_{\mathrm{M}}(M_K^1)} C(w) = \max_{w \in \mathcal{L}_{\mathrm{M}}(M_K^1)} C(w)$. Likewise, if the strings producing a minimal cost were removed, then it follows that $\min_{w \in \mathcal{L}_{\mathrm{M}}(M_K^1)} C(w) < \min_{w \in \mathcal{L}_{\mathrm{M}}(M_K^1)} C(w)$; otherwise it is immediately clear that $\min_{w \in \mathcal{L}_{\mathrm{M}}(M_K^1)} C(w) = \min_{w \in \mathcal{L}_{\mathrm{M}}(M_K^1)} C(w)$. ■

Proposition 3: Given G and M_K, if there is a substructure M'_K of M_K that is controllable, then Algorithm III.1 will find it. Further, Algorithm III.1 will find the values of $X_1, \ldots, X_n$ that solve Problem 1.

Proof: It suffices to show that Algorithm III.1 will implicitly enumerate all possible substructures of M_K that can be obtained by disabling controllable transitions. Trivially, in Step 1 of the branching process, we remove a transition and add this transition to the immutable transitions at node N–we do this by adding index i to the immutable set. If we return to Step 1 at Node N, this transition will never be removed again in any child node of N because the immutable set

Algorithm Description III.1 – Branch-and-Bound

Initialization: Given $M_K(X_1, \ldots, X_n)$ and G, initialize $X_1 = X_2 = \cdots = X_n = 1$.

1) If Π is not reachable from $M_K(X_1, \ldots, X_n)$ then stop–there is no feasible solution. If $\mathcal{L}_{\mathrm{M}}(M_K(X_1, \ldots, X_n))$ is controllable with respect to $\mathcal{L}_{\mathrm{M}}(G)$, GOTO 3.
2) If $M_K(X_1, \ldots, X_n)$ is not controllable with respect to G, apply the Kumar/Griffin Algorithm. If the Kumar/Griffin Algorithm returns $\emptyset$, STOP, the problem is infeasible. Otherwise, let A be the machine output from the Kumar/Griffin Algorithm. Use A to determine current values of $X_1, \ldots, X_n$. If Π is reachable from $M_K(X_1, \ldots, X_n)$, then GOTO 3. Otherwise, STOP, the problem is infeasible.
3) If $M_K(X_1, \ldots, X_n)$ satisfies Π, then set the incumbent objective value C_0 to the objective value of $\mathcal{L}_{\mathrm{M}}(M_K)$. Declare the incumbent solution to be $X_1^0 = X_1, \ldots, X_n^0 = X_n$. Otherwise, set $C_0 = \infty$ and set the incumbent solution to NULL.
4) Initialize $\mathcal{I}$ to the set of variable indices of variables that are currently zero.

Branching Step: Suppose we have arrived at node N with $X_1, \ldots, X_n$ and immutable set $\mathcal{I}$.

1) Let $Y_1 = X_1, \ldots, Y_n = X_n$. Choose i from 1 to n such that i is not in $\mathcal{I}$ and set $Y_i = 0$. (Effectively, we will delete a transition τ from $M_K(X_1, \ldots, X_n)$.) Move i to $\mathcal{I}$. If no such variable is available, then fathom this node.
2) If $M_K(Y_1, \ldots, Y_n)$ is controllable with respect to $\mathcal{L}_{\mathrm{M}}(G)$ then GOTO 4. Otherwise GOTO 3.
3) Apply the Kumar/Griffin Algorithm $M_K(Y_1, \ldots, Y_n)$. If the Kumar/Griffin Algorithm returns $\emptyset$, then GOTO 1. Otherwise, let A be the machine output from the Kumar/Griffin Algorithm. Use A to determine current values of $Y_1, \ldots, Y_n$. If any variable Y_j has been set to zero, but j is in $\mathcal{I}$ then GOTO 1. Otherwise, add the indices of any of the $Y_1, \ldots, Y_n$ that were set to zero by the the Kumar/Griffin Algorithm to $\mathcal{I}$. GOTO 4.
4) Test that Π is reachable from $M_K(Y_1, \ldots, Y_n)$. If not, then GOTO 1. Otherwise GOTO 5.
5) If $\min_{w \in \mathcal{L}_{\mathrm{M}}(M_K(Y_1, \ldots, Y_n))} C(w) > C_0$, then GOTO 1. Otherwise GOTO 6.
6) If Π is satisfied by $M_K(Y_1, \ldots, Y_n)$, and $C_0 > \max_{w \in \mathcal{L}_{\mathrm{M}}(M_K(Y_1, \ldots, Y_n))} C(w)$, then set $C_0 = \max_{w \in \mathcal{L}_{\mathrm{M}}(M_K(Y_1, \ldots, Y_n))} C(w)$ and set $X_1^0 = Y_1, \ldots, X_n^0 = Y_n$. GOTO 7.
7) Create a branch with $Y_1, \ldots, Y_n$ and a **copy** of the current set $\mathcal{I}$ from node N. GOTO 1 on $X_1 = Y_1, \ldots, X_n = Y_n$ and $\mathcal{I}'$.

