Proceedings of the 28th Conference
on Decision and Control
Tampa, Florida • December 1989

ON THE 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 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 discrete event system model P is thought of as characterizing the *valid* dynamical behavior of the physical plant. Another DES model A represents design objectives which specify the *allowable DES behavior* which is "contained in" the valid behavior. Assuming that the allowable behavior can be attained, 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 not only should the controller choose the plant inputs so that the plant behavior is allowable but that it should also, in some sense, be "optimal". It is within this context that we formulate an "optimal controller synthesis problem", i.e. how to construct a controller to achieve 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 a "heuristic function" which is used to focus the search for an optimal solution. The problem one encounters though is that it is, in general, quite difficult to find an appropriate heuristic function for particular applications. This is resolved here, and in fact we extend the theory of heuristic search by showing that a metric space approach can be used to specify such 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 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 were originally established in [31] and related work is given in [30].) A new DES controller synthesis methodology is introduced in Section 2. The technique requires viewing the plant model P as characterizing the *valid DES behavior*. 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 optimal. To quantify this, a performance index is defined in terms of the costs for the events to occur. The optimal controller synthesis problem for deterministic DES 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 minimize the performance index. This will result in "controller synthesis for optimal allowable DES behavior".

Our solution to this optimal control problem (in Section 3) 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 an optimal solution among all the possible solutions to the optimal controller synthesis problem. We outline the main results from the theory of the widely used A^* algorithm adapted to the controller synthesis problem and discuss computational complexity. The heuristic search approach relies on the choice of a "heuristic function" which is used to focus the search for an optimal solution. The problem one encounters though is that it is, in general, quite difficult to find an appropriate heuristic function for many applications. This is in part due to the fact that it is imperative that the heuristic function be "admissible" and desirable for it to satisfy a "monotone condition". In Section 4 we extend the theory of heuristic search by showing that a metric space approach can be used to specify the heuristic function in a systematic way for a wide variety of DESs. It is shown that our approach to specifying heuristic functions will always result in their satisfying the admissibility and monotone conditions. In cases where it is known that the costs of the events can be specified with a metric it is proven that we can expect computational complexity to be further reduced. Hence, the results offer a computationally efficient solution to the optimal controller synthesis problem. To illustrate the results we provide a solution to a minimum-time and a minimum-input/event cost problem and discuss applications to artificial intelligence systems in Section 5. Comparisons to relevant research are made throughout the paper.

2. THE 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. A state transition can occur in a non-deterministic fashion relative to time so certain asynchronous DESs can be modelled. 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 addition of the cost for an event to occur is a new addition to the standard automaton model P. It allows for the development of a theory on the optimal control of logical DESs and the analysis of an important class of applications. Intuitively, we think of "changes", i.e. state transitions that occur in the system, as having an associated cost. Certain events or sequences of events may then be more desirable since they will result in a lower "overall cost" to be defined precisely below.

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 Ø. 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

CH2642-7/89/0000-2713$1.00 © 1989 IEEE

$$E_s(P) \subset E(P) \quad (3)$$

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'''),.... 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

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

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 similar use of state trajectories in DESs is found in, for instance, [26].

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

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

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 [35,38]. 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.

In controller synthesis the design objectives often specify what plant behavior is desirable with a model. Then using the plant model and this model of the desired behavior a controller is synthesized to manipulate the plant inputs to ensure that the plant behavior is desirable. Often, optimal plant behavior is desired so the controller must select the inputs to minimize some performance index and achieve desirable behavior. It is within this context that we find the above approach using allowable DES behavior intuitively appealing as a control theoretic approach to optimal controller synthesis for DESs. Moreover, it appears promising for future extensions and developments.

The Optimal Controller Synthesis Problem

The *performance index*

$$J: X_a^* \rightarrow \mathbb{R}_+ \quad (6)$$

is defined in terms of the costs of the events by

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

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 optimal control the objective is to find an input sequence that will produce a state trajectory s that will minimize J(s). Here, we are particularly interested in state trajectories that end in X_{af}. Let $s^*_{x>} \in \mathfrak{X}(A,x,X_{af})$ denote such an *optimal state trajectory*, then

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

where $x \in X_a$. Notice that the inf is achieved for $|s|$ finite, indeed $|J(s)| \rightarrow \infty$ if $|s| \rightarrow \infty$ since $\chi(x,x')>\delta'$ for all $(x,x') \in E(A)$. Since the graph of A is locally finite there are only a finite number of state trajectories of finite length; hence

$$J(s^*_{x>}) = \min\{J(s_{x>}): s_{x>} \in \mathfrak{X}(A,x,X_a)\} \quad (9)$$

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

$$\mathfrak{X}^*(A,x,X_z) \subset \mathfrak{X}(A,x,X_z). \quad (10)$$

We are concerned with $\mathfrak{X}^*(A,x_0,X_{af})$ the set of optimal allowable state trajectories that begin with x_0 and end in X_{af}. Next, we state the optimal controller synthesis problem for P.

