Discrete Event Dynamic Systems: Theory and Applications 2, (1992): 139–172

A Graph-Theoretic Optimal Control Problem for Terminating Discrete Event Processes*

RAJA SENGUPTA AND STÉPHANE LAFORTUNE
Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122

Received October 7, 1991; Revised May 11, 1992

Abstract. Most of the results to date in discrete event supervisory control assume a "zero-or-infinity" structure for the cost of controlling a discrete event system, in the sense that it costs nothing to disable controllable events while uncontrollable events cannot be disabled (i.e., their disablement entails infinite cost). In several applications however, a more refined structure of the control cost becomes necessary in order to quantify the tradeoffs between candidate supervisors. In this paper, we formulate and solve a new optimal control problem for a class of discrete event systems. We assume that the system can be modeled as a finite acylic directed graph, i.e., the system process has a finite set of event trajectories and thus is "terminating." The optimal control problem explicitly considers the cost of control in the objective function. In general terms, this problem involves a tradeoff between the cost of system evolution, which is quantified in terms of a path cost on the event trajectories generated by the system, and the cost of impacting on the external environment, which is quantified as a dynamic cost on control. We also seek a least restrictive solution. An algorithm based on dynamic programming is developed for the solution of this problem. This algorithm is based on a graph-theoretic formulation of the problem. The use of dynamic programming allows for the efficient construction of an "optimal subgraph" (i.e., optimal supervisor) of the given graph (i.e., discrete event system) with respect to the cost structure imposed. We show that this algorithm is of polynomial complexity in the number of vertices of the graph of the system.

Key Words: discrete event systems, optimal control, graph theory, dynamic programming

1. Introduction

1.1. Problem Description

This paper formulates and solves a new optimal control problem for logical discrete event systems. The uncontrolled discrete event system is modeled as a finite acylic directed graph. A set of paths (sequences of edges) is associated with this graph. This set is referred to as the language associated with the graph. This language represents the behavior of the discrete event system (DES); the system may execute any one of the paths of this language in a given time line. Since this language is finite, we say that the system process is "terminating."

Control is assumed to be the removal of edges directed outwards from a vertex. This is analogous to the disabling of events in the paradigm of the supervisory control theory for DESs initiated by Ramadge and Wonham [1987]. A control law is a function, defined at every vertex, which removes edges from the given graph. The controlled DES is thus

*Research supported in part by the National Science Foundation under grant ECS-9057967 with additional support from GE and DEC.

a subgraph of the uncontrolled graph and the language described by the controlled system is a sublanguage of the language of the uncontrolled DES.

We define two cost functions on the set of edges of the uncontrolled digraph. They are called the path cost function and the control cost function. Both functions map into the set of nonnegative real numbers. The path cost represents the cost incurred in traversing or executing an edge, whereas the control cost is the cost of removing or disabling the edge. We use these two costs to define the cost associated with a path in a controlled DES to be

(i) The sum of the path cost of each edge in the path
(ii) The sum of the control cost of each edge that lies in the uncontrolled DES but not in the controlled DES, and is attached to a vertex visited by the path

Thus the cost associated with one path lying in two different subgraphs or controlled DESs may be different.

It is now evident that there is a maximum cost path in each controlled DES or subgraph which represents the worst case for the control law defining the subgraph. Our aim is to construct the subgraph which minimizes this maximum cost. This is our concept of optimality.

1.2. Motivation

Various notions of optimality have already been examined in the context of logical DESs. In the Ramadge and Wonham [1987] framework the supremal controllable sublanguage of a given language has been considered optimal in the sense of being minimally restrictive within the given specifications of legality. The same spirit of optimality pervades the development of the infimal closed and controllable superlanguage and the examination of supervisory control with blocking by Chen and Lafortune [1990, 1991]. Accordingly in our investigation we have tried to remain consistent with this idea. We have examined the construction of a minimally restrictive optimal solution. This solution is supremal in a class of interesting optimal solutions, in the sense that every other optimal solution in this class is a subgraph of it.

In the area of cost-oriented optimal control of DESs, we first draw attention to the work of Passino and Antsaklis [1989] and Brave and Heymann [1990]. The former examines optimality with respect to a single event cost function which is used to associate a cost with every possible state trajectory. The objective is then to find an input sequence which forces a state trajectory that minimizes this cost. Our problem specializes to this when the control costs are set to zero. The report by Brave and Heymann [1990] on the optimal attraction in discrete event processes also analyzes a cost-oriented optimization problem. The DES is modeled by a finite state directed graph. The edges of the directed graph are assigned weights which are analogous to our path costs. This defines a maximum cost path in a controlled system. The objective is to build a supervisor that achieves minimal cost attraction with respect to an arbitrary initial state. Once again we differ in our formulation due to the additional control cost function.

In recent work, Kumar and Garg [1991] have also formulated a cost-oriented optimal control problem for DESs. Their definition of control cost is quite similar to ours. However

the control objective in our problem is quite different from theirs. Kumar and Garg attempt to reach a specified target behavior, and optimality is defined by proximity to this specified behavior. In our problem we have no specified target behavior. Instead we have a path cost function against which the control costs are traded off.

In a general sense, our problem formulation is motivated by the paradigm of classical optimal control theory. The system dynamics are represented by the finite digraph of the uncontrolled DES. The cost associated with a path in the uncontrolled DES is merely the sum of the path costs. There are no control costs in the uncontrolled system. We may attempt to exclude the maximum cost path in the uncontrolled DES but hitherto absent control costs are introduced. As we restrict behavior to smaller subsets of the superset of behavior, the control costs associated with the surviving paths are found to rise in general. On the other hand we can use more control to leave out the paths with higher path costs. Our formulation thus captures the fundamental tradeoff which has motivated the science of optimal control—the tradeoff between the cost of the system trajectory and the cost of the control necessary to produce the trajectory.

The rationale for the existence of a positive control cost lies in our interpretation of control. The control objective of the supervisor is to drive the system to a terminal state which represents some completed task or terminated process. To fulfill this objective in minimum time or space (depending on the interpretation of path cost), the supervisor disables certain events. We view this as impacting on the environment in which the system operates. The environment is inhibited by the action of the supervisor and consequently cost is incurred. For instance the disabling of events may obstruct other processes occurring in the environment of the system for which the supervisor is not responsible. The cost of control would then represent a penalty for such unknowing interference. We present an example in Section 7 to illustrate this interpretation.

It may be noted that our optimal solution is a sublanguage of a given uncontrolled language. In this sense we conform to the paradigm of supervisory control theory where the aim is to restrict system behavior to a subset of the set of all possible behaviors. Control does not, in general, force one particular behavior or one single path. Given the validity of this paradigm and the intuition of optimal control theory, we believe that the problem discussed in this paper is a natural and general formulation of a deterministic optimal control problem for logical DESs.

The reader will note that the forthcoming results assume no specification of illegal behavior in the uncontrolled system. We hope to extend our results to cover specifications of greater sophistication in the future.

1.3. Organization

Our presentation is organized as follows. Section 2 contains several definitions that are necessary for the graph-theoretic framework adopted in this paper. Given these definitions, the optimal control problem is precisely formulated in Section 3, and our principal results stated in Section 4. The solution algorithm together with an intuitive explanation is presented in Section 5. The proofs of our results, together with the necessary intermediate results, are grouped in Section 6. A simple example and a conclusion follow in Sections 7 and 8,

respectively. Some technical results, used in this paper, are collected in the appendix. A synopsis of the first four sections of this paper (excluding Section 4.3) appears in Sengupta and Lafortune [1991a]. Several proofs have also been omitted in this paper. The full exposition may be found in Sengupta and Lafortune [1991b].

2. Preliminary Definitions

2.1. The DES Model

As already mentioned in the introduction, the DES is modeled as a finite directed graph. Accordingly we define a digraph G to have the following structure: $G = (V, E)$ where V is the finite set of vertices of G and E is the finite set of directed edges of G. Moreover, each edge $e \in E$ may be expressed as a pair, $e = (v, v')$ where $\{v, v'\} \subseteq V$, and the edge is assumed to be directed from v to v'. If A be any digraph then for notational convenience, we define the functions $V_f(A)$ and $E_f(A)$ as returning the set of vertices and the set of edges of A respectively. A path lying in the digraph A is a sequence of directed edges $p = e_1e_2 \ldots e_{n-1}e_n$, $n \in \mathbb{N}$, where $\forall i$ s.t. $1 \leq i \leq n$, $e_i = (v_i, v_{i+1}) \in E_f(A)$. The vertex v_1 is called the start vertex. The vertex v_{n+1} is called the end vertex. The number n is the length of the path. We adopt the notation $\| p \| = n$ to denote this length and $p[v_a, v_b]$ to represent a path p with start vertex v_a and end vertex v_b. Observe that a path is also a digraph. Thus the functions $E_f(p)$ and $V_f(p)$ are defined. Accordingly a path p lies in a digraph G iff $E_f(p) \subseteq E_f(G)$ and $V_f(p) \subseteq V_f(G)$. We admit the operation of concatenation of paths in the usual way. It will be denoted by the symbol $\circ$.

The terms acyclic, accessibility and coaccessibility are frequently used in the subsequent development. A path is said to be acyclic if no two vertices on it are the same. An acyclic graph is one which has no acyclic paths. A vertex v is accessible in G w.r.t. v_o if there is a path in G from v_o to v. A vertex v is coaccessible in G w.r.t. v_m if there is a path in G from v to v_m.

Our investigations have thus far been confined to systems modeled by a restricted class of finite directed graphs. We refer to this class as the set of admissible graphs.

DEFINITION 2.1. An *admissible graph* G is a 4-tuple $G = \langle V_G, E_G, v_0, v_m \rangle$ having the following properties:

(i) $V_G = V_f(G)$ is finite.
(ii) $E_G = E_f(G)$ is finite.
(iii) G is acyclic.
(iv) There exists one and only one vertex of zero indegree denoted v_0.
(v) There exists one and only one vertex of zero outdegree denoted by v_m.
(vi) There is at most one edge between a pair of vertices.
(vii) Every vertex in G is accessible w.r.t. v_0.
(viii) Every vertex in G is coaccessible w.r.t. v_m.

It may be noted that henceforth the symbol G will always denote the admissible graph representing the uncontrolled DES.

An admissible path lying in an admissible graph G is one which starts at v_o and terminates at v_m. The set of admissible paths associated with the graph G is denoted by $L_m(G)$. The suffix-prefix closure of $L_m(G)$ is denoted by $L_{sp}(G)$ and is the set of all possible paths lying in G.

$L_m(G)$ represents the set of all possible acceptable behaviors of the uncontrolled system. It is the (marked) language generated by G. Each path in $L_m(G)$ represents a possible, appropriately terminated behavior of the system. Clearly if A_1 is a subgraph of A_2, then $L_m(A_1) \subseteq L_m(A_2)$.

By the finiteness of the admissible graph we can associate with each of its vertices a number which is the length of the longest path from the vertex to the vertex v_m. This is a useful way of ordering the vertices when examining them algorithmically.

