Proc. of the Allerton Conf. , Univ. of Illinois at Champaign-Urbana, Sept. 1989.

NEAR-OPTIMAL CONTROL OF DISCRETE EVENT SYSTEMS

K.M. PASSINO and P.J. ANTSAKLIS
Dept. of Electrical and Computer Engineering
University of Notre Dame, Notre Dame IN 46556

Abstract

The problem of the near-optimal control of systems accurately represented with a logical discrete event system (DES) model is formulated and solved for the deterministic case in this paper. The given DES model P characterizes the *valid* dynamical behavior of the plant and another DES model A represents design objectives which specify the *allowable DES behavior*. Under certain conditions a controller may be constructed which will select a sequence of inputs that results in allowable plant behavior. Here we consider the case where there is another part of the design objectives which indicates that the plant behavior should also, in some sense, be "near-optimal". It is within this context that we formulate a "near-optimal controller synthesis problem", i.e. how to construct a controller to achieve near-optimal allowable DES behavior. Our solution to this problem utilizes results from the theory of heuristic search to help overcome problems with computational complexity often encountered with logical DES models. This approach relies on the choice of an "evaluation function" to represent "heuristics" which are used to focus the search for an near-optimal solution. It is shown that when these heuristics over-estimate (e.g. if the heuristics are "filtered"), near-optimal solutions can still be obtained. Next, the notion of "ε-monotonicity" is introduced and used to help quantify the performance of the algorithm used to find near-optimal solutions. We also address the fact that it is, in general, quite difficult to find an appropriate heuristic function (even if it can over-estimate) for particular applications. We show that a metric space approach can be used to specify over-estimating heuristic functions in a systematic way for a wide variety of DESs. Examples are provided.

1. INTRODUCTION

This paper establishes the first steps towards developing the foundations for an near-optimal control theory for systems accurately represented with a logical discrete event system (DES) model. Here, we focus on one class of deterministic logical DES models (an automaton augmented so that it can model costs for events to occur) and the case where complete state information is available. (These results are based on those established in [7,8].) We utilize the DES controller synthesis methodology introduced in [7]. The technique requires viewing the plant model P as characterizing the *valid DES behavior* while another DES model A represents design objectives which specify the *allowable* (desirable) *DES behavior* which is "contained in" the valid behavior. Here, we are concerned with those times when the design objectives dictate that not only should the DES behavior be allowable but that it is in some sense near-optimal. To quantify this, a performance index is defined in terms of the costs for the events to occur. The near-optimal controller synthesis problem for deterministic DES (defined in Section 2) consists of the construction of a controller that chooses a sequence of inputs that will transfer the DES from its initial state to one state in a set of final states and produce a near-optimal state trajectory.

Our solution to this near-optimal control problem utilizes approaches developed in the theory of heuristic search to help overcome problems with computational complexity often encountered in the study of logical DESs. The idea is to *search* for a near-optimal solution among all the possible solutions to the near-optimal controller synthesis problem. In Section 3, we outline the main results from the theory of the widely used A^* algorithm adapted to the near-optimal controller synthesis problem. It is shown how even if over-estimating heuristics are used, near-optimal ("ε-optimal") solutions are found by A^*. Utilizing a general formulation we explain the approaches used in the literature for finding ε-optimal solutions. Then we show several new techniques to "filter the heuristics" (dynamic weighting) which will still result in an ε-optimal solution. Next, we introduce ε-monotonicity and show that it is equivalent to "ε-consistency" and that it automatically implies "ε-admissibility". It is proven that if A^* uses an ε-monotone heuristic then it will find ε-optimal state paths to all states expanded. The heuristic search approach relies on the choice of a "heuristic function" which is used to focus the search for an near-optimal solution. The problem one encounters though is that it is, in general, quite difficult to find an ε-admissible and ε-monotone heuristic function

for many applications. In Section 3.3 we show that the metric space approach used in [7] can be extended and used to specify an ε-admissible and ε-monotone heuristic function in a systematic way for a wide variety of DESs. Hence, the results offer a computationally efficient approach to solving the near-optimal controller synthesis problem. To illustrate the results we provide a solution to a near-minimum-input/event cost problem in Section 4 and study ε-monotone heuristics for the case where it is known that the event costs lie in a certain region. Comparisons to relevant research are made throughout the paper.

2. THE NEAR-OPTIMAL CONTROLLER SYNTHESIS PROBLEM FOR DETERMINISTIC DESs

We consider DESs that can be accurately modelled with

$$P=(X,Q,\delta,\chi,x_0,X_f) \tag{1}$$

where

(i) X is the possibly infinite set of plant states,
(ii) Q is the finite set of plant inputs (controller outputs),
(iii) $\delta:Q\times X\rightarrow X$ is the plant state transition function,
(iv) $\chi:X\times X\rightarrow \mathbb{R}^+$ is the *event cost function*,
(v) x_0 is the initial plant state, and
(vi) $X_f\subset X$ is the non-empty finite set of *final states*.

The model P is limited to representing DESs that are deterministic in the sense that for a given input there is exactly one possible next state. The set

$$E(P)=\{(x,x')\in X\times X:\ x'=\delta(q,x)\}\cup\{(x_d,x_0)\} \tag{2}$$

denotes the (possibly infinite) set of events for our plant P (x_d is a dummy state, and (x_d,x_0) a dummy event added for convenience). The event cost function $\chi(x,x')$ is defined for all $(x,x')\in E(P)$; it specifies the "cost" for each event (state transition) to occur and it is required that there exist a $\delta'>0$ such that $\chi(x,x')\geq\delta'$ for all $(x,x')\in E(P)$. The *discrete event controller* (DEC) is $C=(Q,X,\xi,q_0)$ where (i) Q is the finite set of controller states (plant inputs), (ii) X is the set controller inputs (plant outputs), (iii) $\xi:X\times Q\rightarrow Q$ is the controller transition function, and (iv) q_0 is the initial controller state. The closed loop *discrete event control system* (DECS) is formed by connecting the outputs of the plant (states) to the inputs of the controller and the outputs of the controller to the inputs of the plant. The state of the DECS is given by certain pairs $(q,x)\in Q\times X$. The pair (q_0,x_0) is a valid DECS state where q_0 and x_0 are the initial states of the controller and plant respectively. If (q,x) is a valid state for the DECS then a valid next state is given by (q',x') where $q'=\xi(x,q)$ and $x'=\delta(q',x)$.