The Optimal Controller Synthesis Problem (CSP)

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 the system A along a state trajectory $s^* \in \mathfrak{X}(A,x_0,X_{af})$ such that $J(s^*)=\min\{J(s): s \in \mathfrak{X}(A,x_0,X_{af})\}$.

Once an 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. An alternative optimal control formulation for an automaton can be found in [16] or some general conventional optimal control formulations could also be adapted. The particular formulation above lends itself to a computationally efficient solution, allows for the specification of "minimum-time" and "minimum-input/event cost" optimal control problems, and is quite useful in applications.

In summary, to perform controller synthesis to achieve optimal allowable DES behavior the following steps are taken:

(i) Model the DES with P,

(ii) Specify the allowable DES behavior with A such that A[P], and

(iii) Find a sequence of inputs that will produce an "optimal" allowable DES behavior and construct the controller

Assuming that the first two steps have been taken, Step (iii) is now explained in detail. This amounts to solving the above optimal control problem for DESs.

3. OPTIMAL CONTROLLER SYNTHESIS VIA HEURISTIC SEARCH

In this section we solve the problem of how to find an 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 is to use a *search algorithm* to successively generate candidate state trajectories until an optimal one is found. A brute-force approach to solving this problem may produce an algorithm whose computational complexity would prohibit solving all but the simplest of controller synthesis problems (CSPs). Here we use an approach which seeks to minimize the number of state trajectories considered as we discuss next.

The solution to the CSP could be obtained by adapting certain shortest path algorithms (See, for instance, [1,7,3]). Here, however, the focus is on using an algorithm which will produce a computationally efficient solution to the shortest path problem for a wide variety of plants where certain information about the plant may be known and used to improve search efficiency. Also, we are concerned with the case where $|X|$ can be infinite, the graph of P can be defined *implicitly* rather than explicitly, and we search for the shortest path to a set of states. One could use a conventional dynamic programming solution [2] but due to the problem of state space explosion often found using logical DES models [13,36] such methods can result in an inefficient algorithm with large memory requirements. Often, a branch and bound technique is chosen in such situations to produce either optimal or near-optimal solutions (See, for instance, [18,17,22,7]). This is the approach taken here. We use a particular class of branch and bound algorithms called "heuristic"

search algorithms which utilize the "principle of optimality" of dynamic programming [2] and the advantages of branch and bound algorithms that allow certain candidate solutions (state trajectories) to be eliminated from consideration by using information from the plant. The particular heuristic search algorithm used here is called the "$\mathbf{A}^*$ algorithm". In [14,23] it is shown how $\mathbf{A}^*$ is actually a branch and bound technique while in [7] it is discussed how $\mathbf{A}^*$ is a generalization of Dijkstra's and Moore's shortest path algorithms. The main results for the theory of heuristic search using the $\mathbf{A}^*$ algorithm were established in [10,11,6]. The formal properties of $\mathbf{A}^*$ are given in [24,25] or in [33] and are summarized below.

Heuristic search techniques have been applied to problems where computational complexity of search problems is either very high or intractable. The $\mathbf{A}^*$ 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" (part of which is the heuristic function).

The Evaluation Function

The definition of the evaluation function depends on the performance index and what information is available to $\mathbf{A}^*$ at the various stages where the evaluation function is computed. Suppose that some state path $s=s_x s_{x>}$ and $s\in \mathfrak{X}(A,x_0,X_a)$, then $J(s)=J(s_x)+J(s_{x>})$. If $s^*\in \mathfrak{X}^*(A,x_0,X_{af})$ and $s^*=s^*_x s^*_{x>}$ then $J(s^*)=J(s^*_x)+J(s^*_{xx'})$ where $J(s^*_x)=\min\{J(s_x): s_x\in \mathfrak{X}(A,x_0,x)\}$ and $J(s^*_{xx'})=\min\{J(s_{xx'}): s_{xx'}\in \mathfrak{X}(A,x,X_{af})\}$. If $x_b\in X_a$ is some state that is not on any optimal state trajectory and $x_b\in s$ for some $s\in \mathfrak{X}(A,x_0,X_{af})$ then $J(s)>J(s^*)$ for $s^*\in \mathfrak{X}(A,x_0,X_{af})$. This shows that J(s) obeys the "principle of optimality" of dynamic programming [2] stating that a path is optimal iff each segment of it is optimal. Also, J(s) is a "perfect discriminator" since it can provide an exact indication of when states are not on an optimal state trajectory; hence using J(s) as defined above with $\mathbf{A}^*$ would yield a perfect search strategy which would only consider states on optimal state trajectories [33]. The problem is that the appropriate information is not available for the $\mathbf{A}^*$ algorithm to compute $J(s^*)=J(s^*_x)+J(s^*_{xx'})$. Suppose that the current path found to x is s_x. It is, in general, not known if $s_x=s^*_x$ so $J(s^*_x)$ cannot be computed. For $J(s^*_{xx'})$ the entire path $s_{xx'}$ (optimal or not) is not known (this is what the algorithm is searching for). If $\mathbf{A}^*$ tried to compute the extra information by searching the graph of P the main objective is not met, which is to avoid excessive search [33].