DEFINITION 2.2. *Maximum path length from a vertex.*

Let G be the usual admissible graph. We define the function l_m: $V_f(G) \to \mathbb{N} \cup \{0\}$ by

$$l_m(v) = \max_{p[v,v_m] \in L_{sp}(G)} \|p\|.$$

2.2. *The Optimization Space*

We are interested in searching the space of subgraphs of G to find a subgraph which minimizes a cost function. The following definitions define this space.

DEFINITION 2.3. Let G be as usual and $v_d \in V_f(G)$. A graph A is an *admissible subgraph* of G rooted at v_d iff

(i) $V_f(A) \subseteq V_f(G)$ and $E_f(A) \subseteq E_f(G)$.
(ii) $A = \langle V_f(A), E_f(A), v_d, v_m \rangle$ is admissible.

We also define the set

$$\mathcal{S}(G, v_d) = \{A: A \text{ is an admissible subgraph of } G \text{ rooted at } v_d\}.$$

The results stated in Section 4 establish that it is possible to construct an optimal solution by a backward recursive dynamic programming algorithm. For this procedure, we require the one-step subgraph.

DEFINITION 2.4. Let $v_d \in V_f(G)$. A is a *one-step subgraph* of G rooted at v_d iff

(i) $v_d \in V_f(A)$.
(ii) $v \in V_f(A) \Rightarrow v = v_d$ or $\exists (v_d, v) \in E_f(A)$.
(iii) $V_f(A) \subseteq V_f(G)$ and $E_f(A) \subseteq E_f(G)$.
(iv) $(v, v') \in E_f(A) \Rightarrow v = v_d$.

We also define the set

$$\mathcal{S}_1(G, v_d) = \{A: A \text{ is a one-step subgraph of } G \text{ rooted at } v_d\}.$$

It may be noted that A is not an admissible graph.

Next a binary operation called Merge, denoted by $\oplus$, is defined in the space of graphs.

DEFINITION 2.5. *Merge.*

Let A, B be finite directed graphs. Then $A \oplus B$ is a finite directed graph such that

(i) $V_f(A \oplus B) = V_f(A) \cup V_f(B)$.
(ii) $E_f(A \oplus B) = E_f(A) \cup E_f(B)$.

This operation is used to construct larger graphs from smaller ones. The following two definitions illustrate the natural utility of this operation. The definitions are subsequently illustrated with examples.

DEFINITION 2.6. Define

$$S(G, v_d) = \bigoplus_{A \in \mathcal{S}(G, v_d)} A, \qquad v_d \in V_f(G).$$

Then $S(G, v_d)$ is the *maximal admissible subgraph* of G rooted at v_d. The set $\mathcal{S}(G, v_d)$ is finite. Thus $S(G, v_d)$ is well defined by inductively applying Theorem 4.1 on $\mathcal{S}(G, v_d)$ (see Section 4). The proof is straightforward and hence omitted.

DEFINITION 2.7. $S_1(G, v_d)$ is the *maximal one-step subgraph* of G rooted at v_d iff

(i) $S_1(G, v_d)$ is a one-step subgraph of G rooted at v_d.
(ii) $\forall A \in \mathcal{S}_1(G, v_d)$ we have $V_f(A) \subseteq V_f(S_1(G, v_d))$ and $E_f(A) \subseteq E_f(S_1(G, v_d))$.

EXAMPLE 2.1. Figures 1 to 5 illustrate the above definitions. The overall uncontrolled system G is given in Figure 1. The graph A in Figure 2 is a particular member of the set $\mathcal{S}(G, v_d)$ and the graph in Figure 3 represents $S(G, v_d)$, the merge of all such As. A_1 in Figure 4 is a particular member of the set $\mathcal{S}_1(G, v_0)$ while Figure 5 is the maximal one-step subgraph $S_1(G, v_0)$.

2.3. The Cost Structure

As mentioned in the introduction, two cost functions are defined on $E_f(G)$. They are

$$c_p: E_f(G) \to \mathbb{R}^+ \cup \{0\}$$

and

$$c_c: E_f(G) \to \mathbb{R}^+ \cup \{0\}.$$

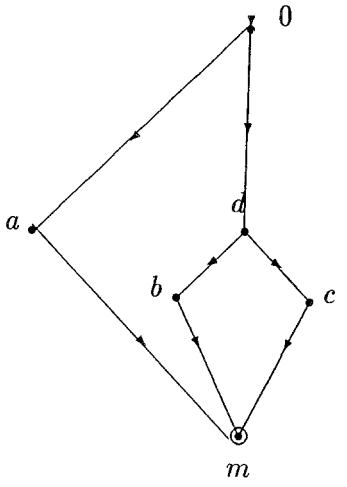


Figure 1. G of Examples 2.1 and 3.1.

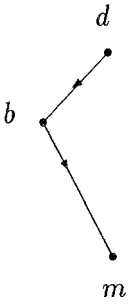


Figure 2. A of Example 2.1.

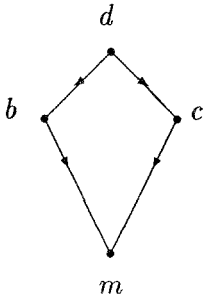


Figure 3. $S(G, v_d)$ of Example 2.1.

Figure 4. A_1 of Example 2.1.

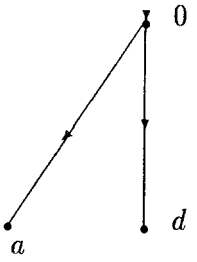


Figure 5. $S_1(G, V_0)$ of Example 2.1.

They denote the path and control costs respectively and map into the set of nonnegative real numbers.

DEFINITION 2.8. *Cost of a path lying in an admissible subgraph A of G.*

This cost is a function denoted by $c(\cdot, \cdot)$ and defined $\forall p[v_a, v_b] \in L_{sp}(A)$ by

$$c(p, A) = \sum_{e \in E_f(p)} c_p(e) + \sum_{e \in X} c_c(e),$$

where

$$X = \{(v, v'): (v, v') \in E_f(G) - E_f(A), v \in V_f(p) - \{v_b\}\}.$$

If $p \notin L_{sp}(A)$ then the function is undefined.

The set X in the definition above represents the set of edges lying in G but not in A, which are connected to vertices visited by the path p.

DEFINITION 2.9. *Maximum cost of a subgraph.*

This is a function defined on the set of all admissible subgraphs of G and denoted by $c_{\max}(\cdot)$. Let A be an admissible subgraph of G. Then

$$c_{\max}(A) = \max_{p_m \in L_m(A)} c(p_m, A).$$

The required definitions are now complete.

3. Problem Formulation

Let $G = \langle V_G, E_G, v_0, v_m \rangle$ represent the uncontrolled DES. Consider the set $\mathcal{S}(G, v_0)$ of all admissible subgraphs of G rooted at v_0. Pick any $A \in \mathcal{S}(G, v_0)$ and consider the function $c_{\max}(A)$ of Definition 2.9. The set $L_m(A)$ represents all possible behaviors or paths of the controlled system A. With each possible behavior, we have the associated cost $c(p_m, A)$. Observe that the control cost is dynamic since it is only incurred if the path visits a vertex where an edge in the set X has to be disabled. The maximization in $c_{\max}(A)$ thus represents the worst-case cost which may be incurred by the evolution of the system A. It is thus evident that for the purposes of a worst case analysis, $c_{\max}(\cdot)$ is the appropriate cost function or performance index. Consequently, our optimization problem is stated as follows:

We wish to find $A_0 \in \mathcal{S}(G, v_0)$ such that $c_{\max}(A_0) = \min_{A \in \mathcal{S}(G,v_0)} c_{\max}(A)$.

It may be noted that we search over the entire set $\mathcal{S}(G, v_0)$. The admissibility of G ensures that this set is finite. Moreover, we assume that each member of $\mathcal{S}(G, v_0)$ is constructible or there exists some control law which will yield the particular subgraph. Thus any edge can be removed or equivalently we have total controllability.

It may be noted that the optimal solution is not unique. It is also true that all solutions are not subgraphs of one another. Thus there exist incomparable optimal solutions. Moreover the dynamic programming (DP) principle is not universally applicable to all optimal solutions. In other words there exist optimal solutions which have no optimal substructure. The following example illustrates these observations.

EXAMPLE 3.1. Let the uncontrolled system be the graph G in Figure 1.

The costs are

Edge	c_p	c_c
0a	4	10
0d	1	20
db	1	2
dc	1	3
am	3	5
bm	1	5
cm	1	5

The largest optimal solution is obviously G itself with an associated worst-case cost of seven. The optimal solution rooted at v_d is depicted in Figure 6 and has an associated cost of two. However, the two subgraphs in Figures 7 and 8 are also optimal solutions at v_0.

These examples establish that the optimal solution is not unique. Note that both of the solutions in Figures 7 and 8 violate the DP principle at vertex v_d. Moreover neither solution is a subgraph of the other; i.e., there exist incomparable optimal solutions.

The maximization part of the min-max problem is performed over a space of paths but the minimization part is performed over a space of graphs. In this aspect of the problem

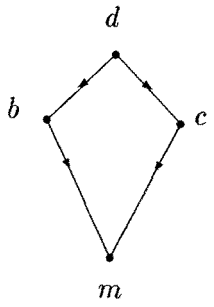


Figure 6. Unique optimal solution at v_d for Example 3.1.

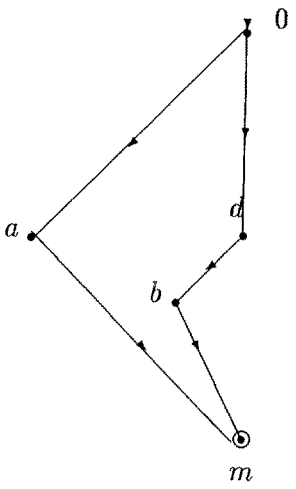


Figure 7. An optimal solution at v_0 for Example 3.1.

we find ourselves entirely in uncharted territory. In the following section we present a sequence of seven theorems which allow us to resolve these issues and extract a unique interesting minimally restrictive optimal solution to the aforementioned problem.

4. Statement of Principal Results

There are seven theorems stated in this section. Their basic purposes are as follows.

- Theorem 4.1 establishes the closure of the space of admissible graphs under the merge operation.
- Theorem 4.2 captures the tradeoff between the path and control costs of path.
- Theorem 4.3 is the dynamic programming (DP) equation for the max part of the min-max problem.

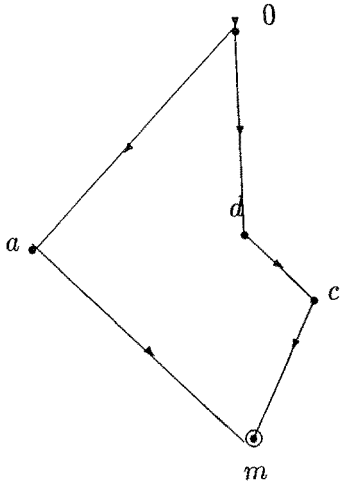


Figure 8. An optimal solution at v_0 for Example 3.1.