Convergence: When the algorithm is finished execution, either $X_1^0, \ldots, X_n^0$ is the optimal solution or the problem will have been identified as infeasible.

propagates down the tree as shown in Step 7. Hence, at least we know that all branches are independent and if we only executed these two operations we would eventually consider all possible ways of disabling controllable transitions in M_K.

In Steps 2 and 3, if neither $M_K(Y_1, \ldots, Y_n)$ nor A is controllable, then by the results of [4] and [14], we know that no substructure of $M_K(Y_1, \ldots, Y_n)$ can ever be controllable. Hence, we may implicitly enumerate all child nodes that occur below this node by fathoming this node. On the other hand, suppose that an index in I (the immutable set) is removed in creating A, then by the independence of the branches, this structure has already been enumerated in an earlier branch and we may ignore it.

Finally, if A is controllable and no immutable variables have been changed, then we know by the results of [4] and [14] that we have removed the minimum number of transitions from $M_K(Y_1, \ldots, Y_n)$ to achieve controllability and we may continue.

In Step 4, if Π is unreachable from $M_K(Y_1, \ldots, Y_n)$, then there is no combination of transition disabling that can force our decidable predicate constraints to be true along the remainder of the current branch. Hence, we may fathom away all further branches resulting from this node.

We have shown in Lemma 2 that minimal cost increases as the tree descends. Hence, if the minimal cost of a string in $M_K(Y_1, \ldots, Y_n)$ is greater then the current incumbent cost, this branch can yield no better solution and all subsequent nodes can be immediately fathomed.

In Step 6, we identify new incumbent solutions. Hence, whenever a new controllable and feasible substructure $M_K(Y_1, \ldots, Y_n)$ is identified with lower maximal cost than the current incumbent, it is deemed the new incumbent solution.

Thus we have shown that in the absence of Steps 2-4 of the branching process in Algorithm III.1 we would enumerate all possible substructures of M_K. We then showed that Steps 2-5 allow us to implicitly enumerate a number of branching steps, thus reducing the algorithm running time. Hence, it follows that if Algorithm III.1 returns a solution it is optimal and controllable and satisfies Π. Thus, it must be the solution to Problem 1. ■

D. Discussion of Algorithm Complexity

It is clear that the branch-and-bound algorithm provided is NP-complete. Checking for controllability is known to be polynomial when both the plant and controller are finite state machines. It can be shown that for the case when the proposed control language K is accepted by a deterministic pushdown machine, that the problem of deciding controllability (and of computing the supremal controllable sublanguage) is P-complete [18]; meaning that for arbitrarily chosen K, any polynomial problem can be reduced to it.

In the worst case, deciding whether $M_K(X_1, \ldots, X_n) \vdash \Pi$ is also NP-complete since it reduces to the satisfiability problem, which is known to be NP-complete. The power of the algorithm comes in the judicious use of the sentences of Π, which can significantly reduce the search space associated

with the problem. Additionally, if supremal controllable sublanguage identification algorithms are available for non-prefix closed languages, the minimum string cost test is effective at creating elmininating a number of non-optimal solutions [18].

IV. Example

In this section, we show a simple example of the branch-and-bound tree produced when analyzing a small controller. For simplicity, we assume the plant language $L = \mathcal{L}_{\mathrm{M}}(M_K(1,1,\ldots,1))$ and that $M_K(1,1,\ldots,1) = M$, the specification. That is, we are interested in finding a substructure of the plant model that acts as an optimal controller. We assume $\Sigma_c = \{b_1, b_2, c_1, c_2\}$.

The plant (and base controller $M_K(1,1,\ldots,1)$) is shown in Figure 2

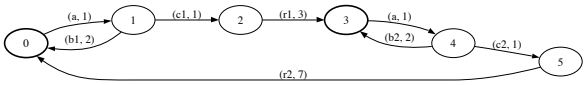

Fig. 2. The controller M_K that begins the branch-and-bound process.

We assume the following sentences are in Π, which we write in English for simplicity:

1) At least one of the transitions $(1, b_1, 0)$ or $(1, c_1, 2)$ is enabled.
2) At least one of the transitions $(3, b_2, 4)$ or $(4, c_2, 5)$ is enabled.