For the $\mathbf{A}^*$ algorithm $J(s^*)$ is estimated by some easily computable *evaluation function* given by

$$\hat{f}: X_a^* \to \mathbb{R}_+ \tag{11}$$

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 evaluation function $\hat{f}$ is obtained by approximating both $J(s^*_x)$ and $J(s^*_{xx'})$ with appropriately defined functions. Let $s=s_x s_{x>}$ be some state path. The value of $J(s^*_x)$ will be estimated using

$$\hat{g}: X_a^* \to \mathbb{R}_+ \tag{12}$$

where $\hat{g}(s_x)=J(s_x)$ for all $s_x\in \mathfrak{X}(A,x_0,X_a)$. Note that $\hat{g}(s_x)=0$ if $s_x=x_0$ the initial state of A. The function $\hat{g}(s_x)$ is used to keep track of the cost expended to reach the current state x. Hence, it can use the entire path from x_0 to x to produce its estimate. Notice that it is not necessarily the case that $\hat{g}(s_x)=J(s^*_x)$ where $s^*_x\in \mathfrak{X}^*(A,x_0,x)$. Intuitively, $J(s^*_x)$ is estimated with the cost of the current path found to x. To estimate $J(s^*_{xx'})$, the remaining cost to be incurred from state x to some final state $x'\in X_{af}$, the function

$$\hat{h}: X_a \to \mathbb{R}_+ \tag{13}$$

is used with $\hat{h}(x)=0$ if $x\in X_{af}$. For $\hat{h}$, generally speaking, the only information available to determine an estimate of $J(s^*_{xx'})$ is the current state under consideration and the fixed set X_{af}. The function $\hat{h}$ is called the "heuristic function" since it provides the facility for supplying the $\mathbf{A}^*$ algorithm with special information about the particular search problem under consideration to focus the search of $\mathbf{A}^*$. The proper choice of the heuristic function can result in efficient search, i.e., an efficient solution to the CSP. The evaluation function is chosen to be

$$\hat{f}(s_x)=\hat{g}(s_x)+\hat{h}(x) \tag{14}$$

where $x\in X_a$ is the current state considered by the $\mathbf{A}^*$ algorithm. The proper interpretation of $\hat{f}(s_x)$ is that it estimates the cost of a state path from x_0 to $x'\in X_{af}$ that goes through the state x. It is not to be interpreted as being just the cost of s_x.

The A* Algorithm

The $\mathbf{A}^*$ 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 $\mathbf{A}^*$, 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 $s\leftarrow x^1 s$, and go to (a). The operation of finding the set $\mathfrak{E}(x)=\{x': x'\in X_a \text{ and } x'=\delta_a(q,x)\}$ 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'. Next, the $\mathbf{A}^*$ algorithm which produces an optimal allowable state trajectory $s^*\in \mathfrak{X}^*(A,x_0,X_{af})$ assuming that A, such that A[P], is (x_0,X_{af})-reachable is given.

$\mathbf{A}^*$:

(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 $s_{x'}\in \mathfrak{X}^*(A,x_0,X_{af})$, an optimal state trajectory.

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

(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 $O\leftarrow O\cup\{(x',x'')\}$.

The algorithm is nearly the same as that originally given in [10] except that the "pointers" are included explicitly in the algorithm and the lists of "open and closed nodes" are replaced with sets of events O and C that essentially define the pointers.

Theory of the A* Algorithm

$\mathbf{A}^*$ is said to be *complete* since it terminates with a solution. A heuristic function $\hat{h}(x)$ is said to be *admissible* if $0\le\hat{h}(x)\le J(s^*_{x>})$ for all $x\in X_a$ such that $s^*_{x>}\in \mathfrak{X}^*(A,x,X_{af})$. Let $\mathbf{A}^*(\hat{h}(x))$ denote an $\mathbf{A}^*$ algorithm which uses $\hat{h}(x)$ as its heuristic function. If $\hat{h}(x)$ is admissible then $\mathbf{A}^*(\hat{h}(x))$ 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 $\hat{h}_2$ is said to be *more informed than* $\hat{h}_1$ if both are admissible and $\hat{h}_2(x)>\hat{h}_1(x)$ for all $x\in X_a-X_{af}$. If heuristic $\hat{h}_2$ is more informed than $\hat{h}_1$ then $\mathbf{A}^*(\hat{h}_2)$ is said to be *more informed than* $\mathbf{A}^*(\hat{h}_1)$. An algorithm $\mathbf{A}^*(\hat{h}_1)$ is said to *dominate* $\mathbf{A}^*(\hat{h}_2)$ if every state expanded by $\mathbf{A}^*(\hat{h}_1)$ is also expanded by $\mathbf{A}^*(\hat{h}_2)$. If $\mathbf{A}^*(\hat{h}_2)$ is more informed than $\mathbf{A}^*(\hat{h}_1)$, then $\mathbf{A}^*(\hat{h}_2)$ dominates $\mathbf{A}^*(\hat{h}_1)$. For all states $x\in X_a$ expanded by $\mathbf{A}^*$, $\hat{f}(s_x)\le J(s^*)$ for $s_x\in \mathfrak{X}(A,C,O)$ and $s^*\in \mathfrak{X}^*(A,x_0,X_{af})$. Every state $x\in X_a$ such that $(\cdot,x)\in O$ and $\hat{f}(s_x)<J(s^*)$ for $s_x\in \mathfrak{X}(A,C,O)$ and $s^*\in \mathfrak{X}^*(A,x_0,X_{af})$ will be expanded before termination by $\mathbf{A}^*$.

