K. M. Passino and P. J. Antsaklis, "Solutions to Optimal Control Problems for Discrete Event Systems,"
Control Systems Technical Report #70, Dept. of Electrical Engineering, Univ. of Notre Dame, July 1990.

Control Systems Tehnical Report #70

Solutions to Optimal Control Problems for Discrete Event Systems*

July 1990

Kevin M. Passino and Panos J. Antsaklis
Dept. of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, IN 46556

*Submitted for Journal Publication.

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

K. M. Passino and P. J. Antsaklis, "Solutions to Optimal Control Problems for Discrete Event Systems," Control Systems Technical Report #70, Dept. of Electrical Engineering, Univ. of Notre Dame, July 1990.

SOLUTIONS TO OPTIMAL CONTROL PROBLEMS FOR DISCRETE EVENT SYSTEMS

Kevin M. Passino[1] and Panos J. Antsaklis[2]

Abstract

We introduce a class of not necessarily finite state "logical" discrete event system (DES) models which can also model the *costs* for events to occur. Let P and A denote two such models. Suppose that P characterizes the *valid* behavior of a dynamical system and A represents certain design objectives which specify the *allowable DES behavior* which is *contained in* the valid behavior. An optimal control problem for P and A is how to choose the sequence of inputs to P so that the DES behavior lies in A (i.e., it is allowable) and so that a performance index defined in terms of the costs of the events is minimized. Our solution to this type of optimal control problem utilizes results from the theory of heuristic search to help overcome problems with computational complexity often encountered with logical DES models. To ensure a reduction in computational complexity over conventional techniques one must be able to specify an *admissible* and *monotone* "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 such heuristic functions for many 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 admissible and monotone heuristic functions in a systematic way for several optimal control problems for a wide variety of DES applications. The results are applied to an automated factory, an optimal parts distribution problem in flexible manufacturing systems, and artificial intelligence planning problems.

Index Terms: Discrete Event Systems, Optimal Control, Search Algorithms

1. INTRODUCTION

We introduce a class of not necessarily finite state (or input) "logical" discrete event system (DES) models which can also model the *costs* for events to occur. Let P and A denote two such models. Suppose that P characterizes the *valid* behavior of a dynamical system, some of which may exhibit *undesirable* properties. Furthermore, suppose that A represents certain design objectives which specify the *allowable DES behavior*. The allowable DES behavior A is specified so that it is *contained in* the valid DES behavior P and it normally characterizes the acceptable DES behavior, i.e., the DES behavior with *desirable* properties. As it is explained in Section 2, this use of allowable DES behavior contained in valid DES behavior is similar to the minimal acceptable and legal languages for a plant used for the "supervisor synthesis problems" in the language-theoretic Ramadge-Wonham framework [36]. An optimal control problem for P and A is how to choose the sequence of inputs to P so that: (i) the DES behavior lies in A (i.e., it is allowable) and (ii) we minimize a performance index defined in terms of the costs for events to occur. In this paper we focus on the formulation of several such optimal control problems and on developing computationally efficient solutions to them. Our results establish the first steps towards developing the foundations for an optimal control theory for logical DES.

1. Please address all correspondence to Kevin Passino at the Dept. of Electrical Eng., The Ohio State University, 2015 Neil Ave., Columbus, OH 43210.
2. Panos Antsaklis is with the Dept. of Electrical Eng., University of Notre Dame, Notre Dame, IN 46556.

As it is explained in Section 3, problems with computational complexity for logical DES models prohibit the use of a conventional dynamic programming solution to the above optimal control problem. The standard shortest path algorithms (e.g., Dijkstra's, Moore's, Ford's, and Bellman's [6]) cannot be used to solve the above optimal control problem due to the fact that we search from a state to a set of states on an *implicit graph* that is possibly infinite. It is for these reasons that we utilize a *branch and bound* algorithm called the "$\mathbf{A}^*$ algorithm" [9] that can use certain information about the plant (to be defined precisely in Section 3) to focus the search for a solution to the optimal control problem and hence reduce computational complexity. Note that it is possible to solve the optimal control problem via a generalized version of Dijkstra's algorithm but this generalized Dijkstra's algorithm is a special case of $\mathbf{A}^*$ [6]. Moreover, in Proposition 2 it is shown that in solving the optimal control problem, if $\mathbf{A}^*$ operates with an *admissible* and *monotone heuristic function* it will always visit fewer states than the generalized Dijkstra's algorithm. Also, via Remarks 1-3 we introduce several other optimal and *near-optimal* control problems that can be solved efficiently via our approach provided one can specify a monotone heuristic function. The main conclusions from Section 3 are that $\mathbf{A}^*$ can solve the optimal control problems considered in this paper and it can solve them efficiently provided that an admissible and monotone heuristic function can be specified.