We have appended cost information to the transitions in M_K. There are four variables $X_1, \ldots, X_4$ where X_1 corresponds to the transition labeled b_1 connecting State 1 and State 0; X_2 corresponds to the transition labeled c_1 connecting State 1 with State 2; X_3 corresponds to the transition connecting State 4 with State 3 labeled b_2; and X_4 corresponds to the transition connecting State 4 with State 5 labeled c_2. The resulting branch-and-bound tree is shown in Figure 3. In the branch-and-bound tree, we show the variable values to the left of the tree nodes and we show the corresponding transition that is removed along the edges of the tree. The shade of the node indicates whether the node is feasible or fathomed. If it is feasible, the shade indicates whether a new incumbent solution is defined at this point. If the node is infeasible, the shade indicates why the node was fathomed. The resulting solution is shown in Figure 4

V. Previous Work

Optimal control of discrete event systems has been investigated by a number of authors using various techniques. Optimal control of discrete event systems was originally investigated by Passino and Antsaklis [19]. This approach modifies the basic assumptions of the Supervisory Control Theory of Ramadge and Wonham by assuming a forced event model instead of an enabled event model. Kumar and Garg [20] were the first to formally study optimal supervisory control using the enabled event paradigm. An alternate formulation of optimal control in discrete event systems is in

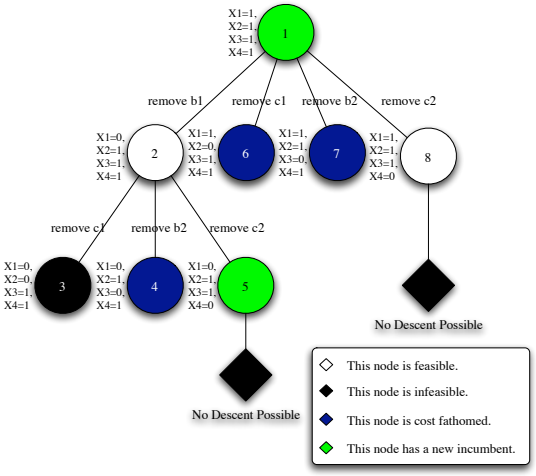

Fig. 3. The branch-and-bound tree resulting from Algorithm III.1 when running on the problem given above.

Fig. 4. Solution to problem given above.

Brave and Heymann [21]. In their formulation, they attempt to minimize the cost of keeping a state machine within a finite set of states. Brave and Heynmann do not include a control cost (a cost for disabling an event) and hence there approach is not as general as [20]. The most general study of optimal control supervisory systems is undertaken by Sengupta and Lafortune [8]. They consider only *non-blocking* supervisors. This work is carried on by Marchand, Boivineau and LaFortune in [9]. The latest investigation into optimal control of discrete event systems was undertaken by Ray et al. [10], [22], [23]. In these papers, the authors proposed a *language measure*. This measure is based on the transition structure of an automaton accepting the language.

A. Relation to [8]

We mentioned that [8] uses a dynamic programming (DP) approach to solve an optimal discrete event control problem. We do not use a DP approach for two reasons:

- [8] assumes no a priori controller structure, only a finite state machine plant. Their technique builds a non-blocking finite state machine controller that minimizes cost. This is not the problem we study in this paper.
- A backwards DP analysis relies on a knowledge of the termination states. When M_K is a DPDM, the potential termination states are given by the final states *and* all possible stack configurations and hence form an infinite discrete (hence disconnected) set. The branch-and-bound method proposed here seems to be a more straight-forward approach to solving this problem. However, an open question that results from this research is whether DP can be applied to solve the optimal parametric control problem we've posed.

B. When the constant $\beta = 1$

In [8] Sengupta and LaFortune do not use a constant β. Instead, they assume that every closed cycle in the controller has net cost zero. If we have $\beta = 1$, then this is essential to ensure that the objective function is finite for all strings in K, the resulting controllable language. It is not clear whether this can be verified when $M_K(X_1, \ldots, X_n)$ is a DPDM and not a FSM. Hence, for $\beta = 1$, this problem may only be well defined and solvable when $M_K(X_1, \ldots, X_n)$ produces an FSM for all values of $X_1, \ldots, X_n$.

VI. Conclusion and Future Directions

In this paper we defined the optimal parametric control problem for discrete event control. We provided a branch-and-bound algorithm that solves this problem when the parametric control function $M_K(X_1, \ldots, X_n)$ has range properly contained in a set of DPDM or FSM and the depreciation constant $\beta < 1$. We showed that this algorithm was convergent.