A heuristic function $\hat{h}(x)$ is said to be *monotone* if $\hat{h}(x)\le\chi(x,x')+\hat{h}(x')$ for all $(x,x')\in E(A)$. A heuristic function $\hat{h}(x)$ is said to be *consistent* (equivalent to being monotone) if

$\hat{h}(x) \le J(s^*_{xx'}) + \hat{h}(x')$ 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) $\mathbf{A}^*(\hat{h}(x))$ 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 $s^*_x \in \mathfrak{X}^*(A,x_0,x)$, (b) The $\hat{f}(s_x)$ values for the sequence of states expanded by $\mathbf{A}^*(\hat{h}(x))$ are non-decreasing, and (c) The necessary condition for expanding state x is $\hat{h}(x) \le J(s^*) - J(s^*_x)$ while the sufficient condition is $\hat{h}(x) < J(s^*) - J(s^*_x)$ for $s^*_x \in \mathfrak{X}^*(A,x_0,x)$ and $s^* \in \mathfrak{X}^*(A,x_0,X_{af})$. An algorithm $\mathbf{A}^*(\hat{h}_2)$ is said to *largely dominate* $\mathbf{A}^*(\hat{h}_1)$ if every state expanded by $\mathbf{A}^*(\hat{h}_2)$ is also expanded by $\mathbf{A}^*(\hat{h}_1)$, except perhaps some states $x \in X_a$ for which $\hat{h}_1(x) = \hat{h}_2(x) = J(s^*) - J(s^*_x) = J(s^*_{x>})$. If $\hat{h}_2(x) \ge \hat{h}_1(x)$ for all $x \in X_a$ and $\hat{h}_1(x)$ and $\hat{h}_2(x)$ are monotone, then $\mathbf{A}^*(\hat{h}_2)$ largely dominates $\mathbf{A}^*(\hat{h}_1)$. 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 $\mathbf{A}^*$ algorithm can be simplified by removing Step 5 (iii) since events (pointers) will never be taken from C and placed in O.

One serious problem with $\mathbf{A}^*$, that has limited its applicability, is the requirement that the heuristic function $\hat{h}$ be specified in a special form (admissible and monotone) to guarantee the return of an optimal solution and to obtain efficient search in the worst case for all problems. In general, it is often very difficult, or seemingly impossible to specify $\hat{h}$ so that it possesses the required properties. As we shall discuss in Section 4 there have been many approaches to solve the problem of how to pick the heuristic function, but that each of these has certain deficiencies. We then introduce the idea of specifying the heuristic function using a metric space formulation and show that the approach is applicable to a wide class of DESs.

Computational Complexity of $\mathbf{A}^*$

Following [20] in a worst case analysis of $\mathbf{A}^*$ we assume as a basic operation the expansion of a state. Let $X_e = \{x \in X: \hat{f}(s_x) \le J(s^*)$ for $s_x \in \mathfrak{X}(A,C,O)$ and $s^* \in \mathfrak{X}^*(A,x_0,X_{af})\}$. No more than $|X_e|$ states will be expanded at termination. If the heuristic function is only admissible (and not monotone) then it is possible that $\mathbf{A}^*$ expands $O(2^r)$ (where $r = |X_e|$) states since for each state expanded every other state that is expanded by termination could also be expanded [20]. If the heuristic function is known to be monotone (hence also admissible) then each state is only expanded once so $\mathbf{A}^*$ runs in $O(|X_e|)$ steps [20].