Mathematical Preliminaries: State Trajectories and (x,X_z)-Reachability

Let Z be an arbitrary set. Z^* denotes the set of all finite strings over Z including the empty string $\emptyset$. For any $s,t\in Z^*$ such that $s=zz'\cdots z''$ and $t=yy'\cdots y''$, st denotes the concatenation of the strings s and t, and $t\in s$ is used to indicate that t is a substring of s, i.e., $s=zz'\cdots t\cdots z''$. For brevity, the notation $s_{zz''}$ is used to denote a string $s\in Z^*$ such that $s=zz'\cdots z''$ begins with the element $z\in Z$ and ends with $z''\in Z$. Let z_0 be a distinguished member of the set Z. The notation s_z is used to denote a string $s\in Z^*$ such that $s=z_0z'\cdots z$ begins with z_0 and ends with $z\in Z$. Furthermore, $s_{z>}$ denotes a string $s\in Z^*$ such that $s=zz'z''\cdots$ begins with $z\in Z$ and the end element is not specified. A (finite) *cycle* is a string $s\in Z^*$ such that $s=zz'\cdots z''z$ has the same first and last element $z\in Z$. A string $s\in Z^*$ is *cyclic* if it contains a cycle (for $t_{zz}\in Z^*$, $t_{zz}\in s$), and *acyclic* if it does not. Let $|s|$ for $s\in Z^*$ denote the length of string $s\in Z$, i.e., the number of elements of Z concatenated to obtain s.

A string $s \in X^*$ is called a *state trajectory* or *state path* of P if for all successive states $xx' \in s$, $x'=\delta(q,x)$ for some $q \in Q$. Let

$$E_s(P) \subset E(P) \tag{3}$$

denote the set of all events needed to define a particular state path $s \in X^*$ that can be generated by P. For some state path $s=xx'x''x'''\cdots$, $E_s(P)$ is found by simply forming the pairs (x,x'), (x',x''), (x'',x'''),$\cdots$. An *input sequence* $u \in Q^*$ that produces a state trajectory $s \in X^*$ is constructed by concatenating $q \in Q$ such that $x'=\delta(q,x)$ for all $xx' \in s$. Let $X_z \subset X$ and

$$\mathfrak{X}(P,x,X_z) \subset X^* \tag{4}$$

denote the set of all finite state trajectories $s=xx'\cdots x''$ of P beginning with $x \in X$ and ending with $x'' \in X_z$. Then, for instance, $\mathfrak{X}(P,x_0,X_f)$ denotes the set of all finite length state trajectories for P that begin with the initial state x_0 and end with a final state $x \in X_f$.

A plant P is said to be *(x,X_z)-reachable* if there exists a sequence of inputs $u \in Q^*$ that produces an acyclic state trajectory $s \in \mathfrak{X}(P,x,X_z)$. Then, for instance, if P is (x_0,X_f)-reachable we know that there exists an acyclic state trajectory $s=x_0x\cdots x'$ with $x' \in X_f$.

Allowable DES Behavior

The *valid behavior* that the DES can exhibit which is modelled by P can be characterized by the set of all its valid state trajectories $\mathfrak{X}(P,x,X_f)$ where $x \in X$, along with its input sequences (it is specified with the graph of P). Let $P=(X,Q,\delta,\chi,x_0,X_f)$ specify the valid behavior of the plant and

$$A=(X_a,Q_a,\delta_a,\chi_a,x_{a0},X_{af}) \tag{5}$$

be another DES model which we think of as specifying the "allowable" behavior for the plant P. Allowable plant behavior must also be valid plant behavior. Formally, we say that the allowable plant behavior described by A *is contained in* P, denoted with A[P], if the following conditions on A are met:

(i) $X_a \subset X$,
(ii) $Q_a \subset Q$,
(iii) $\delta_a: Q_a \times X_a \rightarrow X_a$ is given by $\delta_a(q,x)=\begin{cases}\delta(q,x) & \text{if } \delta(q,x) \in X_a \\ \text{undefined} & \text{otherwise}\end{cases}$
(iv) $\chi_a: X_a \times X_a \rightarrow \mathbb{R}^+$ is a restriction of $\chi: X \times X \rightarrow \mathbb{R}^+$,
(v) $x_{a0}=x_0$,
(vi) $X_{af} \subset X_f$.

Also, let $E(A) \subset E(P)$ denote the set of *allowable events* defined as in (2). The model A, specified by the designer, represents the "allowable" DES plant behavior which is contained in the valid DES behavior described by the given P. It may be that entering some state, using some input, or going through some sequence of events is undesirable. Such design objectives relating to what is "permissible" or "desirable" plant behavior are captured with A. This formulation is similar in character to the "supervisor synthesis problem" formulated in a language-theoretic framework and solved in [15,16]. There the authors use languages to specify the "acceptable" (allowable) and "legal" (valid) DES behavior. They develop an algorithm to synthesize a supervisor which will make the behavior of the supervisory control system characterized by yet another language to "lie between" the acceptable and legal languages.