- Theorem 4.4 establishes the closure of the class of DP-optimal solutions under the merge operation.
- Theorem 4.5 is the DP construction for the min part of the min-max problem.
- Theorem 4.6 establishes that the complexity of the one-step minimization required in Theorem 4.5 is that of sorting at most $\| V \|$ numbers. Thus it is not necessary to search over the set of all subsets of the one-step subgraph rooted at a vertex, which may in the worst case be $2^{\| V \|}$.
- Theorem 4.7 proves that the computational complexity of the overall solution is polynomial.

4.1. The Merge Operation and the Cost Function

THEOREM 4.1. Let A, B be admissible subgraphs of an admissible graph G where $G = \langle V_G, E_G, v_0, v_m \rangle$, $A = \langle V_A, E_A, v_a, v_m \rangle$, and $B = \langle V_B, E_B, v_b, v_m \rangle$. Let $v_b \in V_f(A) = V_A$. Then $A \oplus B = \langle V_A \cup V_B, E_A \cup E_B, v_a, v_m \rangle$ is admissible.

This theorem is of fundamental importance in the incremental construction of solutions. The merger of admissible graphs is admissible subject to the simple condition: $v_a \in V_f(B) \vee v_b \in V_f(A)$. This restriction turns out to be quite trivial. The proof of this result follows from a straightforward and exhaustive examination of the definition of admissibility. It has been omitted in this paper.

THEOREM 4.2. Let $G = \langle V_G, E_G, v_0, v_m \rangle$ and $A_1, A_2 \in \mathcal{S}(G, v_d)$, $A_1 \in \mathcal{S}(A_2, v_d)$. Then for all $p_m \in L_m(A_1)$, $c(p_m, A_1) \geq c(p_m, A_2)$.

This theorem represents our contention that our cost structure captures the tradeoff between the cost of control and the path costs. More edges are removed in A_1 than A_2. Thus

for the surviving paths in A_1 control costs are higher than in A_2. On the other hand the maximum cost among the paths in A_1 is possibly lower than the maximum cost among the paths in A_2.

The following result establishes that the maximum cost associated with any admissible subgraph may be recursively computed from the costs associated with smaller admissible subgraphs attached to a one-step subgraph. If the reader looks ahead to the structure of the optimal solution, as indicated in Theorem 4.5, the importance of this result becomes obvious.

THEOREM 4.3. Let $A \in \mathcal{S}(G, v_d)$. Then

$$c_{\max}(A) = \max_{(v_d, v') \in E_f(S_1(A, v_d))} [c((v_d, v'), S_1(A, v_d)) + c_{\max}(S(A, v'))].$$

4.2. *The Dynamic Programming (DP) Principle for Subgraphs*

Recall that if A_0 is optimal in the set $\mathcal{S}(G, v_0)$ then $c_{\max}(A_0) = \min_{A \in \mathcal{S}(G, v_0)} c_{\max}(A)$. We define A_0 to be *DP-optimal* iff

(i) A_0 is optimal, and
(ii) $(\forall v \in V_f(A_0))\ c_{\max}(S(A_0, v)) = \min_{B \in \mathcal{S}(G, v)} c_{\max}(B)$.

We also adopt the notation $\mathcal{S}_D^o(G, v_d)$ for the set of all DP-optimal solutions of the graph G rooted at the vertex v_d.

The content of (ii) above is that every maximal subgraph of a DP-optimal solution is itself DP-optimal in its appropriate class or $\mathcal{S}(G, v)$ for some $v \in V_f(A_0)$. It is nontrivial to prove that the class of DP-optimal solutions is nonempty. This is the content of Corollary 4.1. It is satisfying that there exist optimal solutions whose subsolutions are also optimal. This is suggestive of a dynamic programming principle of sets or languages rather than of single paths. However the development of this idea requires the elucidation of the behavior of DP-optimal solutions under the merge operation. The following theorem together with Theorem 4.5 establishes the existence of a unique maximal DP-optimal solution.

THEOREM 4.4. Let $G = \langle V_G, E_G, v_0, v_m \rangle$ be an admissible graph. Let $S_a^o \in \mathcal{S}_D^o(G, v_a)$ and $S_b^o \in \mathcal{S}_D^o(G, v_b)$ where $v_b \in V_f(S_a^o)$. Then $S_a^o \oplus S_b^o \in \mathcal{S}_D^o(G, v_a)$.

Assuming that the set $\mathcal{S}_D^o(G, v_d)$ is nonempty, we obtain the interesting conclusion that the merger of DP-optimal solutions is DP-optimal. The admissibility of G ensures the finiteness of the set $\mathcal{S}_D^o(G, v_d)$. Thus by induction on Theorem 4.4 the merger of all elements of $\mathcal{S}_D^o(G, v_d)$ is its unique maximal element. Observe that Theorem 4.1 establishes the admissibility of $S_a^o \oplus S_b^o$.

Using the above theorem we are able to establish the following recursive construction of the DP-optimal solution. In the following, for any one-step subgraph S' rooted at a vertex v_d, we define the set of vertices $I(S') = \{v \in V_f(S') : v \neq v_d\}$.

THEOREM 4.5. Let $S'' = S' \oplus (\bigoplus_{v \in I(S')} S_v^o)$, $S' \in \mathcal{S}_1(G, v_d)$, $S_v^o \in \mathcal{S}_D^o(G, v)$ be such that

$$c_{\max}(S'') = \min_{A' \in \mathcal{S}_1(G, v_d)} c_{\max}\left[A' \oplus \left[\bigoplus_{v \in I(A')} S_v^o\right]\right],$$

where $S_v^o \in \mathcal{S}_D^o(G, v)$. Then $S'' \in \mathcal{S}_D^o(G, v_d)$.

COROLLARY 4.1. For all $v_d \in V_G$, $\mathcal{S}_D^o(G, v_d) \neq \emptyset$.

Theorem 4.5 states that if the set $\mathcal{S}_D^o(G, v)$ is nonempty for the descendents v of the vertex v_d, then the set $\mathcal{S}_D^o(G, v_d)$ is also nonempty. Consider a vertex v which is directly connected to v_m. The set $\mathcal{S}_D^o(G, v)$ is obviously nonempty for such a vertex. Proceeding intuitively from this case soon makes Corollary 4.1 apparent. The proof quite naturally is based on induction. It is omitted.

We next state another corollary of Theorem 4.5. This corollary establishes that the cost associated with S'' may be directly computed from its structural components as defined in the theorem.

COROLLARY 4.2. Let $S'' = S' \oplus (\bigoplus_{v \in I(S')} S_v^o) \in \mathcal{S}_D^o(G, v_d)$, where $S_v^o \in \mathcal{S}_D^o(G, v)$ for all $v \in I(S')$. Then

$$c_{\max}(S'') = \max_{(v_d, v) \in E_f(S')} [c((v_d, v), S') + c_{\max}(S_v^o)].$$

The proof of this corollary has been omitted since it is a straightforward application of Theorem 4.3 to Theorem 4.5.

The definition of S'' in Theorem 4.5 shows how to construct an element of $\mathcal{S}_D^o(G, v_d)$ given $S_v^o \in \mathcal{S}_D^o(G, v)$ where v is a direct descendent of v_d. This construction may be done by a backward DP-recursion. The theorem also shows that while the various S_v^o may be known, it is still necessary to find the minimizing S' in the set $\mathcal{S}_1(G, v_d)$. The structural properties of this set are elucidated in the following subsection.

To conclude this section we state the last and most important corollary of Theorem 4.5, on the existence of maximal DP-optimal solutions. Before stating the result it is necessary to introduce some notation. We define $M_D^o(G, v_d)$ to be the *maximal DP-optimal solution* of G rooted at v_d iff

(i) it is DP-optimal
(ii) $\forall A \in \mathcal{S}_D^o(G, v_d)$, $A \in \mathcal{S}(M_D^o(G, v_d), v_d)$.

We use the shorthand M_{v_d} for the symbol $M_D^o(G, v_d)$ in most places. The reader is requested to note that all subgraphs of maximal DP-optimal solutions are themselves maximal DP-optimal. We now state the corollary.

COROLLARY 4.3. For all $v_d \in V_G$, $M_D^o(G, v_d)$ exists and is unique. Moreover

$$M_D^o(G, v_d) = M' \oplus \left[\bigoplus_{v \in I(M')} M_D^o(G, v) \right],$$

where M' is the unique maximal A' satisfying the hypotheses of Theorem 4.5.

The proof of the existence part is obtained by inductively applying Corollary 4.1 and Theorem 4.4 on the merger of DP-optimal solutions. The proof of the constructive part is easily obtained from Theorems 4.4, 4.5, and the finiteness of the set $\mathcal{S}_1(G, v_d)$. The proof is straightforward and hence omitted.

The maximal DP-optimal solution may be viewed as the minimally restrictive optimal solution in this class of interesting solutions. Thus it possesses the optimality of Ramadge and Wonham's supremal controllable sublanguage and other work in the same paradigm.

It may be noted that conformity with the above DP principle is sufficient for optimality. This is evident from Theorem 4.5. But this conformity is not necessary. As seen in Example 3.1, it is easy to construct specific cases where a solution is optimal in the set $\mathcal{S}(G, v_0)$ but some subgraphs of the solution are not optimal in their respective classes.

4.3. Computation of One-Step Subgraphs for Optimal Solutions

The results of the prior subsection indicate that for all $v_d \in V_G$ it is possible to compute a DP-optimal solution by recursing backwards from v_m to v_d. At each step it is necessary to search through the set of all subsets of $\mathcal{S}_1(G, v_d)$. This set could have cardinality 2^n for an n vertex graph. This subsection establishes that if the descendent set of a vertex v be appropriately ordered then the maximal DP-optimal solution at the vertex v may be computed in as many steps as there are descendents. The sort algorithm is $O(n \log n)$ and it bounds the complexity of this stage.

Since subgraphs of maximal DP-optimal solutions are also maximal DP-optimal we state the following equivalent form of M_{v_d}:

$$M_{v_d} = M' \oplus \left[\bigoplus_{v \in I(M')} M_v \right],$$

where $M' = S_1(M_{v_d}, v_d)$ and M_v denotes the maximal DP-optimal solution in the class $\mathcal{S}(G, v)$. Next consider the set $E_f(S_1(G, v_d))$. We order this set by increasing cost as follows. If $E_f(S_1(G, v_d)) = \{e_1, \ldots, e_n\}$, $e_i = (v_d, v_i)$, then $\forall i, j$, $1 \leq i \leq j \leq n$, we have

$$c_p(e_i) + c_{\max}(M_{v_i}) \leq c_p(e_j) + c_{\max}(M_{v_j}).$$

Note that no control costs are to be considered. For all j such that $0 \leq j \leq n$ define $E_j = \{e_1, \ldots, e_{n-j}\}$. Thus $E_0 = E_f(S_1(G, v_d))$ and $E_n = \emptyset$. It is evident that E_j defines

a unique one-step subgraph in the class $\mathcal{S}_1(G, v_d)$. Let this graph be denoted by E_j'. Using this one-step subgraph we define a graph A_j for all j as follows:

$$A_j = E_j' \oplus \left[\bigoplus_{i=1}^{i=n-j} M_{v_i} \right].$$

It is obvious that $A_j \in \mathcal{S}(G, v_d)$ and $A_{j+1} \in \mathcal{S}(A_j, v_d)$. The principal result follows:

THEOREM 4.6. Let M_{v_d} be as defined above and let $\langle A_j \rangle_{j=0}^{j=n-1}$ be the associated sequence of graphs constructed as above. Let M' denote the maximal one-step subgraph of M_{v_d}. Then $\exists j$, $0 \leq j \leq n-1$, s.t. $M_{v_d} = A_j$.