It is also important to note that the computational complexity of $\mathbf{A}^*$ is optimized relative to a certain class of algorithms that are "equally informed" about the plant and return an optimal solution [4]. The class of algorithms considered include only those which use state expansion as their primitive computational step, only expand states that were generated by the algorithm, and that begin the expansion process with the initial state x_0. This excludes, for instance, bidirectional searches. In [4] it is shown that in the case where it is known that the heuristic function for $\mathbf{A}^*$ is monotone, then $\mathbf{A}^*$ uses the most effective scheme of any admissible algorithm for utilizing the heuristic information provided by the heuristic function. In other words, any other admissible algorithm must expand at least as many states as $\mathbf{A}^*$ by termination. It must be stressed however that for particular problems one may be able to construct an admissible algorithm that will expand fewer states than $\mathbf{A}^*$. But, no equally informed admissible algorithm will expand fewer states than $\mathbf{A}^*$ for all problems that can be described with the plant model P. This sort of optimality with respect to computational complexity is significant in light of the results in the next Section where is is shown how to pick a heuristic function that is both admissible and monotone for a wide variety of applications.

4. THE HEURISTIC FUNCTION

There has been extensive work on the problem of how to automatically generate heuristics for an arbitrary problem. In [5], [8,9], and [32] the authors respectively introduced the related "problem similarity", "auxiliary problem", and "relaxed model" approaches to the generation of heuristics. The main deficiencies of these approaches is that there is no way to systematically produce similar and auxiliary problems or relaxed models. Furthermore, in [37] it was proven that the approach in [9] can be computationally inefficient. Approaches similar to these have also been used in operations research. See, for instance, [18] or [12].

As an extension to Pearl's (and the others) work the authors in [15] suggest a method for modelling a problem (plant) P that will always lead to the derivation of a set of "simplified" problems, say P_i, from which admissible and monotone heuristics can be derived algorithmically for the original plant P. Their algorithm uses a problem decomposition algorithm to obtain the problems P_i and then uses exhaustive search to find the minimal cost optimal path in each P_i. From this the heuristic which is admissible and monotone is generated. The problem with this approach is the reliance on an exhaustive search. While the authors have found computationally efficient solutions to several specific simple problems, the approach of decomposing the problem to generate heuristics was not proven to be computationally efficient in general.

All of the above approaches are in some ways related. They rely on using another model to specify the heuristics for the problem (plant) at hand. Their subsequent disadvantages are correspondingly similar. The problem of finding a heuristic function to reduce the complexity of search is solved with another mechanical procedure (either a search method such as backtracking or an exhaustive search) that is, in general, computationally inefficient. The above results have, however, been successfully applied to a variety of problems and have offered much insight into the nature of heuristic information and how it should be used in a problem solving process. In our metric space approach to specifying the heuristic function there is no need to perform a search or use a mechanical procedure to find the heuristic. In this way we do not defeat the main purpose of using the $\mathbf{A}^*$ algorithm - to reduce the computational complexity of search.

The Specification of Heuristic Functions: A Metric Space Approach

In this Section the concern is with whether given a plant P, and allowable behavior A, such that A[P], an appropriate admissible and monotone heuristic function $\hat{h}(x)$ can be specified for all $x \in X_a$. The approach taken here relies on the use of metric spaces. Let Z be an arbitrary non-empty set and let $\rho: Z \times Z \to \mathbb{R}$ where ρ has the following properties: (i) $\rho(x,y) \ge 0$ 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) $\rho(x,y) \le \rho(x,z) + \rho(z,y)$ 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.

The first theorem says that if the heuristic function is chosen to be the distance between a point x and a set X_{af} as defined in a metric space and the metric satisfies a certain constraint then it will be both admissible and monotone.

Theorem 1: 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 $\rho(x,x') \le \chi(x,x')$ for all $(x,x') \in E(A)$ then $\hat{h}(x)$ is admissible and monotone.

Proof: For admissibility let $s_{\bar{x}x''} \in \mathfrak{X}(A,\bar{x},X_{af})$ where $\bar{x} \in X_a$ and let $xx' \in s_{\bar{x}x''}$ be two successive states on $s_{\bar{x}x''}$. From the triangle inequality, $\rho(x,x'') \le \rho(x,x') + \rho(x',x'')$. Using repeated applications of the triangle inequality along $s_{\bar{x}x''}$ we know that

$$\rho(\bar{x},x'') \le \sum_{(x,x') \in E_t(A)} \rho(x,x') \qquad (15)$$

and with the assumption that $\rho(x,x') \le \chi(x,x')$ for all $(x,x') \in E(A)$

$$\sum_{(x,x') \in E_t(A)} \rho(x,x') \le \sum_{(x,x') \in E_t(A)} \chi(x,x') \qquad (16)$$

where $t = s_{\bar{x}x''}$. Since this is true for any state path it is true for optimal ones also. Let $s^*_{\bar{x}x''} \in \mathfrak{X}^*(A,\bar{x},X_{af})$ (we need only consider cases where one exists). Then,

$$0 \le \rho(\bar{x},x'') \le \sum_{(x,x') \in E_t(A)} \chi(x,x') = J(s^*_{\bar{x}x''}) \qquad (17)$$

where $t = s^*_{\bar{x}x''}$. So, by the definition of $\hat{h}(x)$ we have $0 \le \hat{h}(\bar{x}) \le J(s^*_{\bar{x}x''})$ for all $\bar{x} \in X_a$ and $s^*_{\bar{x}x''} \in \mathfrak{X}^*(A,\bar{x},X_{af})$ which guarantees the admissibility of $\hat{h}(x)$. For 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 $\rho(x,x') \le \chi(x,x')$, $\hat{h}(x) \le \chi(x,x') + \hat{h}(x')$ for all $x,x' \in X_a$ such that $xx' \in s$ where $s \in \mathfrak{L}(A, \bar{x}, A_{af})$ which guarantees the monotonicity of $\hat{h}(x)$. ■