The problem one encounters though is that it is, in general, quite difficult to find an admissible and monotone heuristic function for many applications. In Section 4 we extend the theory of heuristic search by showing that a metric space approach can be used to specify admissible and monotone heuristic functions in a systematic way for various optimal control problems for a wide variety of DES so that they can be solved efficiently. Specifically, the main results of the paper are as follows: Theorem 1 provides a metric space approach to specifying admissible and monotone heuristic functions. Theorem 2 shows that it is not necessary to use a metric to specify the heuristic function. In Theorem 3 and Remark 4 we show how to *automatically* specify admissible and monotone heuristic functions for a wide class of DES that can be modelled via a set of states X such that $X \subset \mathbb{R}^n$ (e.g. Extended Petri nets [35], Vector DES [18], and other Petri net models [16,12,38]). In cases where it is known that the costs of the events can be specified with a metric and all the states are isolated points we can expect computational complexity to be further reduced. We introduce a new class of "good" heuristic functions and show in Theorem 4 that these are admissible and monotone. Theorem 6 quantifies how good heuristic functions can be expected to focus the search for solutions to optimal control problems. The theoretical results in this paper are based upon and an extension of those in [26-28,30].

In Section 5 we apply the method in Sections 3 and results in Section 4 to three DES applications: (i) an automated factory, (ii) an optimal part distribution problem in flexible manufacturing systems, and (iii) artificial intelligence (AI) planning problems. In each case we show how the results of Section 4 can be used to specify admissible and monotone heuristic functions; then we use these in A^* to solve optimal control problems for each of the three applications. The results clearly illustrate that by using our approach to specify the heuristic functions for A^* significant computational savings can be obtained over the generalized Dijkstra's algorithm for solving optimal control problems. Section 6 contains the concluding remarks.

2. AN OPTIMAL CONTROL PROBLEM FOR DISCRETE EVENT SYSTEMS

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

$\mathbb{R}^+$ denotes the set of positive reals and $\mathbb{R}_+=\mathbb{R}^+\cup\{0\}$. The model P is limited to representing DES 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 asynchronous DES 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 the DES 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 $(x,x')\in E(P)$. (For convenience however we define $\chi(x_d,x_0)=0$.) Finally, we require that for each $x\in X$, $|\{\delta(q,x):q\in Q\}|$ is finite, i.e., that the graph of P is *locally finite*.

The addition of the cost for an event to occur is a new addition to the standard automaton model P for DES. It allows for the development of a theory on the optimal control of logical DES and the analysis of an important class of DES 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 mathematical notation in this paper is as follows: 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'···z" and t=yy'···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'···t···z". For brevity, the notation $s_{zz"}$ is used to denote a string $s \in Z^*$ such that s=zz'···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"··· begins with $z \in Z$ and the end element is not specified. The string $s_{<z>}$ denotes the string $s \in Z^*$ such that $s=z_0z'\cdots zz"\cdots$, i.e., a string that begins at z_0, passes through z, and whose end element is not specified. A (finite directed) *cycle* is a string $s \in Z^*$ such that s=zz'···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'"···, $E_S(P)$ 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 $X_z \subset X$ then

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

denotes the set of all finite state trajectories s=xx'···x" of P beginning with $x \in X$ and ending with $x" \in X_z$. Then, for instance, $\mathfrak{L}(P,x_0,X_f)$ 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 $x \in X_f$ and $\mathfrak{L}(P,x,X)$ denotes the set of all valid state trajectories for P that begin with $x \in X$. A DES P is said to be *(x,X_z)-reachable* if there exists a sequence of inputs $u \in Q^*$ that produces an state trajectory $s \in \mathfrak{L}(P,x,X_z)$.