It is evident from this theorem that if the set $E_f(S_1(G, v_d))$ be sorted as recommended then the number of candidates for M' is restricted to the cardinality of the set $E_f(S_1(G, v_d)) = E_0$. Thus the complexity of this step is linear in n.

On the basis of Theorems 4.5 and 4.6 we are able to prove the computational complexity of the algorithm stated in Section 6. Accordingly the following theorem concludes our investigation of the problem formulated in Section 3.

THEOREM 4.7. Let G be as usual and $\| V_G \| = n$. Then the computation of $M_D^o(G, v_o)$ is at most $O(n^2 \log n)$.

The proof of the above theorem is straightforward. Theorem 4.5 establishes that there are at most n steps in the backward DP-recursion. At each step of the recursion there are two computations, namely the sorting of the set $E_f(S_1(G, v_d))$ and cost computation for the sequence $\langle A_j \rangle_{j=0}^{j=n-1}$. The former is of order $O(n \log n)$ and the latter of order $O(n)$. Since the first grows faster than the second the total complexity is $O(n^2 \log n)$. A detailed proof is omitted.

5. Algorithm

We present in this section an algorithm to compute the maximal DP-optimal solution of an admissible graph G described as usual by $G = \langle V_G, E_G, v_o, v_m \rangle$. The application of this algorithm will be illustrated by a simple example in Section 7.

All new notation adopted in this section is defined below for easy reference.

(i) SL = solved list of vertices, i.e., optimal solution already computed.
(ii) UL = unsolved list of vertices, i.e., optimal solution to be computed.
(iii) $E_{\text{opt}}(v)$ = the set of edges of $S_1(M_D^o(G, v), v)$.
(iv) $CL = \{(v, c_{\max}(M_D^o(G, v))): v \in V_G\}$. This is the cost list maintained by the algorithm for the recursive computation of the cost associated with a particular subgraph.
(v) C = set of vertices to be optimized in the current iteration.
(vi) P = set of predecessor vertices from which C is to be chosen.

(vii) $P_f(C) = \{v \in V_G: (v, v') \in E_G, v' \in C\}$.
(viii) $S_f(v) = \{v' \in V_G: (v, v') \in E_G\}$. This is the set of successor vertices of the vertex v.

To enhance modularity we present the algorithm as a main program and two subprograms referred to as Optimize and One-Step Optimize respectively. The main program calls the program Optimize which in turn calls the subprogram One-Step Optimize.

The main program determines which vertices may be solved in the next iteration. These are contained in the set C. Thus a vertex is a member of C if and only if the maximal DP-optimal solutions are known for all its successor or descendent vertices. Since all paths terminate at v_m and the graph G is acyclic the set C is guaranteed never to be empty. The program Optimize iterates until all vertices in C are exhausted. It also orders the edges directed out of a vertex in C by increasing cost as required for the use of Theorem 4.6. The program One-Step Optimize simply examines the ordered set of edges starting with the last edge and determines which is the maximal DP-optimal subset. It uses the termination condition given by the two parts of Lemma 6.3 which is presented in the section on intermediate results (6.1).

5.1. The Main Program

(i) Input: V_G, E_G, v_o, v_m.
(ii) Initialize: $C = \{v_m\}$, $P = \emptyset$, $SL = \{v_m\}$, $UL = V_G - SL$, $CL = \emptyset$.
(iii) Optimize: Call subprogram Optimize with argument C.
(iv) Termination Condition: Is $v_o \in C$? If yes then STOP. Otherwise continue.
(v) Computation of the vertices to be optimized in the next iteration: First compute

$$P \leftarrow P \cup P_f(C) - C \text{ and then find } C = \{v \in P: S_f(v) \subseteq SL\}.$$

(vi) Update set of optimized vertices: $SL \leftarrow SL \cup C$.
(vii) Update remaining set of vertices: $UL \leftarrow UL - C$.
(viii) GOTO (iii).

5.2. Optimize

This program aside from iterating until C is empty essentially orders in step (v) the descendents of v_d. This ordering together with the construction of the A_j's referred to in Section 4.3 and the statement of Theorem 4.6, will make One-Step Optimize linear in $\| V \|$.

(i) Input: C.
(ii) Let $C_o = C$.
(iii) Pick any $v_d \in C_o$.
(iv) Update C_o: $C_o \leftarrow C_o - \{v_d\}$.

(v) Compute

$$E_0 = E_f(S_1(G, v_d)) = \{e_1, \ldots, e_n\},^1$$

$$V_0 = V_f(S_1(G, v_d)) - \{v_d\} = \{v_1, \ldots, v_n\},$$

where $e_i = (v_d, v_i)$ and order the e_i such that

$$i < j \Rightarrow c_p(e_i) + c_{\max}(M_D^o(G, v_i)) \leq c_p(e_j) + c_{\max}(M_D^o(G, v_j)).^2$$

(vi) If $E_0 = \emptyset$ then set $c_{\max}(E_0) = 0$.
Else set $c_{\max}(E_0) = c_p(e_n) + c_{\max}(M_D^o(G, v_n))$.

(vii) Call subprogram One-Step Optimize with arguments E_0, $c_{\max}(E_0)$, v_d.

(viii) Termination condition: Is $C_o = \emptyset$? If yes then return to the main program. Otherwise GOTO (iii).

5.3. *One-Step Optimize*

In this section we indulge in an abuse of notation. The symbol $c_{\max}(E')$ where $E' = \{e_1, \ldots, e_j\}$ is used to denote the following calculation:

$$c_{\max}(E') = c_p(e_j) + c_{\max}(M_D^o(G, v_j)) + \sum_{i=j+1}^{i=n} c_c(e_i),^3$$

where it is assumed that $E_0 = \{e_1, \ldots, e_n\}$ is ordered by increasing cost as described in (v) of Optimize. The notational abuse is justified since the set E', generated in each iteration of One-Step Optimize, is bijectively identifiable with the A_j's constructed for the statement of Theorem 4.6 in Section 4.3. Because of the DP principle the optimal A_j may be determined by a purely local or one-step view. Hence each step of the algorithm only looks at the one step subgraph rooted at v_d. All other required information has already been computed in prior iterations and stored in *CL* and the sets $E_{\text{opt}}(v)$ where v is a child of v_d.

(i) Input: E_0, $c_{\max}(E_0)$, v_d.

(ii) Initialize: $E = E_0$, $E' = E_0$, $CMAX = c_{\max}(E_0)$, $E_{\text{opt}}(v_d) = \emptyset$.

(iii) Compute[4] $E' \leftarrow E' - \{e_i: i = \max_{e_j \in E'} j\}$.
If $E' \neq \emptyset$ then set

$$c_{\max}(E') = c_p(e_{i-1}) + c_{\max}(M_D^o(G, v_{i-1})) + \sum_{k=i}^{k=n} c_c(e_k).$$

(iv) Termination condition: $E' = \emptyset$? If yes then set

$$E_{\text{opt}}(v_d) \leftarrow E_{\text{opt}}(v_d) \cup E \quad \text{and} \quad CL \leftarrow CL \cup \{(v_d, c_{\max}(E))\}$$

and return to Optimize.[2] Otherwise continue.

(v) Recursion condition:[4] Is $c_{\max}(E') < CMAX$? If not then GOTO (iii). If yes then continue.

(vi) Set $E = E'$, $CMAX = c_{\max}(E')$.

(vii) GOTO (iii).

It may be noted that in the above E represents the set of edges currently being minimized and $CMAX$ represents the current minimum cost. We point out that the above subprogram basically performs the minimization inherent in the following statement:

$$c_{\max}(M_{v_d}) = \min_{S' \in \mathcal{S}_1(G, v_d)} c_{\max}\left[S' \oplus \left[\bigoplus_{v \in I(S')} M_D^o(G, v) \right] \right].$$

The aim is to find the A_j that will satisfy Theorem 4.6.

6. Development of Principal Results

6.1. Intermediate Results

The following results will be used in Section 5.2 to prove the theorems of the previous section. The first lemma is essentially a mathematically convenient alternative form of the merge of two or more subgraphs.

LEMMA 6.1.

$$S(S_a \oplus S_b, v_d) = \left[\bigoplus_{v \in V_f(S(S_a \oplus S_b, v_d)) \cap V_f(S_a)} S(S_a, v) \right] \oplus \left[\bigoplus_{v \in V_f(S_b) \cap V_f(S(S_a \oplus S_b, v_d))} S(S_b, v) \right],$$

where $v_d \in V_f(S_a)$, $S_a \in \mathcal{S}(G, v_a)$, and $S_b \in \mathcal{S}(G, v_b)$.

Proof. Let

$$X_a = V_f(S(S_a \oplus S_b), v_d) \cap V_f(S_a),$$

$$X_b = V_f(S(S_a \oplus S_b), v_d) \cap V_f(S_b),$$

$$A = \left[\bigoplus_{v \in X_a} S(S_a, v) \right] \oplus \left[\bigoplus_{v \in X_b} S(S_b, v) \right].$$

(a) We first prove the following claim: $A \in \mathcal{S}(G, v_d)$.
Since $v_d \in V_f(S_a)$ we have

$$v_d \in X_a \Rightarrow v_d \in V_f(A).$$

To prove by contradiction we assume that there exists $v \in V_f(A)$ such that

$$p[v, v_d] \in L_{sp}(A) \subseteq L_{sp}(G).$$

Since

$$v \in V_f(A) = \left[\bigcup_{v' \in X_a} V_f(S(S_a, v')) \right] \cup \left[\bigcup_{v' \in X_b} V_f(S(S_b, v')) \right],$$

let it be assumed w.l.o.g. that $v \in V_f(S(S_a, v'))$ where $v' \in X_a$. Then there exists $p[v', v] \in L_{sp}(S(S_a, v')) \subseteq L_{sp}(G)$. But

$$v' \in X_a \Rightarrow v' \in V_f(S(S_a \oplus S_b, v_d))$$

$$\Rightarrow \exists\, p[v_d, v'] \in L_{sp}(S_a \oplus S_b) \subseteq L_{sp}(G).$$

Thus $p[v, v_d] \circ p[v_d, v'] \circ p[v', v] \in L_{sp}(G)$, which implies that G is cyclic. Since this contradicts the admissibility of G, $\exists p[v, v_d] \in L_{sp}(A)$ and v_d is of zero indegree.