This Theorem was first proven in [29] for the case where $|X_{af}|=1$ and later in [30] for the case of multiple final states. Note that in the proof of Theorem 1 we could have just chosen to prove that the heuristic function was monotone because this automatically implies that the admissibility condition is true. Theorem 1 says that if the standard metric space notion of distance is used for the heuristic function (with constraints on the metric), then we automatically get an admissible and monotone heuristic function. The use of the metric space definition of distance shows how the $\mathbf{A}^*$ algorithm can be seeking one final state as it expands states along some path then switch to seek a different final state even though it still expands states along the same path. This helps to clarify the operation of $\mathbf{A}^*$. Theorem 1 also gives support to the discussions in [32,33] on why heuristic functions that have been proven to be admissible are normally monotone also. Roughly speaking, it supports this idea since we often think of distance (when we devise heuristics) in the same way as is done in metric spaces. It is, however, not necessary to use this metric space notion of distance as Theorem 2 shows. Let $\theta: X_a \times X_a \to \mathbb{R}_+$ (θ not necessarily a metric) and suppose that $\theta(x,x') \le \chi(x,x')$ for all $(x,x') \in E(A)$.

Theorem 2: For the DES P and A[P] there exist heuristic functions
(a) $\hat{h}(x)=\inf\{\theta(x,x'): x' \in X_{af}\}$ such that $\theta \notin \Delta(X_a)$, or
(b) $\hat{h}(x) \le \inf\{\theta(x,x'): x' \in X_{af}\}$ such that $\theta \in \Delta(X_a)$
that are admissible and in some cases monotone.

Proof: For case (a) suppose that $\theta(x,x')=0$ for all $x,x' \in X_a$. Then $\theta \notin \Delta(X_a)$ but when θ is used in the heuristic function we have $\hat{h}(x)=0$ for all $x \in X_a$ which is clearly an admissible and monotone heuristic function. (Notice that if $\theta(x,x')=0$ just for $(x,x') \in E(A)$ then it is not necessarily the case that $\hat{h}(x)$ satisfy the monotone condition. Also, if $\hat{h}(x)$ is monotone then it is not necessarily the case that θ satisfy the triangle inequality.) For case (b) $\hat{h}(x) \le \hat{h}'(x)$ for all $x \in X_a$, where $\hat{h}'(x)$ satisfies the conditions of Theorem 4, so by the definition of admissibility $\hat{h}(x)$ is admissible (not necessarily monotone). ■

Theorems 1 and 2 place the statements made in the theory of heuristic search about "distance" between points and between points and sets in a precise mathematical setting. It also clarifies the relationship between monotonicity and the triangle inequality which has only, in the past, been loosely referred to (See [10,33]).

The Specification of Good Heuristic Functions

Consider the DES model $P'=(X,Q,\delta,\chi',x_0,X_f)$ defined as in (1) except $\chi': X \times X \to \mathbb{R}_+$ where $\chi' \in \Delta(X)$, i.e. the costs for the events are characterized by a metric. Also, in terms of the metric space $\{X,\chi'\}$ every $x \in X$ is assumed to be an isolated point. The allowable behavior $A'=(X_a,Q_a,\delta_a,\chi'_a,x_{a0},X_{af})$ such that A'[P'] as in (5). A heuristic function is said to be *good* if $\hat{h}(x)=\inf\{\chi'(x,x_f): x_f \in X_{af}\}$ for all $x \in X_a$ where $\chi' \in \Delta(X_a)$. Considering Theorems 1 and 2, the motivation for this definition lies in the desire to choose $\hat{h}(x)$ as large as possible to get efficient search.

Theorem 3: For the DES P' and A'[P'] if $\hat{h}(x)$ is good then $\hat{h}(x)$ is admissible and monotone.