Intuitively, 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 $\mathfrak{L}(P,x,X)$ 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) $X_a \subset X$, $Q_a \subset Q$,

(ii) $\delta_a: Q_a \times X_a \rightarrow X_a$, $\delta_a(q,x)=\delta(q,x)$ if $\delta(q,x) \in X_a$ and $\delta_a(q,x)$ is undefined otherwise

(iii) $\chi_a: X_a \times X_a \rightarrow \mathbb{R}^+$ is a restriction of $\chi: X \times X \rightarrow \mathbb{R}^+$,

(iv) $x_{a0}=x_0$, $X_{af} \subset X_f$.

Also, let $E(A) \subset E(P)$ denote the set of *allowable events* defined as in (2), χ_a is defined for all $(x,x') \in E(A)$, and $E_s(A)$ is defined as in (3) above. In a graph-theoretic sense, A is a *partial subgraph* of P. 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 to that used for the "supervisor synthesis problems" in the language-theoretic Ramadge-Wonham framework [36]. Ramadge and Wonham introduced a minimal acceptable language and legal language and study the synthesis of supervisors so that the resulting language controlled by the supervisor in the plant lies between the acceptable and legal languages. Their minimal acceptable and legal languages specify what we call the allowable behavior A which is contained in P.

To specify optimal control problems let the *performance index* be

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

where the cost of a state trajectory s is defined by

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

for all $x \in X_a$ and $s \in \mathfrak{L}(A,x,X_a)$. By definition, $J(s)=0$ if $s=x$ where $x \in X_a$. Let A describe the allowable behavior for a plant P such that A[P] then we have:

Optimal Control Problem (OCP): Assume that A is (x_0,X_{af})-reachable. Find a sequence of inputs $u \in Q_a^*$ that drives the system A along an *optimal state trajectory* s^*, i.e., $s^* \in \mathfrak{L}(A,x_0,X_{af})$ such that $J(s^*)=\inf\{J(s): s \in \mathfrak{L}(A,x_0,X_{af})\}$.

Since we require that $u \in Q_a^*$ and $s \in X_a^*$, the solutions to the OCP will achieve not only optimal but also allowable DES behavior. There may, in general, be more than one optimal state trajectory, i.e., the solution to the OCP is not necessarily unique. The set of optimal

state trajectories for A, beginning at state $x \in X_a$, and ending at state $x' \in X_z$, where $X_z \subset X_a$, is denoted by

$$\mathfrak{T}^*(A,x,X_z) \subset \mathfrak{T}(A,x,X_z). \tag{8}$$

In this paper we are concerned with finding only one optimal state trajectory for the OCP and finding it in a computationally efficient manner. We shall also show how other optimal and *near-optimal* control problems can also be studied in our framework.

3. SOLUTIONS TO OPTIMAL CONTROL PROBLEMS VIA HEURISTIC SEARCH

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 optimal control problems. Here we use an approach which seeks to minimize the number of state trajectories considered and hence produces a solution in a computationally efficient manner for a wide variety of DES.

A conventional dynamic programming solution could be used for the OCP but due to the problem of state space explosion often found using logical DES models such methods can result in an inefficient algorithm with large memory requirements [1,40]. Often, a branch and bound technique is chosen in such situations to produce either optimal or near-optimal solutions (See, for instance, [6,15,17,22]). This is the approach taken here. We use a particular class of branch and bound algorithms called "heuristic search" algorithms [13,23] which utilize the "principle of optimality" of dynamic programming 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 "A^* algorithm" and it was introduced in [9,10,4]. The formal properties of A^* are given in [24,25,34] and are briefly summarized below to provide the necessary background for this paper.