The Near-Optimal Controller Synthesis Problem

The *performance index*

$$J: X_a^* \rightarrow \mathbb{R}_+ \tag{6}$$

is defined in terms of the costs of the events by

$$J(s) = \sum_{(x,x')\in E_s(A)} \chi(x,x') \tag{7}$$

for all $s\in \mathfrak{X}(A,x,X_a)$ where $x\in X_a$. By definition, $J(s)=0$ if $s=x$ where $x\in X_a$. As in conventional near-optimal control the objective is to find an input sequence that will produce a state trajectory s such that J(s) is close to its minimum value. Here, we are particularly interested in state trajectories that end in X_{af}. Let $s^*_{x>}\in \mathfrak{X}(A,x,X_{af})$ denote an *optimal state trajectory*, then

$$J(s^*_{x>})=\inf\{J(s_{x>}):s_{x>}\in \mathfrak{X}(A,x,X_{af})\} \tag{8}$$

where $x\in X_a$. Since the graph of A is locally finite there are only a finite number of state trajectories of finite length; hence $J(s^*_{x>})=\min\{J(s_{x>}):s_{x>}\in \mathfrak{X}(A,x,X_a)\}$ where $x\in X_a$. There may, in general, be more than one state path where the minimum is achieved. Let $X_z\subset X_a$. The set of minimum cost state paths for A, beginning at state $x\in X_a$, and ending at state $x'\in X_z$ is denoted by $\mathfrak{X}^*(A,x,X_z)\subset \mathfrak{X}(A,x,X_z)$; consequently, $\mathfrak{X}^*(A,x_0,X_{af})$ denotes the set of optimal allowable state trajectories that begin with x_0 and end in X_{af}. A state trajectory $s\in \mathfrak{X}(A,x,X_{af})$ will be said to be "near-optimal" or "ε-optimal" if

$$J(s)\le(1+\varepsilon)J(s^*) \tag{9}$$

for $s^*\in \mathfrak{X}^*(A,x,X_{af})$ where $\varepsilon\ge 0$. Notice that s and s^* may not end in the same state.

<u>The Near-Optimal Controller Synthesis Problem</u> (NCSP)

Let A describe the allowable behavior for a plant P such that A[P]. Assume that A is (x_0,X_{af})-reachable where x_0 is the initial plant state. Find a controller that will generate a sequence of inputs that drives A along a state trajectory $s\in \mathfrak{X}(A,x_0,X_{af})$ such that $J(s)\le(1+\varepsilon)J(s^*)$ where $J(s^*)=\min\{J(s):s\in \mathfrak{X}(A,x_0,X_{af})\}$ with $\varepsilon\ge 0$.

Once an ε-optimal state trajectory s is found, an input sequence, say $u\in Q^*$ can be constructed to drive P along s. From this a controller C can be constructed. It is possible that some general conventional near-optimal control formulations could be adapted for the above problem but the particular formulation above lends itself to a computationally efficient solution, allows for the specification of, for instance, a "near-minimum-input/event cost" control problem, and is quite useful in applications.

3. NEAR-OPTIMAL CONTROLLER SYNTHESIS VIA HEURISTIC SEARCH

In this section we solve the problem of how to find an ε-optimal allowable state trajectory in A beginning at x_0 and ending in X_{af} assuming that A is (x_0,X_{af})-reachable and A[P] for a given plant P. The approach here, as in [7], is to use the A^* algorithm.

3.1 The A^* Algorithm [3,6,10]

The A^* algorithm is one of the most widely used heuristic search algorithms. It utilizes information about how promising it is that particular state paths are on an optimal state trajectory to reduce the computational complexity. Such information is referred to collectively as "heuristic information". The heuristic information is quantified with the "evaluation function". Since the appropriate information is not available to compute $J(s^*)=J(s^*_x)+J(s^*_{xx'})$ A^* estimates $J(s^*)$ with some easily computable *evaluation function* given by

$$\hat{f}:X^*_a\to\mathbb{R}_+ \tag{10}$$

which is defined for all $s\in X^*_a$ such that $s\in \mathfrak{X}(A,x,X_a)$ where $x\in X_a$. The value of $J(s^*_x)$ is estimated using

$$\hat{g}:X^*_a\to\mathbb{R}_+ \tag{11}$$

where $\hat{g}(s_x)=J(s_x)$ for all $s_x\in \mathfrak{X}(A,x_0,X_a)$. To estimate $J(s^*_{xx'})$, the remaining cost to be incurred from state x to some final state $x'\in X_{af}$, the function

$$\hat{h}:X_a\to\mathbb{R}_+ \tag{12}$$

is used with $\hat{h}(x)=0$ if $x\in X_{af}$. The function $\hat{h}$ is called the "heuristic function" since it provides the facility for supplying the A^* algorithm with special information about the particular search problem under consideration to focus the search of A^*. The proper choice of the heuristic function can result in efficient search. The evaluation function is often chosen to be $\hat{f}(s_x)=\hat{g}(s_x)+\hat{h}(x)$ where $x\in X_a$ is the current state considered by the A^* algorithm. Here we shall consider several other possible choices for $\hat{f}$ and show that these choices result in A^* returning an optimal or near-optimal solution. The evaluation function must take on a certain form as we now discuss. Let s_x and s'_x denote two state trajectories that end in state $x\in X_a$, i.e., $s_x, s'_x\in \mathfrak{X}(A,x_0,x)$. Let $\bar{s}\in \mathfrak{X}(A,x,X_a)$. The evaluation function $\hat{f}(s)$ must be chosen so that it is "order preserving", i.e., that $\hat{f}(s_x)\geq\hat{f}(s'_x) \Rightarrow \hat{f}(s_x\bar{s})\geq\hat{f}(s'_x\bar{s})$. This requirement is a version of the principle of optimality that states that if a path s_x is judged to be more meritous that another path s'_x then no common extension $\bar{s}$ of s_x or s'_x can reverse the judgement [1,10].