Proof: Since every $x \in X$ is an isolated point there exists a $\delta'>0$ such that $\chi'(x,y) \ge \delta'$ for every $x,y \in X$ such that $x \ne y$. Since $\mathbf{A}^*$ prunes cycles it will not repeatedly investigate any single $(x,x') \in E(A')$ with $x=x'$ and $\chi'(x,x')=0$; hence $\mathbf{A}^*$ is complete. By Theorem 1 $\hat{h}(x)$ is admissible and monotone. ■

This indicates that if we have a plant P' without costs or a plant where it is not known how to specify the costs then Theorem 3 offers a method to assign the costs so that an efficient search may be possible. More importantly it illustrates how information from the plant (the knowledge that the costs were modelled with a metric) is used to focus the $\mathbf{A}^*$'s search for an optimal solution. This is further quantified by showing that if a good heuristic function $\hat{h}(x)$ is used we can expect $\mathbf{A}^*(\hat{h}(x))$ to more narrowly focus its search.

Theorem 4: For the DES P' and A'[P'] if $\hat{h}(x)=\inf\{\chi'(x,x_f): x_f \in X_{af}\}$ for all $x \in X_a$ with $\chi' \in \Delta(X_a)$ then $|\hat{h}(x)-\hat{h}(x')| \le \chi'(x,x')$ for all $(x,x') \in E(A')$.

Proof: From monotonicity $\hat{h}(x) \le \chi'(x,x') + \hat{h}(x')$ for all $(x,x') \in E(A')$. Also, with a simple rearrangement, $-\chi'(x',x) \le \hat{h}(x) - \hat{h}(x') \le \chi'(x,x')$. Since χ' is a metric, $\chi'(x,x')=\chi'(x',x)$ for all $x,x' \in X_a$ so we have $|\hat{h}(x) - \hat{h}(x')| \le \chi'(x,x')$ for all $(x,x') \in E(A')$. ■

If the heuristic function is monotone then the estimate of the remaining cost at the next state cannot be too much *smaller* than the estimate of the remaining cost at the current state. This tends to guarantee that we have good heuristic information (large $\hat{h}(x)$) so fewer states will be expanded. If χ is a metric which specifies the costs for the events and is used to guide the search then it is also the case that the estimate of the remaining cost at the next state cannot be too much *larger* than the estimate of the remaining cost at the current state. This tends to guarantee that $\mathbf{A}^*$ will not get side-tracked too much from finding an optimal solution. We see that when the heuristic function is based on a metric that is used to specify the costs of the events for the plant P' then enough information from the plant is used so that we are guaranteed to get an admissible and monotone heuristic function and we are guaranteed that the change in the estimate of the remaining cost is bounded by costs of events in P'.

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 [34] (e.g. General or Extended Petri nets) where the states are n-tuples of natural numbers or the more recent work in [19] where numerical n-tuples are used. It is easy to specify a wide variety of metrics on $\mathbb{R}^n$ [21]; hence most often there is no problem in finding a heuristic function for P. In addition, if one of these metrics can be used to model the costs of the events then the model P' can be used and the benefits of a focused search can be obtained. It is interesting to note that if the typical metrics on $\mathbb{R}^n$ are used for a particular application, the designer is still given some flexibility in that certain parameters in the metric itself can be chosen. This is important since it gives the designer another way to "load heuristic information into $\hat{h}(x)$" so that the search can proceed in a more efficient manner.

5. EXAMPLES

In this section we briefly outline three examples which illustrate some of the above results. First, consider the 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 $\mathbf{A}^*(\hat{h}(x))$ where $\hat{h}(x)$ satisfies the conditions of Theorem 1 will result in an optimal solution s^* to the CSP. The sequence of

inputs generating s^* will utilize a minimum input/event cost of all possible input sequences. A particular application that may benefit from this approach is a job shop where one desires to minimize the use of input resources and the cost of operating machines to complete a job.

To achieve minimum-time control of deterministic DESs suppose that the DES model P operates in a synchronous fashion relative to a clock with an interval between ticks of $T \in \mathbb{R}^+$. Let $\chi(x,x')=T$ for all $(x,x') \in E(P)$. Suppose that some allowable DES behavior A such that A[P] is specified. $A^*(\hat{h}(x))$ where $\hat{h}(x)$ satisfies the conditions of Theorem 1 will result in an optimal solution s^* to the CSP. The sequence of inputs generating s^* will result in the state of P to be transferred from x_0 to $x_f \in X_{af}$ in minimum time. A particular application may be a synchronous manufacturing system where one desires to find the sequence of processing steps that will minimize the total processing time.

Recently, relationships between artificial intelligence (AI) planning systems and control systems have been identified [29] and certain AI planning systems have been shown to be DESs [27,28]. In these papers some of the above techniques have been applied to simple AI planning problems such as the blocks world, N-puzzle, triangle and peg, and missionaries and cannibals problems. These examples serve to illustrate that certain AI systems where there is an inherent feedback are amenable to analytical study with DES theoretic techniques. This provides a new application area for the control community.

Acknowledgment: The authors gratefully acknowledge the partial support of the Jet Propulsion Laboratory and an Arthur J. Schmitt Fellowship.

References

[1] Aho A.V., Hopcroft J.E., Ullman J.D., The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass., 1975.