Note that: (i) $|X|$ (and $|X_a|$) can be infinite, (ii) the graph of P (and A) is defined *implicitly* (via P and A) rather than explicitly, and (iii) we search for the shortest state path from one state to a <u>set</u> of states. Hence, Dijkstra's algorithm [6] cannot, in general, be used to solve the OCP. It is for similar reasons that Moore's, Ford's, and Bellman's algorithms [6] cannot be used to solve the OCP. Dijkstra's algorithm can be generalized so that it can also operate even when (i)-(iii) hold; this "generalized Dijkstra's algorithm" is actually a special case of the A^* [6] which will be used here. In fact, below we will show that the worst case computational complexity A^* is always less than or equal to that of the generalized Dijkstra's algorithm and that for a wide class of DES we can significantly

reduce the amount of computations taken to solve the OCP compared with the generalized Dijkstra's algorithm.

Theory of the A* Algorithm

Heuristic search techniques have been applied to problems where computational complexity of search problems is either very high or intractable. 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. To do this, $J(s^*)$ is estimated by some easily computable *evaluation function* given by

$$\hat{f}:X_a^* \rightarrow \mathbb{R}_+ \tag{9}$$

which is defined for all $s \in X_a^*$ such that $s \in \mathfrak{L}(A,x,X_a)$ where $x \in X_a$ (often "$\hat{f}(x)$" is used [34] but we use the mathematically correct notation "$\hat{f}(s_x)$"). If $s^* \in \mathfrak{L}^*(A,x_0,X_{af})$ and $s^*=s_{<x^*>}$ then $J(s^*)=J(s_x^*)+J(s_{xx'}^*)$ where $J(s_x^*)=\min\{J(s_x):s_x \in \mathfrak{L}(A,x_0,\{x\})\}$ and $J(s_{xx'}^*)=\min\{J(s_{xx'}): s_{xx'} \in \mathfrak{L}(A,x,X_{af})\}$. The evaluation function $\hat{f}$ is obtained by approximating both $J(s_x^*)$ and $J(s_{xx'}^*)$ with appropriately defined functions. The value of $J(s_x^*)$ will be estimated using

$$\hat{g}:X_a^* \rightarrow \mathbb{R}_+ \tag{10}$$

where $\hat{g}(s_x)=J(s_x)$ for all $s_x \in \mathfrak{L}(A,x_0,X_a)$ ("$\hat{g}(x)$" is often used in the literature). Note that $\hat{g}(s_x)=0$ if $s_x=x_0$ the initial state of A. To estimate $J(s_{xx'}^*)$ the function

$$\hat{h}:X_a \rightarrow \mathbb{R}_+ \tag{11}$$

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 evaluation function is chosen to be

$$\hat{f}(s_x)=\hat{g}(s_x)+\hat{h}(x) \tag{12}$$

where $x \in X_a$. The function $\hat{f}(s_x)$ estimates the cost of a state path from x_0 to $x' \in X_{af}$ that goes through the state x.

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 operation of finding the set $\mathcal{E}(x)=\{x' \in X_a: x'=\delta_a(q,x)\}$ 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'. The A* algorithm which produces an optimal state trajectory $s^* \in \mathfrak{L}^*(A,x_0,X_{af})$ assuming that A, such that A[P], is (x_0,X_{af})-reachable is given by:

The A^* Algorithm:

(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 $\hat{f}(s_{x'})$ 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 $s_{x'}\in \mathfrak{L}^*(A,x_0,X_{af})$, an 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 $O\leftarrow O\cup\{(x',x'')\}$.
(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 $O\leftarrow O\cup\{(x',x'')\}$.

(6) Go to step (2).

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 $\mathfrak{L}(A,C,O)$. Each state path $s_{x'}\in \mathfrak{L}(A,C,O)$ 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 $s_{x'}\in \mathfrak{L}(A,C,O)$ 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 A^* algorithm above is nearly the same as that originally given in [9] except for clarity the "pointers" (events) are included explicitly in the algorithm via O and C.