Next pick any $v' \in V_f(A)$. Then $v' \in V_f(S(S_a, v))$ or $v' \in V_f(S(S_b, v))$ for some $v \in X_a$ or X_b, and $\exists p[v, v'] \in L_{sp}(S(S_a, v))$ or $L_{sp}(S(S_b, v)) \subset L_{sp}(S(S_a \oplus S_b, v_d))$. But

$$v \in X_a \text{ or } X_b \Rightarrow v \in V(S(S_a \oplus S_b, v_d)) \Rightarrow \exists p[v_d, v] \in L_{sp}(S(S_a \oplus S_b, v_d)).$$

Thus $p[v_d, v'] = p[v_d, v] \circ p[v, v'] \in L_{sp}(S(S_a \oplus S_b, v_d))$ and v_d is the one and only vertex of zero indegree. Since the other requirements of admissibility are trivial, we have

$$A = \langle V_A, E_A, v_d, v_m \rangle,$$

where

$$E_A = \left[\bigcup_{v \in X_a} E_f(S(S_a, v)) \right] \cup \left[\bigcup_{v \in X_b} E_f(S(S_b, v)) \right],$$

$$V_A = \left(\bigcup_{v \in X_a} V_f(S(S_a, v)) \right) \cup \left(\bigcup_{v \in X_b} V_f(S(S_b, v)) \right).$$

(b) Let $p_m = p[v_d, v_m] \in L_m(A)$. Then $E_f(p_m) \subseteq E_A \subseteq E_f(S_a) \cup E_f(S_b)$ and by Lemma A.1, $p_m \in L_m(S(S_a \oplus S_b, v_d))$. Therefore $L_m(A) \subseteq L_m(S(S_a \oplus S_b, v_d))$.

(c) Again let $p_m \in L_m(S(S_a \oplus S_b, v_d))$. Then $p_m = p[v_d, v_m]$ and $E_f(p_m) \subseteq E_f(S_a) \cup E_f(S_b)$. Now $p_m \in L_m(S(S_a \oplus S_b, v_d)) \Rightarrow p_m \in L_{sp}(S(S_a \oplus S_b, v_d))$ whence, $\forall e \in E_f(p_m)$, $e = (v, v'')$, we have

$$\begin{aligned} & e \in E_f(S(S_a \oplus S_b, v_d)) \subseteq E_f(S_a) \cup E_f(S_b) \\ & \Rightarrow v' \in V_f(S(S_a \oplus S_b, v_d)) \subseteq V_f(S_a) \cup V_f(S_b) \\ & \Rightarrow v' \in X_a \quad \text{or} \quad v' \in X_b \\ & \Rightarrow e \in E_f(S(S_a, v')) \quad \text{or} \quad e \in E_f(S(S_b, v')) \\ & \Rightarrow E_f(p_m) \subseteq E_f(A). \end{aligned}$$

Accordingly, by Lemma A.1, $p_m \in L_m(A)$ and thus $L_m(S(S_a \oplus S_b, v_d)) \subseteq L_m(A)$.

(d) From (b) and (c) we obtain $L_m(A) = L_m(S(S_a \oplus S_b, v_d))$. The result then follows by Lemma A.2. Q.E.D.

We next present a lemma which is essential to prove our main theorem on the recursive construction of DP-optimal solutions. This lemma is derived from Theorem 4.4. While the proof of Theorem 4.4 is presented in the following section it does not use this lemma.

LEMMA 6.2. Consider $A := S(S' \oplus (\oplus_{v_i \in I(S')} S^o_{v_i}), v_{i_1})$ where

(i) $S' \in \mathcal{S}_1(G, v_d)$.
(ii) $I(S') = \{v_1, \ldots, v_n\}$, where $v_i \in V_f(S')$, $v_i \neq v_d$, $1 \leq i \leq n$.
(iii) $\forall v_i \in I(S')$, $S^o_{v_i} \in \mathcal{S}^o_D(G, v_i)$.
(iv) $v_{i_1} \in I(S')$.

Then A is DP-optimal, i.e., $A \in \mathcal{S}^o_D(G, v_{i_1})$.

Proof. We define for each $v_i \in I(S')$

$$A_i = \langle \{v_d\} \cup V_f(S^o_{v_i}), \{(v_d, v_i)\} \cup E_f(S^o_{v_i}), v_d, v_m \rangle.$$

Then $A_i \in \mathcal{S}(G, v_d)$ and $S' \oplus (\oplus_{v_i \in I(S')} S^o_{v_i}) = \oplus_{1 \leq i \leq n} A_i \in \mathcal{S}(G, v_d)$ by Theorem 4.1. Consequently

$$A = S\left[\bigoplus_{1\le i\le n} A_i,\ v_{i_1}\right].$$

Consider A_i, corresponding to $v_i \in I(S')$. We wish to evaluate $S(A_{i_1} \oplus (\bigoplus_{\substack{1\le i\le n\\ i\ne i_1}} A_i),\ v_{i_1})$. By Lemma 6.1

$$S\left[A_{i_1} \oplus \left[\bigoplus_{\substack{1\le i\le n\\ i\ne i_1}} A_i\right],\ v_{i_1}\right] = \left[\bigoplus_{v\in X_1} S(A_{i_1},\ v)\right] \oplus \left[\bigoplus_{v\in X_2} S\left[\bigoplus_{\substack{1\le i\le n\\ i\ne i_1}} A_i,\ v\right]\right],$$

where

$$X_1 = V_f\left[S\left[A_{i_1} \oplus \left[\bigoplus_{\substack{1\le i\le n\\ i\ne i_1}} A_i\right],\ v_{i_1}\right]\right] \cap V_f(A_{i_1}),$$

$$X_2 = V_f\left[S\left[A_{i_1} \oplus \left[\bigoplus_{\substack{1\le i\le n\\ i\ne i_1}} A_i\right],\ v_{i_1}\right]\right] \cap V_f\left[\bigoplus_{\substack{1\le i\le n\\ i\ne i_1}} A_i\right].$$

We proceed by induction on n, which is the cardinality of $I(S')$. Let $n = 1$. Then

$$S\left[\bigoplus_{1\le i\le n} A_i,\ v_1\right] = A = S(A_1,\ v_1) = S^o_{v_1} \in \mathcal{S}^o_D(G,\ v_1),$$

which is the desired result. Next let $S(\bigoplus_{1\le i\le n} A_i,\ v_{i_1}) \in \mathcal{S}^o_D(G,\ v_{i_1})$. Consider

$$S\left[\bigoplus_{1\le i\le n+1} A_i,\ v_{i_1}\right] = A = S\left[A_{i_1} \oplus \left[\bigoplus_{\substack{1\le i\le n+1\\ i\ne i_1}} A_i\right],\ v_{i_1}\right].$$

Rearranging the indices we have

$$A = S\left[A_{i_1} \oplus \left[\bigoplus_{1\le j\le n} A_j\right],\ v_{i_1}\right].$$

Let

$$X = V_f\left[S\left[A_{i_1} \oplus \left[\bigoplus_{1\le j\le n} A_i\right]\right],\ v_{i_1}\right] \cap V_f(A_{i_1}),$$

$$X' = V_f\left[S\left[A_{i_1} \oplus \left[\bigoplus_{1\le j\le n} A_j\right]\right], v_{i_1}\right] \cap V_f\left[\bigoplus_{1\le j\le n} A_j\right].$$

By Lemma 6.1

$$A = \left[\bigoplus_{v\in X} S(A_{i_1}, v)\right] \oplus \left[\bigoplus_{v\in X'} S\left[\bigoplus_{1\le j\le n} A_j, v\right]\right].$$

Now $\forall v \in X$, we have $v \neq v_d$ and thus $S(A_{i_1}, v) = S(S^o_{v_{i_1}}, v) \in \mathcal{S}^o_D(G, v)$, since $S^o_{v_{i_1}} \in \mathcal{S}^o_D(G, v_{i_1})$. Also, $\forall v \in X'$ we have $v \neq v_d$. Thus

$$S\left[\bigoplus_{1\le j\le n} A_j, v\right] = S\left[S\left[\bigoplus_{1\le j\le n} A_j, v_k\right], v\right]$$

for some k such that $1 \le k \le n$.

But by our induction hypothesis

$$S\left[\bigoplus_{1\le j\le n} A_j, v_k\right] \in \mathcal{S}^o_D(G, v_k)$$

$$\Rightarrow S\left[\bigoplus_{1\le j\le n} A_j, v\right] \in \mathcal{S}^o_D(G, v) \qquad \text{where } v \in X'.$$

Therefore by Theorem 4.4 we have $A \in \mathcal{S}^o_D(G, v_{i_1})$. Q.E.D.

We present next the primary lemma needed for the proof of Theorem 4.6. Three lemmas referred to in the following proof may be found in the appendix. The notation A_j is as defined in Section 4.3.

LEMMA 6.3.

(i) If $c_{\max}(A_{j+1}) < \min_{0\le i\le j} c_{\max}(A_i)$ then $M_{v_d} = A_{j+1}$ or $M_{v_d} \in \mathcal{S}(A_{j+2}, v_d)$.
(ii) If $c_{\max}(A_{j+1}) \ge \min_{0\le i\le j} c_{\max}(A_i) = c_{\max}(A)$, where $A = A_k$ for the smallest k such that $0 \le k \le j$, then $M_{v_d} = A$ or $M_{v_d} \in \mathcal{S}(A_{j+2}, v_d)$.

Proof. Part (i). Let $M_{v_d} \neq A_{j+1}$ and $M_{v_d} \notin \mathcal{S}(A_{j+2}, v_d)$. By Lemma A.8, $M_{v_d} \in \mathcal{S}(A_{j+1}, v_d)$, which in conjunction with Lemma A.6 implies that $e_{n-j+1} \in E_f(M_{v_d})$. It is now immediate from Lemma A.7 that $M_{v_d} = A_{j+1}$. The result follows by contradiction.

Part (ii). We prove by induction.

BASE CASE. $j = 0$. Let $c_{\max}(A_1) \geq c_{\max}(A_0)$. We wish to show that $M_{v_d} = A_0$ or $M_{v_d} \in \mathcal{S}(A_2, v_d)$. Let it be assumed that $M_{v_d} \neq A_0$ and $M_{v_d} \notin \mathcal{S}(A_2, v_d)$. By Lemma A.6 we have that $e_n \in E_f(M_{v_d})$ or $e_{n-1} \in E_f(M_{v_d})$. If $e_n \in E_f(M_{v_d})$ then by Lemma A.7, $M_{v_d} = A_0$. Thus $e_n \notin E_f(M_{v_d})$. If $e_{n-1} \in E_f(M_{v_d})$ and $e_n \notin E_f(M_{v_d})$ then by Lemma A.7

$$M_{v_d} = A_1 \Rightarrow c_{\max}(M_{v_d}) = c_{\max}(A_1) \geq c_{\max}(A_0).$$

By the maximality of M_{v_d} we see that $M_{v_d} = A_0$. Consequently by contradiction the base case is established.

INDUCTION HYPOTHESIS. If $c_{\max}(A_j) \geq \min_{0 \leq r \leq j-1} c_{\max}(A_r) = c_{\max}(A')$, where $A' = A_k$ for the smallest k such that $0 \leq k \leq j - 1$, then $M_{v_d} = A'$ or $M_{v_d} \in \mathcal{S}(A_{j+1}, v_d)$.

PROPOSITION. If $c_{\max}(A_{j+1}) \geq \min_{0 \leq i \leq j} c_{\max}(A_i) = c_{\max}(A)$, where A is as usual, then $M_{v_d} = A$ or $M_{v_d} \in \mathcal{S}(A_{j+2}, v_d)$.