We call this method parametric because the proposed problem is similar to a classical optimal parametric control problem, but rephrased for discrete event systems. Recall, classical parametrized control asks the question, "Assuming a PID controller is to be used what are the optimal tuning parameters to use in order to minimize a cost function J." In the case of statistical parametric control, the cost function may be the mean square error leading to MMSE parametric control or a cost function involving the gradient of the control function if large deviations in the controllable vector are undesirable [24]. In this paper, we have not discussed explicit methods for computing the objective function. In our work we found the simplest way was to approximate it using a secondary branch-and-bound problem. When M_K is given by a finite state machine, it may be possible to compute a functional description of the objective function explicitly using regular expressions for the language K. We are not certain whether there is a way to obtain a closed form expression when M_K is a DPDM. Additionally, our current computational experience shows that this algorithm is very slow. Methods for speeding up problem solving should be investigated, particularly when M_K is a DPDM.

References

[1] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206–230, 1987.

[2] ——, "On the supremal controllable sublanguage of a given language," *SIAM J. Control Optim.*, vol. 25, no. 3, pp. 637–659, 1987.

[3] R. D. Brandt, V. K. Garg, R. Kumar, F. Lin, S. I. Marcus, and W. M. Wonham, "Formulas for calculating supremal controllable and normal sublanguages," *Systems Control Lett.*, vol. 15, pp. 111–117, 1990.

[4] R. Kumar, V. K. Garg, and S. I. Marcus, "On controllability and normality of discrete event dynamic systems," *Systems Control Lett.*, vol. 17, no. 3, pp. 157–168, 1991.

[5] R. Sreenivas, "A note on deciding the controllability of a language k with respect to a language l," *IEEE Trans. Autom. Control*, vol. 38, no. 4, pp. 658–662, April 1993.

[6] B. A. Brandin and W. M. Wonham, "Supervisory control of timed discrete-event systems," *IEEE Trans. Autom. Control*, vol. 39, no. 2, pp. 329–342, Feb. 1994.

[7] K. C. Wong and W. M. Wonham, "Hierarchical control of discrete-event systems," *Discrete Event Dynamic Systems*, vol. 6, pp. 241–273, 1996.

[8] R. Sengupta and S. Lafortune, "An optimal control theory for discrete event systems," *SIAM J. Control Optim.*, vol. 36, no. 2, pp. 488–541, 1998.

[9] H. Marchand, O. Boivineau, and S. LaFortune, "On the synthesis of optimal schedulers in discrete event control problems with multiple goals," *SIAM J. Control Optim.*, vol. 39, no. 2, pp. 512–532, 2000.

[10] A. Ray and S. Phoha, "A language measure for discrete event automata," in *International Federation of Automatic Control World Congress*, Barcelona, Spain, 2002.

[11] Y. Cao and M. Ying, "Supervisory Control of Fuzzy Discrete Event Systems," *IEEE Trans. Autom. Control*, vol. 35, no. 2, pp. 366–371, 2005.

[12] C. Griffin, "A note on deciding controllability in pushdown systems," *IEEE Trans. Autom. Control*, vol. 51, no. 2, pp. 334–337, February 2006.

[13] A. E. C. da Cunha and J. E. R. Cury, "Hierarchical Supervisory Control Based on Discrete Event Systems with Flexible Marking," *IEEE Trans. Autom. Control*, vol. 32, no. 12, pp. 2242–2253, 2007.

[14] C. Griffin, "A note on the properties of the supremal controllable sublanguage in pushdown systems," *IEEE Trans. Autom. Control*, In Press, 2008.

[15] C. G. Cassandras and S. LaFortune, *Introduction to Discrete Event Systems*. Boston, MA, USA: Kluwer Academic Publishers, 1999.

[16] S. Simpson, "Mathematical logic," Available at: http://www.math.psu.edu/simpson/courses/math557/logic.pdf, 2000.

[17] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001.

[18] C. Griffin, "Decidability and Optimality in Pushdown Control Systems: A New Approach to Discrete Event Control," Ph.D. dissertation, Department of Industrial Engineering, The Pennsylvania State University, University Park, PA, December 2007.

[19] K. M. Passino and P. J. Antsaklis, "On the optimal control of discrete event systems," in *Proc. 28th IEEE Conf. on Decision and Control*, Tampa, FL, USA, 1989, pp. 2713–2718.

[20] R. Kumar and V. Garg, "Optimal supervisory control of discrete event dynamical systems," *SIAM J. Control Optim.*, vol. 33, pp. 419–439, 1995.

[21] Y. Brave and M. Heymann, "On optimal attraction of discrete-event processes," *Inform. Sci.*, vol. 67, pp. 245–276, 1993.

[22] X. Wang and A. Ray, "Signed real measure of regular languages," in *Proc. American Control Conference*, Anchorage AK, USA, 2002.

[23] J. Fu, A. Ray, and C. Lagoa, "Unconstrained optimal control of regular languages," in *Proc. 41st IEEE Conference on Decision and Control*, Las Vegas, NV, USA, December 2002, pp. 799–804.

[24] E. D. Castillo, *Statistical Process Adjustment for Quality Control*. New York, NY: Wiley-Interscience, 2002.