The A^* algorithm proceeds by generating candidate state trajectories which are characterized with two sets $C\subset E(A)$ and $O\subset E(A)$. The contents of C and O change at different stages of the algorithm but it is always the case that there does not exist $(x^1,x^2)\in C\cup O$ and $(x^3,x^4)\in C\cup O$ such that $x^2=x^4$ and $x^1\neq x^3$. Let the set of state trajectories of A, investigated by A^*, be denoted by $\mathfrak{X}(A,C,O)$. Each state path $s_{x'}\in \mathfrak{X}(A,C,O)$ begins with x_0, the initial state, and has an end state $x'\in X_a$ such that $(\cdot,x')\in C\cup O$. For $s,s'\in X_a^*$ let $s\leftarrow ss'$ denote the operation of *replacing* s by ss'. To find $s_{x'}\in \mathfrak{X}(A,C,O)$ from C and O choose $(x,x')\in C\cup O$ and let $s=xx'$. Repeat the following steps until x_d is encountered: (a) Find $(x^1,x^2)\in C\cup O$ with $x^2=x$ where $s=x\cdots$, (b) Let $s\leftarrow x^1s$, and go to (a). The operation of finding the set $\mathcal{E}(x)=\{x':x'\in X_a \text{ and } x'=\delta_a(q,x)\}$ is called *expanding the state* $x\in X_a$. For Z and Z' arbitrary sets let $Z\leftarrow Z'$ denote the *replacement* of Z by Z'.

<u>A^*</u>:

(1) Let $C=\{\}$ and $O=\{(x_d,x_0)\}$.

(2) If $|O|>0$, then go to Step 3. If $|O|=0$, then exit with no solution.

(3) Choose $(x,x')\in O$ so that $\hat{f}(s_x x')$ is a minimum (resolve ties arbitrarily).
Let $O\leftarrow O-\{(x,x')\}$ and $C\leftarrow C\cup\{(x,x')\}$.

(4) If $x'\in X_{af}$ then exit with $s_{x'}\in \mathfrak{X}^*(A,x_0,X_{af})$, a (near-) optimal state trajectory.

(5) For each $x''\in \mathcal{E}(x')$:

(i) If for all $\bar{x}\in X_a$, $(\bar{x},x'')\notin C\cup O$ then let $O\leftarrow O\cup\{(x',x'')\}$.

(ii) If there exists $\bar{x}\in X_a$ such that $(\bar{x},x'')\in O$ and $\hat{f}(s_{x'}x'')<\hat{f}(s_{\bar{x}}x'')$ then let $O\leftarrow O-\{(\bar{x},x'')\}$ and $C\leftarrow C\cup\{(x',x'')\}$.

(iii) If there exists $\bar{x}\in X_a$ such that $(\bar{x},x'')\in C$ and $\hat{f}(s_{x'}x'')<\hat{f}(s_{\bar{x}}x'')$then let $C\leftarrow C-\{(\bar{x},x'')\}$ and $O\leftarrow O\cup\{(x',x'')\}$.

A^* is said to be *complete* since it terminates with a solution. A heuristic function $\hat{h}(x)$ is said to be *admissible* if $0\leq\hat{h}(x)\leq J(s^*_{x>})$ for all $x\in X_a$ such that $s^*_{x>}\in \mathfrak{X}^*(A,x,X_{af})$. Let $A^*(\hat{h}(x))$ denote an A^* algorithm which uses $\hat{h}(x)$ as its heuristic function. If $\hat{h}(x)$ is admissible then $A^*(\hat{h}(x))$ is said to be *admissible* since it is guaranteed to find an optimal state trajectory when one exists, i.e., when A is (x_0,X_{af})-reachable. A heuristic function $\hat{h}(x)$ is said to be *monotone* if $\hat{h}(x)\leq\chi(x,x')+\hat{h}(x')$ for all $(x,x')\in E(A)$. A heuristic function $\hat{h}(x)$ is said to be *consistent* (equivalent to being monotone) if $\hat{h}(x)\leq J(s^*_{xx'})+\hat{h}(x')$ for all $(x,x')\in E(A)$ where $s^*_{xx'}\in \mathfrak{X}^*(A,x,x')$. If $\hat{h}(x)$ a monotone heuristic function then $A^*(\hat{h}(x))$ finds optimal paths to all expanded states, i.e., $\hat{g}(s_x)=J(s^*_x)$ for all $x\in X_a$ with $(\cdot,x)\in C$, $s_x\in \mathfrak{X}(A,C,O)$, and $s^*_x\in \mathfrak{X}^*(A,x_0,x)$. The real utility of knowing that $\hat{h}(x)$ is monotone lies in the fact that states are expanded at most once. This implies that the A^* algorithm can be simplified by removing Step 5 (iii) since events (pointers) will never be taken from C and placed in O.

3.2 The A^* Algorithm for Finding Near-Optimal Solutions

In this section we consider the evaluation function

$$\hat{f}_\varepsilon(s_x)=\alpha_g\hat{g}(s_x)+\hat{h}(x) \tag{13}$$

with $0<\alpha_g\leq 1$ which appears similar to the standard A^* algorithm except we shall, at times, allow the heuristic function to overestimate the remaining cost to find a solution.

The heuristic function $\hat{h}(x)$ is said to be *ε-admissible* if

$$\hat{h}(x)\leq(1+\varepsilon)J(s^*_{x>}) \tag{14}$$

for all $x\in X_a$ and $s^*_{x>}\in\mathfrak{X}^*(A,x,X_{af})$ where $\varepsilon\geq 0$. A^* is said to be ε-admissible if it returns an ε-optimal solution (state trajectory).