Case 1. Let A_j be such that $c_{\max}(A_j) \geq \min_{0 \leq i \leq j-1} c_{\max}(A_i) = c_{\max}(A')$. Then $\min_{0 \leq i \leq j} c_{\max}(A_i) = \min_{0 \leq i \leq j-1} c_{\max}(A_i) = c_{\max}(A') = c_{\max}(A)$. Obviously $A' = A$ and by our induction hypothesis $M_{v_d} = A$ or $M_{v_d} \in \mathcal{S}(A_{j+1}, v_d)$.

Case 2. Let A_j be such that $c_{\max}(A_j) \leq \min_{0 \leq i \leq j-1} c_{\max}(A_i)$. It is obvious that $A = A_j$ and by Lemma 6.3, $M_{v_d} = A_j = A$ or $M_{v_d} \in \mathcal{S}(A_{j+1}, v_d)$.

We now continue on both cases by contradiction. Let it be assumed that $M_{v_d} \neq A$ and $M_{v_d} \notin \mathcal{S}(A_{j+2}, v_d)$. Then by Lemma A.6, $e_{n-j+1} \in E_f(M_{v_d})$ and hence by Lemma A.7, $M_{v_d} = A_{j+1}$. Thus $c_{\max}(M_{v_d}) = c_{\max}(A_{j+1}) \geq c_{\max}(A)$. By the maximality of M_{v_d} we see that $M_{v_d} = A$. This is a contradiction and thus our proposition is established. Q.E.D.

6.2. *Proofs of Principal Results*

As noted earlier the proof of the conservation of admissibility under the merge operation stated in Theorem 4.1 has been omitted. For the sake of readability, the statement of the principal results in Section 4 are repeated before their proofs.

The following is the proof of the monotonicity of cost expressed in Theorem 4.2.

THEOREM 4.2. Let $G = \langle V_G, E_G, v_0, v_m \rangle$ and $A_1, A_2 \in \mathcal{S}(G, v_d)$, $A_1 \in \mathcal{S}(A_2, v_d)$. Then for all $p_m \in L_m(A)$, $c(p_m, A_1) \geq c(p_m, A_2)$.

Proof. Since $A_1 \in \mathcal{S}(A_2, v_d)$, $p_m \in L_m(A_1)$ implies $p_m \in L_m(A_2)$. Thus $c(p_m, A_2)$ is defined. In general for $p_m \in L_m(A_i)$,

$$c(p_m, A_i) = \sum_{e \in E_f(p_m)} c_p(e) + \sum_{e \in X_i} c_c(e),$$

where

$$X_i = \{(v, v'): (v, v') \in E_G - E_f(A_i) \text{ and } v \in V_f(p_m)\}.'$$

Since $A_1 \in \mathcal{S}(A_2, v_d)$ we have

$$E_f(A_1) \subseteq E_f(A_2)$$

$$\Rightarrow E_G - E_f(A_2) \subseteq E_G - E_f(A_1)$$

$$\Rightarrow X_2 \subseteq X_1$$

$$\Rightarrow \sum_{e \in X_2} c_c(e) \leq \sum_{e \in X_1} c_c(e), \qquad \text{since } c_c(e) \geq 0$$

$$\Rightarrow \sum_{e \in E_f(p_m)} c_p(e) + \sum_{e \in X_2} c_c(e) \leq \sum_{e \in E_f(p_m)} c_p(e) + \sum_{e \in X_1} c_c(e)$$

$$\Rightarrow c(p_m, A_2) \leq c(p_m, A_1). \text{ Q.E.D.}$$

The following is the proof of the recursive computation of the cost associated with a subgraph referred to in Theorem 4.3.

THEOREM 4.3. Let $A \in \mathcal{S}(G, v_d)$. Then

$$c_{\max}(A) = \max_{(v_d, v') \in E_f(S_1(A, v_d))} [c((v_d, v'), S_1(A, v_d)) + c_{\max}(S(A, v'))].$$

Proof. Since $A \in \mathcal{S}(G, v_d)$ we have

$$c_{\max}(A) = \max_{p_m \in L_m(A)} c(p_m, A)$$

$$= \max_{\substack{(v_d, v'), p'_m \\ (v_d, v') \circ p'_m \in L_m(A)}} [c((v_d, v'), A) + c(p'_m, A)]$$

$$= \max_{\substack{(v_d, v'), p'_m \\ (v_d, v') \circ p'_m \in L_m(A)}} [c((v_d, v'), S_1(A, v_d)) + c(p'_m, S(A, v'))]$$

by Lemmas A.4 and A.5. It is now clear that

$$c_{\max}(A) = \max_{(v_d, v') \in E_f(S_1(A, v_d))} [c((v_d, v'), S_1(A, v_d)) + c_{\max}(S(A, v'))]$$

if one observes that by Lemma A.3

$$(v_d, v') \in E_f(S_1(A, v_d)) \Rightarrow \exists p'_m \in L_m(S(A, v')) \quad \text{s.t. } (v_d, v') \circ p'_m \in L_m(A)$$

and

$$p_m \in L_m(A) \Rightarrow \exists (v_d, v') \in E_f(S_1(A, v_d)) \quad \text{s.t. } p_m = (v_d, v') \circ p'_m.$$

Q.E.D.

The following is the proof of Theorem 4.4 which may be used inductively to establish the existence of a maximal DP-optimal solution.

THEOREM 4.4. Let $G = \langle V_G, E_G, v_0, v_m \rangle$ be an admissible graph. Let $S_a^o \in \mathcal{S}_D^o(G, v_a)$ and $S_b^o \in \mathcal{S}_D^o(G, v_b)$ where $v_b \in V_f(S_a^o)$. Then $S_a^o \oplus S_b^o \in \mathcal{S}_D^o(G, v_a)$.

Proof. Since $v_b \in V_f(S_a^o)$, $S_a^o \oplus S_b^o \in \mathcal{S}(G, v_a)$ by Theorem 4.1. Thus admissibility is established. We prove the DP-optimality by induction.

Base case. Let $\ell_m(v_a) = 1$. Then $S_a^o = \langle \{v_a, v_m\}, \{(v_a, v_m)\}, v_a, v_m \rangle$. Since $v_b \in V_f(S_a^o)$ there are two possibilities for S_b^o.

(i) $\ell_m(v_b) = 0$. Then $S_b^o = \langle \{v_m\}, \emptyset, v_m, v_m \rangle$ and $S_a^o \oplus S_b^o = S_a^o \in \mathcal{S}_D^o(G, v_a)$.
(ii) $\ell_m(v_b) = 1$. Then $v_b = v_a$ and thus $S_a^o = S_b^o$. Again, $S_a^o \oplus S_b^o = S_a^o \in \mathcal{S}_D^o(G, v_a)$.

Thus the result is valid for $\ell_m(v_a) = 1$.

The induction step. Suppose that $\ell_m(v_a) = n$. We hypothesize as follows. Let $v_d \in V_f(G)$ where $\ell_m(v_d) \leq n - 1$. Furthermore let there exist $S_d^o \in \mathcal{S}_D^o(G, v_d)$ and $S_c^o \in \mathcal{S}_D^o(G, v_c)$ where $v_c \in V_f(S_d^o)$. Then $S_d^o \oplus S_c^o \in \mathcal{S}_D^o(G, v_d)$ by the induction hypothesis.

Now let $p_m \in L(S_a^o \oplus S_b^o)$. Then p_m may be partitioned as $p_m = p_1[v_a, v_d] \circ p_2[v_d, v_m]$ where

(i) $v_d \neq v_a$ and $v_d \in V_f(S_a^o) \cup V_f(S_b^o)$.
(ii) $p_1[v_a, v_d] \in L_{sp}(S_a^o)$ or $L_{sp}(S_b^o)$.
(iii) $p_2[v_d, v_m] \in L_m(S(S_a^o \oplus S_b^o, v_d))$.

Thus $l_m(v_d) < n$ since $l_m(v_a) = n$.

By Lemma 6.1

$$S(S_a^o \oplus S_b^o, v_d) = \left[\bigoplus_{v \in V_f(S(S_a^o \oplus S_b^o, v_d)) \cap V_f(S_a^o)} S(S_a^o, v) \right] \oplus \left[\bigoplus_{v \in V_f(S(S_a^o \oplus S_b^o, v_d)) \cap V_f(S_b^o)} S(S_b^o, v) \right].$$

Since $l_m(v) \leq l_m(v_d) < n$ and

$$S_a^o \in \mathcal{S}_D^o(G, v_a) \Rightarrow S(S_a^o, v) \in \mathcal{S}_D^o(G, v),$$

$$S_b^o \in \mathcal{S}_D^o(G, v_b) \Rightarrow S(S_b^o, v) \in \mathcal{S}_D^o(G, v),$$

we obtain from the induction hypothesis that $S(S_a^o \oplus S_b^o, v_d) \in \mathcal{S}_D^o(G, v_d)$. Thus it is established that every subgraph of $S_a^o \oplus S_b^o$ (except possibly itself) is DP-optimal.

It remains to establish the optimality of $S_a^o \oplus S_b^o$. Consider first the case where $p_1[v_a, v_d] \in L_{sp}(S_a^o)$. Then by Theorem 4.2 and Lemma A.4, we have that

$$\begin{aligned} c(p_m, S_a^o \oplus S_b^o) &= c(p_1[v_a, v_d], S_a^o \oplus S_b^o) + c(p_2[v_d, v_m], S_a^o \oplus S_b^o) \\ &\leq c(p_1[v_a, v_d], S_a^o) + c(p_2[v_d, v_m], S(S_a^o \oplus S_b^o, v_d)) \\ &\leq c(p_1[v_a, v_d], S_a^o) + c_{\max}(S(S_a^o \oplus S_b^o, v_d)). \end{aligned}$$

Since $S(S_a^o, v_d) \in \mathcal{S}_D^o(G, v_d)$ and $S(S_a^o \oplus S_b^o, v_d) \in \mathcal{S}_D^o(G, v_d)$, we have

$$c_{\max}(S(S_a^o, v_d)) = c_{\max}(S(S_a^o \oplus S_b^o, v_d)).$$

Accordingly,

$$c(p_m, S_a^o \oplus S_b^o) \leq c(p_1[v_a, v_d], S_a^o) + c_{\max}(S(S_a^o, v_d)) \leq c_{\max}(S_a^o).$$

If $p_1(v_a, v_d] \in L_{sp}(S_b^o)$ then $v_a = v_b$ since $v_b \in V_f(S_a^o)$. Thus by a similar argument

$$c(p_m, S_a^o \oplus S_b^o) \leq c(p_1[v_a, v_d], S_b^o) + c_{\max}(S(S_b^o, v_d)) \leq c_{\max}(S_b^o) = c_{\max}(S_a^o),$$

since $v_b = v_a$.