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 $s^*_{x>}\in \mathfrak{L}^*(A,x,X_{af})$. 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 $\hat{h}_2$ 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 $A^*(\hat{h}_2)$ is said to be *more informed than* $A^*(\hat{h}_1)$. An algorithm $A^*(\hat{h}_1)$ is said to *dominate* $A^*(\hat{h}_2)$ if every state expanded by $A^*(\hat{h}_1)$ is also expanded by $A^*(\hat{h}_2)$. If $A^*(\hat{h}_2)$ is more informed than $A^*(\hat{h}_1)$, then $A^*(\hat{h}_2)$ dominates $A^*(\hat{h}_1)$. For all states $x\in X_a$ expanded by A^*, $\hat{f}(s_x)\leq J(s^*)$ for $s_x\in \mathfrak{L}(A,C,O)$ and $s^*\in \mathfrak{L}^*(A,x_0,X_{af})$. Every state $x\in X_a$ such that $(\cdot,x)\in O$ and $\hat{f}(s_x)<J(s^*)$ for $s_x\in \mathfrak{L}(A,C,O)$ and $s^*\in \mathfrak{L}^*(A,x_0,X_{af})$ will be expanded before termination by A^*.

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 $s^*_{xx'} \in \mathfrak{L}^*(A,x,x')$. If $\hat{h}(x)$ is a monotone heuristic function then: (i) $A^*(\hat{h}(x))$ finds optimal paths to all expanded states, i.e., $\hat{g}(s_x) = J(s^*_x)$ for all $x \in X_a$ with $(\cdot,x) \in C$, $s_x \in \mathfrak{L}(A,C,O)$, and $s^*_x \in \mathfrak{L}^*(A,x_0,x)$, (ii) The $\hat{f}(s_x)$ values for the sequence of states expanded by $A^*(\hat{h}(x))$ are non-decreasing, and (iii) The necessary condition for expanding state x is $\hat{h}(x) \leq J(s^*) - J(s^*_x)$ while the sufficient condition is $\hat{h}(x) < J(s^*) - J(s^*_x)$ for $s^*_x \in \mathfrak{L}^*(A,x_0,x)$ and $s^* \in \mathfrak{L}^*(A,x_0,X_{af})$. An algorithm $A^*(\hat{h}_2)$ is said to *largely dominate* $A^*(\hat{h}_1)$ if every state expanded by $A^*(\hat{h}_2)$ is also expanded by $A^*(\hat{h}_1)$, 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 $\hat{h}_2(x) \geq \hat{h}_1(x)$ for all $x \in X_a$ and $\hat{h}_1(x)$ and $\hat{h}_2(x)$ are monotone, then $A^*(\hat{h}_2)$ largely dominates $A^*(\hat{h}_1)$. 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 A^* algorithm can be simplified by removing Step 5, part (iii) since events (pointers) will never be taken from C and placed in O. Finally, note that if $|X|$ is finite we can allow $\chi(x,x') = 0$ for any $(x,x') \in E(A)$ and all of the theory listed above for the A^* algorithm is still valid. All of the above properties were proven in the DES-theoretic framework established here but the proofs are omitted in the interest of space and the reader is referred to [34] for the main ideas.

Efficient Solutions to Optimal Control Problems for DES

In this Section we show that our adapted A^* algorithm produces efficient solutions to the OCP and other optimal control problems. The following Proposition follows immediately from the above discussion.

Proposition 1: If $\hat{h}(x)$ is admissible then $A^*(\hat{h}(x))$ provides a solution to the OCP.

For a worst case analysis of the complexity of A^* used to solve the OCP in [20] the authors assumes as a basic operation the expansion of a state and that $\hat{f}(s_x)$ is easy to compute. Let $X_e = \{x \in X : \hat{f}(s_x) \leq J(s^*), s_x \in \mathfrak{L}(A,C,O), s^* \in \mathfrak{L}^*(A,x_0,X_{af})\}$. No more than $|X_e|$ states, where $|X_e| \leq |X|$, will be expanded at termination. If $\hat{h}(x)$ is only admissible (and not monotone) then it is possible that A^* expands $O(2^r)$ (where $r = |X_e|$) states in the worst case since for each state expanded every other state that is expanded by termination could also be expanded. If $\hat{h}(x)$ is known to be monotone then each state is only expanded once so A^* has complexity $O(|X_e|)$ in the worst case. In general, if it is assumed that visiting a state is the basic operation then if $\hat{h}(x)$ is monotone, A^* runs in $O(|X_e|^2)$ steps in the worst case. (We shall use this latter characterization of computational complexity to compare A^*

to other conventional algorithms.) 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 [2]. In fact, if $\hat{h}(x)$ for $\mathbf{A}^*$ is monotone, then $\mathbf{A}^*$ uses the most effective scheme of any admissible algorithm for utilizing the heuristic information provided by $\hat{h}(x)$ [2]. The following Proposition follows immediately from the above discussions.