<u>Theorem 1</u>: If $\hat{h}(x)$ is ε-admissible and $0<\alpha_g\leq 1$ then $A^*(\hat{h}(x))$ finds a state trajectory $s\in\mathfrak{X}(A,x_0,X_{af})$ such that $J(s)\leq(1+\varepsilon)J(s^*)/\alpha_g$ where $s^*\in\mathfrak{X}^*(A,x_0,X_{af})$.

Proof: The fact that A^* terminates with a solution is proven in a manner similar to the way termination is proven in [3,10]. Assume the opposite, i.e., that A^* terminates with $x_f\in X_{af}$ so that $\hat{f}_\varepsilon(s_{x_f})>(1+\varepsilon)J(s^*)/\alpha_g$ for $s_{x_f}\in\mathfrak{X}(A,x_0,X_{af})$ and $s^*\in\mathfrak{X}^*(A,x_0,X_{af})$. When x_f was chosen for expansion, for each $(\cdot,x)\in O$ it satisfied $\hat{f}_\varepsilon(s_{x_f})\leq\hat{f}_\varepsilon(s_x)$ where $s_x\in\mathfrak{X}(A,C,O)$. But notice that an optimal path s^* is assumed to exist. For this particular path at any step before termination there must exist at least one state, say x' such that $x'\in s^*$ such that $(\cdot,x')\in O$. Split s^* at state x', i.e., let $s^*=s^*_{x'}s^*_{x'>}$. Also, choose x' so that $|s^*_{x'}|$ is of minimum length, i.e. so that x' is the first state on s^* such that $(\cdot,x')\in O$. Then all the ancestors of x', say x_a have $(\cdot,x_a)\in C$; therefore $\hat{g}(s^*_{x'})=J(s^*_{x'})$. Using the assumption of ε-admissibility $\hat{f}_\varepsilon(s^*_{x'})=\alpha_g\hat{g}(s^*_{x'})+\hat{h}(x')\leq(1+\varepsilon)J(s^*_{x'})+(1+\varepsilon)J(s^*_{x'>})=(1+\varepsilon)J(s^*)$. Hence, before termination there will always exist a state $x'\in s^*$ such that $(\cdot,x')\in O$ and $\hat{f}_\varepsilon(s^*_{x'})\leq(1+\varepsilon)J(s^*)$; hence, $\hat{f}_\varepsilon(s_{x_f})>(1+\varepsilon)J(s^*)/\alpha_g$. This is a contradiction so A^* terminates with $x_f\in X_{af}$ returns s_{x_f} such that $J(s_{x_f})\leq(1+\varepsilon)J(s^*)/\alpha_g$. ■

This result is proven in [9] for the case of $\alpha_g=1$. A slightly different approach is used in [2] where the author introduces a "bandwidth constraint" for the heuristic function and shows that A^* will return an near-optimal solution. Let

$$z_h:X_a\rightarrow\mathbb{R}_+ \tag{15}$$

with $0\leq z_h(x)\leq(1+\varepsilon)$ denote what will be called a "$\hat{h}$-filter" that is used to weight the heuristics. Using Theorem 1 we can overview the results in the literature on finding ε-optimal solutions with A^*. First, consider the case where $\hat{g}$ and $\hat{h}$ are weighted but A^* still returns an <u>optimal</u> solution. Choosing $z_h(x)=\omega$ for all $x\in X_a$ and $\alpha_g=(1-\omega)$ with $0\leq\omega\leq\frac{1}{2}$ results in an $A^*(\hat{h}(x))$ that is admissible provided $\hat{h}(x)=z_h(x)\hat{h}'(x)$ with $\hat{h}'(x)$ admissible. This choice was first considered by Pohl in [12] where he showed that ω can be varied to emphasize the breadth-first (ω small) or depth-first (ω large) components of A^*'s search. The following Corollary to Theorem 1 provides another method to weight the components of the evaluation function.

<u>Corollary 1</u>: If $z_h(x)=\alpha_h$ where $0\leq\alpha_h\leq\alpha_g$ and $\hat{h}(x)=z_h(x)\hat{h}'(x)$ where $\hat{h}'(x)$ is admissible then $A^*(\hat{h}(x))$ is admissible.

This result generalizes Pohl's weighting scheme in [12] by showing how the weights can be set so that optimality can still be obtained. For the remainder of the paper we focus on the case where $\mathbf{A}^*$ returns ε-optimal solutions and $\alpha_g=1$.

Corollary 2: If $\hat{h}(x)=z_h(x)\hat{h}'(x)$ where $\hat{h}'(x)$ is an admissible heuristic function and $\alpha_g=1$ then $\mathbf{A}^*(\hat{h}(x))$ is ε-admissible.

Using Corollary 2 we first outline the results in the literature. Let d(x) denote the depth of state x (i.e., $d(x)=|s_x|$) and N be an upper bound on the depth of the state at the end of the longest path investigated by $\mathbf{A}^*$ before termination; hence, $0\le d(x)\le N$. In [13,14] the author used $z_h(x)=(1+\varepsilon(1-d(x)/N))$ ("dynamic weighting") and $\alpha_g=1$ to obtain a search that will begin with a strong depth-first component and finish with a strong breadth-first component to search. In [9] the authors generalized the results in [13,14] by showing that $\alpha_g=1$ and $z_h(x)=(1+\varepsilon z_\rho(x))$, where $z_\rho(x)$ is defined so that $0\le z_\rho(x)\le 1$ will result in $\mathbf{A}^*$ returning an ε-optimal solution. Corollary 2 slightly generalizes Pearl and Kim's result by showing (i) the full range of values that the filter $z_h(x)$ can take on and still guarantee ε-admissibility, and (ii) that $\hat{g}(s_x)$ can also be weighted with α_g. In [9] the authors also introduce another approach to ε-optimal search via a more complex algorithm than $\mathbf{A}^*$ called $\mathbf{A}^*_\varepsilon$. They argue that the advantage of $\mathbf{A}^*_\varepsilon$ over the approach in [13,14] is that it allows for loading heuristic information into another heuristic function and that N does not have to be known a priori. (For a comparison of $\mathbf{A}^*_\varepsilon$ to Pohl's dynamic weighting approach see [9,10].) Next, we show how the flexibility in choosing the $\hat{h}$-filter also allows the designer to load heuristic information into the evaluation function and no a priori information such as N is needed.