Thus $\forall p_m \in L_m(S_a^o \oplus S_b^o)$ we obtain $c(p_m, S_a^o \oplus S_b^o) \leq c_{\max}(S_a^o)$ which implies that $c_{\max}(S_a^o \oplus S_b^o) \leq c_{\max}(S_a^o)$. But by the optimality of S_a^o, $c_{\max}(S_a^o) \leq c_{\max}(S_a^o \oplus S_b^o)$ whence

$$c_{\max}(S_a^o \oplus S_b^o) = c_{\max}(S_a^o).$$

Thus $S_a^o \oplus S_b^o \in \mathcal{S}_D^o(G, v_a)$ and the proof of the induction step is completed. Q.E.D.

The next result is Theorem 4.5 which establishes the existence of a backward DP recursion method for the construction of DP-optimal solutions. Lemmas A.4 and A.5 referred to in this proof may be found in the appendix. Lemma 6.2 has already been presented in the prior subsection.

THEOREM 4.5. Let $S'' = S' \oplus (\bigoplus_{v \in I(S')} S_v^o)$, $S' \in \mathcal{S}_1(G, v_d)$, $S_v^o \in \mathcal{S}_D^o(G, v)$ be such that

$$c_{\max}(S'') = \min_{A' \in \mathcal{S}_1(G, v_d)} c_{\max}\left[A' \oplus \left[\bigoplus_{v \in I(A')} S_v^o\right]\right],$$

where $S_v^o \in \mathcal{S}_D^o(G, v)$. Then $S'' \in \mathcal{S}_D^o(G, v_d)$.

Proof. We first establish $S' \oplus (\bigoplus_{v \in I(S')} S_v^o) \in \mathcal{S}(G, v_d)$ for any $S' \in \mathcal{S}_1(G, v_d)$.

For each $v \in I(S')$ define $A_v = S_v^o \oplus \langle \{v_d, v\}, \{(v_d, v)\} \rangle$. Then $A_v \in \mathcal{S}(G, v_d)$ since S_v^o is admissible. Next note that

$$\bigoplus_{v \in I(S')} A_v = S' \oplus \left[\bigoplus_{v \in I(S')} S_v^o\right]$$

and by Theorem 4.1, $\bigoplus_{v \in I(S')} A_v \in \mathcal{S}(G, v_d)$. Thus admissibility is established.

By Lemma 6.2, for all $v' \in I(S')$, $S(S' \oplus (\bigoplus_{v \in I(S')} S_v^o), v')$ is DP-optimal for all $S' \in \mathcal{S}_1(G, v_d)$. For any $A \in \mathcal{S}(G, v_d)$ we have $S_1(A, v_d) \in \mathcal{S}_1(G, v_d)$ and hence $\forall v' \in V_f(S_1(A, v_d))$, $v' \neq v_d$,

$$c_{\max}(S(A, v')) \geq c_{\max}\left[S\left[S_1(A, v_d) \oplus \left[\bigoplus_{v \in I(S_1(A, v_d))} S_v^o\right], v'\right]\right].$$

Therefore by Theorem 4.3,

$$\begin{aligned} c_{\max}(A) &= \max_{(v_d, v') \in E_f(S_1(A, v_d))} [c((v_d, v'), S_1(A, v_d)) + c_{\max}(S(A, v'))] \\ &\geq \max_{(v_d, v') \in E_f(S_1(A, v_d))} \left[c((v_d, v'), S_1(A, v_d)) \right. \\ &\qquad \left. + c_{\max}\left[S\left[S_1(A, v_d) \oplus \left[\bigoplus_{v \in I(S_1(A, v_d))} S_v^o\right], v'\right]\right]\right] \\ &\geq \min_{S' \in \mathcal{S}_1(G, v_d)} \left[\max_{(v_d, v') \in E_f(S')} \left[c((v_d, v'), S') \right.\right. \\ &\qquad \left.\left. + c_{\max}\left[S\left[S' \oplus \left[\bigoplus_{v \in I(S')} S_v^o\right], v'\right]\right]\right]\right] \end{aligned}$$

$$= \min_{S' \in \mathcal{S}_1(G, v_d)} c_{\max}\left[S' \oplus \left[\bigoplus_{v \in I(S')} S_v^o\right]\right] = c_{\max}(S''),$$

where the second inequality follows from $S_1(A, v_d) \in \mathcal{S}_1(G, v_d)$ and the equality once again from Theorem 4.3.

Thus $\forall A \in \mathcal{S}(G, v_d)$, $c_{\max}(A) \geq c_{\max}(S'')$ and we have proved that S'' is optimal. It remains to show that S'' is DP-optimal.

Pick any $v'' \in V_f(S'')$, $v'' \neq v_d$, and $S'' = S' \oplus (\bigoplus_{v \in I(S')} S_v^o)$, where S' is the minimizing one among the elements of $\mathcal{S}_1(G, v_d)$. If $v'' \in I(S')$ then the DP-optimality has already been established by Lemma 6.2. If $v'' \notin I(S')$ then

$$S\left[S' \oplus \left[\bigoplus_{v \in I(S')} S_v^o\right], v''\right] = S\left[S\left[S' \oplus \left[\bigoplus_{v \in I(S')} S_v^o\right], v'\right], v''\right]$$

for some $v' \in I(S')$. But the right-hand side subgraph is DP-optimal since, by Lemma 6.2 again,

$$S\left[S' \oplus \left[\bigoplus_{v \in I(S')} S_v^o\right], v'\right] \in \mathcal{S}_D^o(G, v'), \qquad \forall v' \in I(S').$$

Thus $S'' \in \mathcal{S}_D^o(G, v_d)$. Q.E.D.

Finally, we prove the result on the construction of maximal one-step subgraphs. All notation in the following proof is as defined in Section 4.3.

THEOREM 4.6. Let M_{v_d} be as defined above and let $\langle A_j \rangle_{j=0}^{j=n-1}$ be the associated sequence of graphs constructed as above. Let M' denote the maximal one-step subgraph of M_{v_d}. Then $\exists j$, $0 \leq j \leq n-1$, s.t. $M_{v_d} = A_j$.

Proof. For $v_d \in V_G$ consider the set $E_f(S_1(G, v_d)) = E_0$ and the sequence of graphs $\langle A_j \rangle_{j=0}^{j=n-1}$. Let E_0 be ordered by increasing cost. It is obvious that for all j either $c_{\max}(A_{j+1}) < \min_{0 \leq i \leq j} c_{\max}(A_i)$ or $c_{\max}(A_{j+1}) \geq \min_{0 \leq i \leq j} c_{\max}(A_i)$. By the two parts of Lemma 6.3 in both cases M_{v_d} is one of the elements in the sequence $\langle A_i \rangle_{i=0}^{i=j+1}$ or it is a subgraph of A_{j+2}. Since M_{v_d} exists and $A_n = \emptyset$ it is evident that $M_{v_d} = A_j$ for some j.

7. Example

We present here a simple example which illustrates a situation in which our two-cost formulation is meaningful and an optimal solution is constructed by backward DP recursion.

Consider Figure 9 and assume that it represents the floorplan of a house with a robot in it that delivers things from the lobby to the backroom. It is assumed that there are plenty of people moving around this house who may obstruct the robot. The robot can protect

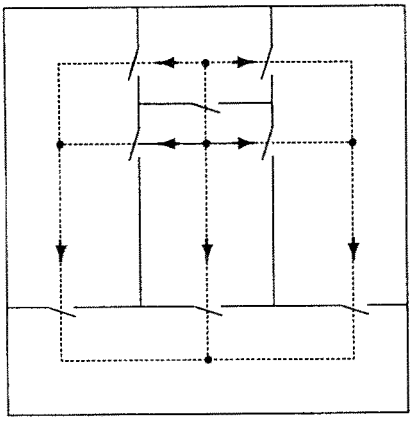


Figure 9. Floorplan of house of Section 7.

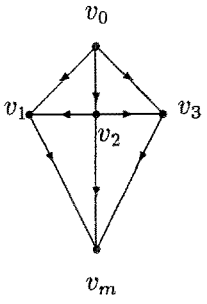


Figure 10. G of Section 7.

its path by closing various doors in the house. However, such control action would obstruct other people in the house and consequently cost is incurred. This is the control cost.

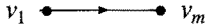
The system may be modeled by the following finite digraph $G = \langle V, E, v_0, v_m \rangle$ where $V = \{v_0, v_1, v_2, v_3, v_m\}$ and $E = \{e_{01}, e_{02}, e_{03}, e_{23}, e_{21}, e_{1m}, e_{2m}, e_{3m}\}$. G is depicted in Figure 10.

The cost functions are defined as follows:

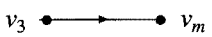
Edge	c_p	c_c
e_{01}	2	2
e_{02}	2	2
e_{03}	2	2
e_{21}	3	1
e_{23}	3	1
e_{1m}	4	0
e_{2m}	4	0
e_{2m}	4	0

The first call to the program Optimize of the algorithm in Section 5 is with $C = \{v_m\}$, which immediately returns since v_m has no successors. At the next step, the main program puts vertices v_1 and v_3 into the set C.

By admissibility, $M_D^o(G, v_1)$ is



with cost $c_{\max}(M_D^o(G, v_1)) = 4$, and $M_D^o(G, v_3)$ is



with cost $c_{\max}(M_D^o(G, v_3)) = 4$ also. At the next step, $C = \{v_2\}$ and it is found that $M_D^o(G, v_2)$ is

with cost $c_{\max}(M_D^o(G, v_2)) = c_p(e_{2m}) + c_c(e_{21}) + c_c(e_{23}) = 6$. To see this, we apply Theorem 4.6 and observe that we need only compare the above subgraph with the subgraphs A_1 and A_2 in Figure 11, since a valid ordering of the set $E_f(S(G, v_2))$ is $\{e_{2m}, e_{21}, e_{23}\}$ (cf. Section 4.3). This ordering would be produced algorithmically by step (v) of Optimize and One-Step Optimize would identify $E_{\text{opt}}(v_2) = \{v_m\}$ as the optimal subgraph of G rooted at v_2. Indeed, $c_{\max}(A_1) = c_p(e_{21}) + c_p(e_{1m}) = 7$ and $c_{\max}(A_2) = c_p(e_{21}) + c_p(e_{1m}) + c_c(e_{23}) = 8$, which proves our contention.

Finally, we proceed similarly to analyze vertex v_0 in the last iteration of the main program. An appropriate ordering of the set $E_f(S_1(G, v_0))$ is $\{e_{01}, e_{03}, e_{02}\}$ and thus we need only compare the subgraphs A_3, A_4, and A_5 depicted in Figure 12.

Using Corollary 4.2, we obtain

$$c_{\max}(A_3) = \max_{i\in\{1,2,3\}} [c_p(e_{oi}) + c_{\max}(M_D^o(G, v_i))]$$
$$= \max[2 + 4, 2 + 6, 2 + 4] = 8$$

$$c_{\max}(A_4) = \max_{i\in\{1,3\}} [c_p(e_{oi}) + c_c(e_{02}) + c_{\max}(M_D^o(G, v_i))]$$
$$= \max[2 + 2 + 4, 2 + 2 + 4] = 8$$

$$c_{\max}(A_5) = c_p(e_{01}) + c_c(e_{02}) + c_c(e_{03}) + c_{\max}(M_D^o(G, v_1)) = 10$$

which shows that $M_D^o(G, v_0) = A_3$.