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

[3] Bertsekas D.P., Tsitsiklis J.N., Parallel and Distributed Computation: Numerical Methods, Prentice Hall, NJ, 1989.

[4] Dechter R., Pearl J., "Generalized Best-First Search Strategies and the Optimality of A*", Journal of the ACM, Vol. 32, No. 3, pp. 505-536, July 1985.

[5] Gaschnig J., "A Problem Similarity Approach to Devising Heuristics", Proc. 6th IJCAI, pp. 301-307, Aug. 20-23, Tokyo 1979.

[6] Gelperin D., "On the Optimality of A*", Artificial Intelligence, Vol. 8, pp. 69-76, 1977.

[7] Gondran M., Minoux M., Graphs and Algorithms, Wiley, NY, 1984.

[8] Guida G., Somalvico M., "Semantics in Problem Representation and Search", Inf. Proc. Letters, Vol. 5, No. 5, pp. 141-145, 1976.

[9] Guida G., Somalvico M., "A Method for Computing Heuristics in Problem Solving", Information Sciences, Vol. 19, pp. 251-259, 1979.

[10] 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.

[11] Hart P.E., Nilsson N.J., Raphael B.,"Correction to: A Formal Basis for the Heuristic Determination of Minimum Cost Paths", SIGART Newsletter Vol. 37, pp. 28-29, 1972.

[12] Held M., Karp R.M., "The Traveling Salesman Problem and Minimum Spanning Trees", Operations Research, Vol. 18, pp. 1138-1162, 1970.

[13] Special Issue on Dynamics of Discrete Event Systems, Proceedings of the IEEE, Vol. 77, No. 1, Jan. 1989.

[14] Ibaraki T., "Branch and Bound Procedure and State-Space Representation of Combinatorial Optimization Problems", Information and Control, Vol. 36, No. 1, pp. 1-27, Jan. 1978.

[15] Irani K.B., Yoo S.I., "A Methodology for Solving Problems: Problem Modelling and Heuristic Generation", IEEE Trans. on Pattern Anal. and Mach. Int., Vol. 10, No. 5, Sept. 1988.

[16] Kalman R.E., Falb P.L., Arbib M.A., Topics in Mathematical System Theory, McGraw-Hill, NY, 1969.

[17] Karp R.M., Held M., "Finite-State Processes and Dynamic Programming", SIAM J. Applied Mathematics, Vol. 15, No. 3, pp. 693-718, May 1967.

[18] Lawler E.L., Wood D.E., "Branch and Bound Methods: A Survey", Operations Research, Vol. 14, No. 4, pp. 699-719, July-Aug. 1966.

[19] 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.

[20] Martelli A., "On the Search Complexity of Admissible Search Algorithms", Artificial Intelligence, Vol. 8, pp. 1-13, 1977.

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

[22] Morin T.L., Marsten T.L., "Branch and Bound Strategies for Dynamic Programming", Operations Res., Vol. 24, No. 4, pp. 611-627, July-Aug. 1976.

[23] Nau D.S., Kumar V., Kanal L., "General Branch and Bound and Its Relation to A* and AO*", Artificial Intelligence, Vol. 23, pp. 29-58, 1984.

[24] Nilsson N.J., Problem-Solving Methods in Artificial Intelligence, McGraw-Hill, NY, 1971.

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

[26] Ozveren C.M., Willsky A.S., Antsaklis P.J., "Stability and Stabilizability of Discrete Event Dynamic Systems", MIT, LIDS Report LIDS-P-1853, Feb. 1989.

[27] Passino K.M., Antsaklis P.J., "Artificial Intelligence Planning Problems in a Petri Net Framework", Proc. of the American Control Conf., pp. 626-631, Atlanta GA, June 1988.

[28] Passino K.M., Antsaklis P.J., "Planning Via Heuristic Search in a Petri Net Framework", Proc. of the Third IEEE Int. Symp. on Intelligent Control, Arlington VA, August 1988.

[29] Passino K.M., Antsaklis P.J., "A System and Control Theoretic Perspective on Artificial Intelligence Planning Systems", Applied Artificial Intelligence, Vol. 3, No. 1, pp. 1-32, 1989.

[30] Passino K.M., Antsaklis P.J., "An Approach to Controller Synthesis for Discrete Event Systems", To appear in the Proc. of the Allerton Conf. on Communication, Control, and Computing, Sept. 1989.

[31] 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.

[32] Pearl J., "On the Discovery and Generation of Certain Heuristics", The AI Magazine, Vol. 4, No. 1, pp. 23-33, 1983.

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

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

[35] 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.

[36] Tsitsiklis J.N., "On the Control of Discrete-Event Dynamical Systems", Math. of Control, Signals, and Systems, Vol. 2, No. 2, pp. 95-107, 1989.

[37] Valtorta M., "A Result on the Computational Complexity of Heuristic Estimates for the A* Algorithm", Information Sciences, Vol. 34, pp. 47-59, 1984.

[38] 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.