We introduce several candidate $\hat{h}$-filters $z_h(x)$ for the case where $\alpha_g=1$. We focus on the case where it increases the computational efficiency of $\mathbf{A}^*$ to alternate between emphasis on a breadth-first component to a depth-first component of search. Such situations occur quite often when there is a portion of the graph which is more efficiently searched with one or the other of these techniques. Suppose, for instance, that at some depth N_s it is most beneficial to switch from breadth-first to depth-first search (or vice versa) then the following $\hat{h}$-filters can be utilized:

(1) $z_h(x)=(1+\varepsilon)\left[\dfrac{d(x)}{N_s+d(x)}\right]$ (for switching from breadth-first to depth-first)

(2) $z_h(x)=(1+\varepsilon)\left[\dfrac{N_s}{N_s+d(x)}\right]$ (for switching from depth-first to breadth-first)

Another possibility is that certain regions of the graph of A can be more efficiently searched with different emphasis on breadth or depth-first search. In this case partition X_a into sets X_{ai} for i=1,2, ... n. Then, let $z_h(x)=z_h(x')$ for all $x,x'\in X_{ai}$ for each i=1,2, ... n. This will allow for a switching in the emphasis between breadth and depth-first search as the algorithm expands states in different parts of the graph of A.

Next, we introduce a new condition on $\hat{h}(x)$ which we call "ε-monotonicity" and study several properties of $\mathbf{A}^*$ algorithms that operate with ε-monotone heuristics. The heuristic function $\hat{h}(x)$ is said to be *ε-monotone* if $\hat{h}(x)\le(1+\varepsilon)\chi(x,x')+\hat{h}(x')$ for all $(x,x')\in E(A)$. The heuristic function $\hat{h}(x)$ is said to be *ε-consistent* if $\hat{h}(x)\le(1+\varepsilon)J(s^*_{xx'})+\hat{h}(x')$ for all $x,x'\in X_a$ where $s^*_{xx'}\in \mathfrak{X}^*(A,x,x')$.

Theorem 2: If $\hat{h}(x)$ is ε-consistent then it is ε-admissible ($\hat{h}(x)$ is ε-monotone iff it is ε-consistent).

Proof: For the first part of the Theorem let $s^*_{xx'} \in \mathfrak{X}^*(A,x,X_{af})$ for any $x \in X_a$, then $\hat{h}(x')=0$ and $\hat{h}(x) \leq (1+\varepsilon)J(s^*_{xx'})$ for all $x \in X_a$. For the second part clearly if $\hat{h}(x)$ is ε-consistent it is ε-monotone. The proof for the other direction follows by repeated application of the ε-monotone condition along an optimal path $s^*_{xx'} \in \mathfrak{X}^*(A,x,x')$. ■

<u>Theorem 3</u>: If $\hat{h}(x)$ is ε-monotone then $A^*(\hat{h}(x))$ finds ε-optimal paths to all expanded states, i.e., $\hat{g}(s_x) \leq (1+\varepsilon)J(s^*_x)$ for all $x \in X_a$ with $(\cdot,x) \in C$, $s_x \in \mathfrak{X}(A,C,O)$, and $s^*_x \in \mathfrak{X}^*(A,x_0,x)$.

Proof: Assume that $A^*(\hat{h}(x))$ selects a state x for expansion so that $\hat{g}(s_x) > (1+\varepsilon)J(s^*_x)$. If $(\cdot,x) \in O$ is the only pair in O then all ancestors of x have been expanded and $\hat{g}(s_x)=J(s^*_x)$. The assumption that $\hat{g}(s_x) > (1+\varepsilon)J(s^*_x)$ implies that s^*_x contains at least one more state $x' \in s^*_x$ such that $(\cdot,x') \in O$. Let $s^*_x = s^*_{x'} s^*_{x'x}$ where x' minimizes $|s^*_{x'}|$. Now, it will be shown that x' and not x (as assumed above) should be selected for expansion. From the proof of Theorem 1 $\hat{g}(s_{x'})=J(s^*_{x'})$ and since $\hat{h}(x)$ is assumed ε-consistent

$$\hat{f}_\varepsilon(s_{x'})=\hat{g}(s_{x'})+\hat{h}(x') \leq (1+\varepsilon)J(s^*_{x'})+(1+\varepsilon)J(s^*_{x'x})+\hat{h}(x)=(1+\varepsilon)J(s^*_x)+\hat{h}(x)$$

The assumption $\hat{g}(s_x) > (1+\varepsilon)J(s^*_x)$ gives $\hat{f}_\varepsilon(s_{x'}) < \hat{f}_\varepsilon(s_x)$ so x' will be expanded which is a contradiction. Therefore, $\hat{g}(s_x) \leq (1+\varepsilon)J(s^*_x)$ for any state x expanded. ■

The approach used to prove Theorems 1, 2, and 3 is similar to the approaches used to prove the analogous results in standard A^* theory [3,6,10]. Unfortunately, unlike the case for monotone heuristics, we cannot guarantee that Step 5 (iii) of the A^* algorithm can be removed; consequently it is possible that A^* will re-expand states that have already been expanded. Since, $\hat{h}(x)-\hat{h}(x') \leq (1+\varepsilon)\chi(x,x')$, for relatively small ε we expect A^* to focus its search nearly as well as the case where A^* operates with a monotone heuristic function.