8. Conclusion

We have formulated and solved a new optimal control problem for DESs in a graph-theoretic framework. To the best of our knowledge, our development of a dynamic programming recursion on subgraphs (Theorem 4.5, Corollary 4.2, and other related results), together

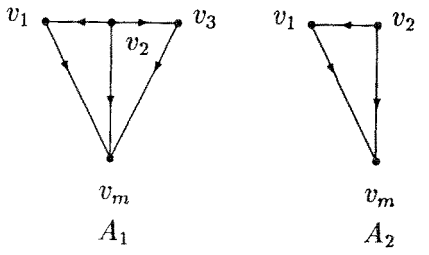


Figure 11. A_1 and A_2 of Section 7.

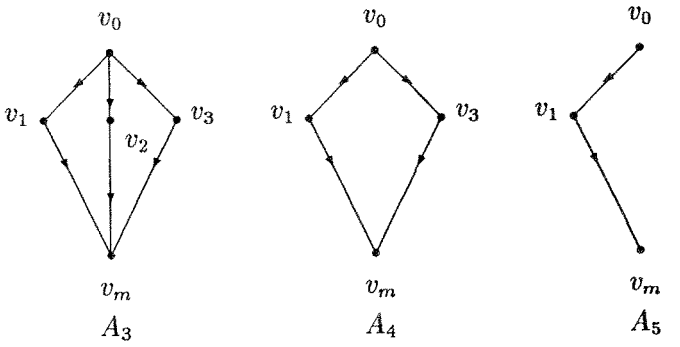


Figure 12. A_3, A_4, and A_5 of Section 7.

with the consideration of a cost structure comprising "path" and "control" costs, are novel contributions to the optimal control theory of DESs. This approach is also suitable for several generalizations of the problem considered in this paper. We are currently engaged in extending this work by relaxing the assumptions defining an admissible graph, thereby enabling our solutions to encompass a wider variety of DESs. Most of the admissibility restrictions are fairly trivially generalized. If accessibility and coaccessibility are violated then we would simply extract the maximal accessible and coaccessible subgraph of G. This computation would be $O(n)$. The single initial vertex restriction can easily be generalized to one of multiple initial vertices without affecting the complexity. The incorporation of uncontrollable events would only entail minor modifications to the algorithm. However we would have to state existence conditions. These would be almost similar to those already in the literature, e.g., Brave and Heymann [1990] and Ramadge and Wonham [1987]. The hard restrictions are the absence of cycles and the assumption of a single terminal vertex. These restrictions are currently under investigation. We believe that the cyclic problem would only be solvable with a higher order of complexity, which may however be better than exponential. We are however unable to make any formal statements on the relaxation of these two restrictions at this point.

The advantage of using an approach based on a graph model is evidenced by the complexity result in Theorem 4.7. In the language-theoretic approach, finding an optimal solution would require examining all the sublanguages of the language $L_m(G)$ corresponding

to the uncontrolled DES. This is of exponential complexity in the cardinality of the set $L_m(G)$. In our graph-theoretic approach, we are able to reduce the complexity to $O(n^2 \log n)$, where n could be as low as the number of Nerode equivalence classes of $L_m(G)$. In general the optimization space is not as large in this case, since not all sublanguages of $L_m(G)$ need be subgraphs of G. But if the graph structure G is rich enough (as is the case in the example in Section 7), then there is no reduction of the optimization space.

Appendix

The proofs of the following results are straightforward from the definitions in Section 2 and omitted here.

LEMMA A.1. Let $G = \langle V_G, E_G, v_0, v_m \rangle$ be an admissible graph and $v_d \in V_f(G)$. Then $p[v_a, v_b] \in L_{sp}(S(G, v_d))$ iff

(i) $E_f(p[v_a, v_b]) \subseteq E_f(G)$.
(ii) $\exists p[v_d, v_a]$ such that $E_f(p[v_d, v_a]) \subseteq E_f(G)$.

In particular, $L_m(S(G, v_d)) = \{p[v_d, v_m]: E_f(p[v_d, v_m]) \subseteq E_f(G)\}$. (It may be noted that (ii) is redundant if $v_d = v_0$, in which case $S(G, v_d) = G$).

LEMMA A.2. Let A and B be admissible subgraphs of an admissible graph G where $A = \langle V_A, E_A, v_a, v_m^a \rangle$ and $B = \langle V_B, E_B, v_b, v_m^b \rangle$. Then $A = B$ iff $L_m(A) = L_m(B)$.

LEMMA A.3. Let $G = \langle V_G, E_G, v_0, v_m \rangle$ be as usual and $v_d \in V_G$. Then $S_1(G, v_d)$ is defined by

(i) $E_f(S_1(G, v_d)) = \{(v_d, v): (v_d, v) \in E_f(G)\}$.
(ii) $V_f(S_1(G, v_d)) = \{v : (v_d, v) \in E_f(G)\} \cup \{v_d\}$.

The next two lemmas are required for the proof of Theorem 4.5. They are stated without proof since they are intuitively clear.

LEMMA A.4. Let $A \in \mathcal{S}(G, v_d)$. Then $c((v_d, v'), A) = c((v_d, v'), S_1(A, v_d))$.

LEMMA A.5. Let $p[v_a, v_b] \in L_{sp}(D)$, where $D \in \mathcal{S}(G, v_d)$. Then $c(p[v_a, v_b], D) = c(p[v_a, v_b], S(D, v_a))$.

We conclude this appendix with three lemmas that are required for the proofs of Lemma 6.3 and Theorem 4.6. All special notation used in this lemmas is as defined in Section 4.3.

LEMMA A.6. If $M_{v_d} \notin \mathcal{S}(A_j, v_d)$ then $\exists i$ such that $e_i \in E_f(M_{v_d}) \subseteq E_0$ and $n - j < i \leq n$.

Proof. We have $M_{v_d} = M' \oplus (\bigoplus_{v \in I(M')} M_v)$ where $E_f(M') \subseteq \{e_1, \ldots, e_n\}$. Let it be assumed that for all i s.t., $e_i \in E_f(M_{v_d})$ we have $1 \leq i \leq n - j$. Then $e_i \in E_j \subseteq E_f(A_j)$ since $A_j = E_j' \oplus (\bigoplus_{i=1}^{i=n-j} M_{v_i})$. Thus $E_f(M') \subseteq E_f(A_j)$ and for each $v \in I(M')$ there exists v_i with $1 \leq i \leq n - j$, whence by the uniqueness of the maximal DP-optimal solution we see that $M_v = M_{v_i}$ if $v = v_i$. Thus $E_f(M_v) \subseteq E_f(A_j)$ where from $E_f(M_{v_d}) \subseteq E_f(A_j)$. It is not immediate that $M_{v_d} \in \mathcal{S}(A_j, v_d)$. Thus the contrapositive is true and the result follows. Q.E.D.

LEMMA A.7. Let $e_{n-j} \in E_f(M_{v_d})$ and $\forall i$ s.t. $n - j < i \leq n$, $e_i \notin E_f(M_{v_d})$. Then $M = A_j$.

Proof. Since $e_{n-j} \in E_f(M_{v_d})$ we have that

$$c_{\max}(M_{v_d}) \geq c_p(e_{n-j}) + c_{\max}(M_{v_{n-j}}) + \sum_{i=n-j+1}^{i=n} c_c(e_i)$$

$$= c_{\max}(A_j),$$

the latter equality being obtained from the definition of A_j and the ordering of the set $E_f(S_1(G, v_d)) = E_0$. By the optimality of M_{v_d} we have that $c_{\max}(M_{v_d}) = c_{\max}(A_j)$. By the maximality of M_{v_d}, A_j is a subgraph of M_{v_d}. By the contrapositive of Lemma A.6, M_{v_d} is a subgraph of A_j. Thus $M_{v_d} = A_j$. Q.E.D.

LEMMA A.8. If $c_{\max}(A_{j+1}) < \min_{0 \leq i \leq j} c_{\max}(A_i)$, then $M_{v_d} \in \mathcal{S}(A_{j+1}, v_d)$.

Proof. Let $c_{\max}(A_{j+1}) < \min_{0 \leq i \leq j} c_{\max}(A_i)$ and $M_{v_d} \notin \mathcal{S}(A_{j+1}, v_d)$. Then by Lemma A.6, there exists $e_i \in E_f(M_{v_d})$ such that $n - (j + 1) < i \leq n$. Let the largest such i be denoted by i_p. Then for all i such that $i_p < i \leq n$, $e_i \notin E_f(M_{v_d})$. By Lemma A.7 we obtain that $M_{v_d} = A_{n-i_p}$.

Now $n - j \leq i_p \leq n \Rightarrow j - n \geq -i_p \geq -n \Rightarrow j \geq n - i_p \geq 0$. Since $0 \leq n - i_p \leq j$ we have

$$c_{\max}(M_{v_d}) = c_{\max}(A_{n-i_p}) \geq \min_{0 \leq i \leq j} c_{\max}(A_i) > c_{\max}(A_{j+1}),$$

which is absurd. Thus $M_{v_d} \in \mathcal{S}(A_{j+1}, v_d)$. Q.E.D.

Acknowledgment

The authors wish to acknowledge useful discussions with Professor Kevin Compton (EECS Department, University of Michigan). We are also grateful to the reviewers for their careful and insightful suggestions on improving the presentation of this paper.

Notes

1. By Collorary 4.3, which is immediate from Theorems 4.4 and 4.5, the maximal DP-optimal solution can be constructed from a subset of E_0.
2. $c_{\max}(M_D^o\ (G,v_j))$ is obtained from *CL* which is indexed by v_j (see One-Step Optimize, step (iv)).
3. Since the set E_0 is ordered as in (v) of Optimize this equation is obtained by Theorem 4.3.
4. The set E' is being searched in order of decreasing cost. By Theorem 4.6 this is sufficient to find the maximal DP-optimal solution. The two cases of the recursion condition are the two parts of Lemma 6.3 which appears in the next section.

References

Brave, Y., and Heymann, M., 1990. On optimal attraction of discrete-event proceses. Center for Intelligent Systems Report 9010, Technion, Haifa, Israel.

Chen, E., and Lafortune, S., 1991. Dealing with blocking in supervisory control of discrete event systems. *IEEE Trans. Automatic Control*, 36(6):724–735.

Kumar, R., and Garg, V., 1991. Optimal control of discrete event dynamical systems using network flow techniques. Preprint, Department of Electrical and Computer Engineering, University of Texas, Austin.

Lafortune, S., and Chen, E., 1990. The infimal closed controllable superlanguage and its application in supervisory control. *IEEE Trans. Automatic Control*, 35(4):398–405.

Passino, K.M., and Antsaklis, P.J., 1989. On the optimal control of discrete event systems. *Proc. 28th IEEE Conf. Decision Control*: 2713–2718.

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

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

Sengupta, R., and Lafortune, S., 1991a. Optimal control of a class of discrete event systems. *Preprints, IFAC Int. Symp. on Distributed Intelligence Systems*: 25–30.

Sengupta, R., and Lafortune, S., 1991b. An optimal control problem for a class of discrete event systems. Technical Report CGR-57, College of Engineering Control Group Reports, University of Michigan.