3.3 Specification of the Heuristic Function

There has been extensive work on the problem of how to automatically generate heuristics for an arbitrary problem. These techniques have been outlined in [7]. Although all these results are somewhat relevant, none directly answer the question of how to find a heuristic function that will be ε-admissible and ε-monotone. This is the problem that is addressed here. We show that the approach developed in [7] can be easily extended so that ε-admissible and ε-monotone heuristics can be specified for a wide variety of DESs. As in [7] the approach taken here relies on the use of metric spaces. Let Z be an arbitrary non-empty set and let $\rho: Z \times Z \rightarrow \mathbb{R}$ where ρ has the following properties: (i) $\rho(x,y) \geq 0$ for all $x,y \in Z$ and $\rho(x,y)=0$ iff $x=y$, (ii) $\rho(x,y)=\rho(y,x)$ for all $x,y \in Z$, and (iii) $\rho(x,y) \leq \rho(x,z)+\rho(z,y)$ for all $x,y,z \in Z$ (triangle inequality). The function ρ is called a *metric* on Z and $\{Z;\rho\}$ is a *metric space*. Let $p \in Z$ and define $d(p,Z)=\inf\{\rho(p,z): z \in Z\}$. The value of $d(p,Z)$ is called the *distance between point p and set Z*. Let $\Delta(Z)$ denote the class of functions that are metrics on the set Z.

<u>Theorem 4</u>: For the DES P and A[P] if $\hat{h}(x)=\inf\{\rho(x,x_f): x_f \in X_{af}\}$ and $\rho \in \Delta(X_a)$ with $\rho(x,x') \leq (1+\varepsilon)\chi(x,x')$ for all $(x,x') \in E(A)$ then $\hat{h}(x)$ is ε-admissible and ε-monotone.

Proof: For ε-monotonicity let $s_{\bar{x}x''} \in \mathfrak{X}(A,\bar{x},A_{af})$ where $\bar{x} \in X_a$ and let $xx' \in s_{\bar{x}x''}$ be two successive states on $s_{\bar{x}x''}$. Notice that for the sequence of states $x \in X_a$ expanded, the state at which the inf is achieved in $\hat{h}(x)=\inf\{\rho(x,x_f):x_f \in X_{af}\}$ may change. Let x_p denote the state at which the inf is achieved for x and x_p' the one for x'. By the triangle inequality, $\rho(x,x_p') \le \rho(x,x')+\rho(x',x_p')$. But by the definition of $\hat{h}(x)$ we know that $\rho(x,x_p) \le \rho(x,x_p')$. It follows that $\rho(x,x_p) \le \rho(x,x')+\rho(x',x_p')$. By the definition of $\hat{h}(x)$ we have $\hat{h}(x) \le \rho(x,x')+\hat{h}(x')$ and since $\rho(x,x') \le (1+\varepsilon)\chi(x,x')$, $\hat{h}(x) \le (1+\varepsilon)\chi(x,x')+\hat{h}(x')$ for all $x,x' \in X_a$ such that $xx' \in s$ where $s \in \mathfrak{X}(A,\bar{x},X_{af})$ which guarantees the ε-monotonicity of $\hat{h}(x)$. From Theorem 2, ε-monotonicity implies ε-admissibility so the proof is done. ■

We have still not said how to specify the exact form of the heuristic function for particular applications. There is a wide class of DESs whose state space can be modelled in terms of $X \subset \mathbb{R}^n$. As evidence of this fact we turn to the many applications of the theory of Petri nets [11] (e.g. General or Extended Petri nets) where the states are n-tuples of natural numbers or the more recent work in [4] where numerical n-tuples are used. It is easy to specify a wide variety of metrics on $\mathbb{R}^n$ [5]; hence most often there is no problem in finding a heuristic function for P.

4. EXAMPLES

In this section we briefly outline two examples which illustrate some of the above results. First, we consider ε-optimal search when it is known that the event costs for P defined by (1) satisfy $\alpha \le \chi(x,x') \le \beta$ for all $(x,x') \in E(A)$. This occurs quite frequently in applications. Suppose that via the results in [7] and Theorem 4 we choose a heuristic function $\hat{h}(x)=\inf\{\rho(x,x_f):x_f \in X_{af}\}$ with $\rho \in \Delta(X_a)$ and $\rho(x,x') \le \beta$ for all $(x,x') \in E(A)$. In this case we are allowing the heuristic function to over-estimate. The near-optimality of the solution returned depends on α and β. The values of ε such that $(1+\varepsilon)\chi(x,x') \ge \beta$ will result in an ε-monotone $\hat{h}(x)$. Hence, provided $\varepsilon \ge (\beta-\alpha)/\alpha$ we have a ε-monotone heuristic so A^* will return an ε-optimal solution. Intuitively, if α is close to β, ε can be chosen to be small and a solution more near the optimal one is found. If $\beta-\alpha$ is large then the original choice for $\hat{h}(x)$ results in a heuristic that will over-estimate by large amounts and solutions farther away from an optimal one are found.

Next, we study the *near-minimum-input/event cost control* of deterministic DESs. Consider a DES modelled with P as defined in (1) except we shall define χ in a special manner. Let $\chi_i:Q \to \mathbb{R}^+$ and require that there exists a $\delta'>0$ such that $\chi_i(q) \ge \delta'$ for all $q \in Q$ and let $\chi_e:X \times X \to \mathbb{R}_+$ be defined for all $(x,x') \in E(P)$. The function χ_i specifies the *cost of each input* and χ_e specifies the cost of each event (state transition) as in (1). For all $q \in Q$, $x,x' \in X$ such that $x'=\delta(q,x)$ define $\chi(x,x')=\chi_i(q)+\chi_e(x,x')$. The function χ then specifies the combined cost of the input and event. Given A such that A[P] and $A^*(\hat{h}(x))$ where $\hat{h}(x)$ is ε-admissible will result in an ε-optimal solution to the NCSP.

We shall discuss different approaches to specifying the heuristic function $\hat{h}(x)$ and illustrate the relationship between ε-monotonicity and monotonicity for this example. At times it is easier to specify a heuristic function in terms of χ_e (rather than χ) since there is a close correspondence between the distance $\rho(x,x')$ between two states x and x' and $\chi_e(x,x')$. Suppose that this is the case and that via Theorem 4, $\hat{h}(x)=\inf\{\rho(x,x_f):x_f \in X_{af}\}$ with $\rho \in \Delta(X_a)$ and $\rho(x,x') \le (1+\varepsilon)\chi_e(x,x')$ for all $(x,x') \in E(A)$ so that $\hat{h}(x)$ is ε-monotone relative to the costs

$\chi_e(x,x')$. If the input costs satisfy $\chi_i(q) \geq \varepsilon\chi_e(x,x')$ for $\varepsilon>0$ where $x'=\delta(q,x)$ then $\rho(x,x') \leq \chi_e(x,x')+\chi_i(q)$; hence, $\hat{h}(x)$ is monotone. This example shows how to specify a monotone heuristic via an ε-monotone one. It also shows that an optimal path (in terms of the costs $\chi(x,x')$) will be returned by A^* but an ε-optimal path in terms of the costs $\chi_e(x,x')$ is returned. Another possibility is to use the results of [7] to find a heuristic function $\hat{h}(x)=\inf\{\rho(x,x_f): x_f \in X_{af}\}$ with $\rho \in \Delta(X_a)$ and $\rho(x,x') \leq \chi(x,x')$ for all $(x,x') \in E(A)$ so that $\hat{h}(x)$ is monotone. Notice that in this case if there exists $\varepsilon>0$ such that $\chi_i(q) \leq \varepsilon\chi_e(x,x')$ whenever $x'=\delta(q,x')$ then $\rho(x,x') \leq (1+\varepsilon)\chi_e(x,x')$ for all $(x,x') \in E(A)$ so that $\hat{h}(x)$ is ε-monotone relative to the costs $\chi_e(x,x')$ (similar to the above result). Work continues on illustrating the results in this paper on physical examples.

Acknowledgment: The authors gratefully acknowledge the partial support of the Jet Propulsion Laboratory.

References

[1] Bellman R., Dynamic Programming, Princeton Univ. Press, NJ, 1957.

[2] Harris L.R., "The Heuristic Search Under Conditions of Error", Artificial Intelligence, Vol. 5, pp. 217-234, 1974.

[3] Hart P.E., Nilsson N.J., Raphael B.,"A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Trans. on Systems Science and Cybernetics, Vol. SSC-4, No. 2, pp. 100-107, July 1968.

[4] Li Y., Wonham W.M., "A State-Variable Approach to the Modelling and Control of Discrete-Event Systems", Proc. of the 26th Allerton Conf. on Communication, Control, and Computing, pp. 1140-1149, Univ. of Illinois at Champaign-Urbana, Sept. 1988.

[5] Michel A.N., Herget C.J., Mathematical Foundations in Engineering and Science: Algebra and Analysis, Prentice-Hall, NJ, 1981.

[6] Nilsson N.J., Principles of Artificial Intelligence, Tioga Pub., NY, 1980.

[7] Passino K.M., Antsaklis P.J., "On the Optimal Control of Discrete Event Systems", To appear in the Proc. of the Conf. on Decision and Control, Tampa Florida, Dec. 1989.

[8] Passino K.M., Analysis and Synthesis of Discrete Event Regulator Systems, Ph.D. Dissertation, Dept. of Electrical and Computer Eng., Univ. of Notre Dame, April 1989.

[9] Pearl J., Kim J.H., "Studies in Semi-Admissible Heuristics", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-4, No. 4, pp. 392-399, July 1982.

[10] Pearl J., Heuristics: Intelligent Search Strategies for Computer Problem Solving, Addison-Wesley, Reading, Mass., 1984.

[11] Peterson J.L., Petri Net Theory and the Modeling of Systems, Prentice Hall, NJ, 1981.

[12] Pohl I., "Heuristic Search Viewed as Path Finding in a Graph", Artificial Intelligence, Vol. 1, pp. 193-204, 1970.

[13] Pohl I. "Practical and Theoretical Considerations in Heuristic Search Algorithms", in E.W. Elcock, D. Michie ed., Machine Intelligence 8, Ellis Horwood, NY, 1977.

[14] Pohl I., "The Avoidance of (relative) Catastrophe, Heuristic Competence, Genuine Dynamic Weighting and Computational Issues in Heuristic Problem Solving", Proc. of the 3rd Int. Joint Conf. on Artificial Intelligence, pp. 12-17, 1973.

[15] Ramadge P.J., Wonham W.M., "Supervisory Control of a Class of Discrete Event Processes", SIAM J. Control and Optimization, Vol. 25, No. 1, Jan. 1987.

[16] Wonham W.M., Ramadge P.J., "On the Supremal Controllable Sublanguage of a Given Language", SIAM J. Control and Optimization, Vol. 25, No. 3, May 1987.