Proposition 2: If $\hat{h}(x)$ is monotone and $|X|$ is finite then the complexity of $\mathbf{A}^*(\hat{h}(x))$ is $O(|X_e|^2)$ and the complexity of the generalized Dijkstra's algorithm is $O(|X|^2)$ where $|X_e| \leq |X|$.

Proposition 2 indicates that: (i) $\mathbf{A}^*$ should always be chosen over the generalized Dijkstra's algorithm to solve the shortest state to set problem - provided that a monotone $\hat{h}(x)$ can be found, and (ii) if a monotone $\hat{h}(x)$ can be found then $||X|-|X_e||$ or the size of $\hat{h}(x)$ for all $x \in X_a$, quantifies the computational savings of $\mathbf{A}^*$ over the generalized Dijkstra's algorithm. Roughly speaking, the larger that $\hat{h}(x)$ can be chosen (still maintaining monotonicity) the fewer states $\mathbf{A}^*$ will have to expand to find an optimal trajectory. Results have in fact shown that for a wide class of graphs, if $\hat{h}(x)$ is monotone then $\mathbf{A}^*$ far outperforms the generalized Dijkstra's algorithm. For instance, in the case where $\hat{h}(x)$ is monotone it has been shown that for a wide class of randomly generated "Euclidean graphs" $\mathbf{A}^*$ operates with an average complexity of $O(|X|)$ [37]. Similar results on the reduction of search complexity obtained with $\mathbf{A}^*$ over the generalized Dijkstra's algorithm are provided in [5]. The following three remarks show it is possible to efficiently solve other optimal control problems via heuristic search.

Remark 1: In [30] the authors showed that a *minimum-input/event cost control problem* (costs for the inputs are allowed) and a *minimum-time control problem* (synchronicity was assumed) could be solved efficiently via the above techniques provided a monotone $\hat{h}(x)$ can be specified.

Remark 2: Assume that there exists some special cycle in X_{af} (e.g., minimum average cost cycle, minimum cost cycle) and name it s_c^*. Let $X_{af}=\{x: x \in s_c^*\}$ and use $\mathbf{A}^*$ to find an optimal state trajectory to X_{af} (if one exists). The result is a solution to an *optimal stabilization problem* with polynomial complexity provided that we can specify $\hat{h}(x)$ so that it is monotone [32].

Remark 3: Let A describe the allowable behavior for a plant P such that A[P] and assume that A is (x_0,X_{af})-reachable. Find a sequence of inputs $u \in Q_a^*$ that drives the system A along a *near-optimal (ε-optimal) state trajectory* s, i.e., $s \in \mathfrak{L}(A,x_0,X_{af})$ such that $J(s) \leq (1+\varepsilon)J(s^*)$ where $J(s^*)=\inf\{J(s):s \in \mathfrak{L}(A,x_0,X_{af})\}$ and $\varepsilon \geq 0$. A solution to this *near-optimal control problem* is provided in [29] where if ε is very small and $\hat{h}(x)$ is ε-monotone ($\hat{h}(x) \leq (1+\varepsilon)\chi(x,x')+\hat{h}(x')$ for all $(x,x') \in E(A)$) the complexity of the algorithm may be satisfactory for special problems. However, the complexity of the algorithm used to solve this problem was exponential in the worst case.

4. THE HEURISTIC FUNCTION

It is clear that it is very important to be able to specify an $\hat{h}(x)$ that is monotone; otherwise the complexity of A^* can become exponential in the worst case for finding solutions to the above optimal control problems. Unfortunately it is not easy to specify monotone heuristic functions for a wide class of applications; hence the use of A^* has been somewhat limited to special situations. This problem is resolved here by showing that a metric space approach provides a method to specify monotone (and hence admissible) heuristic functions for a very wide class of DES applications.

There has been extensive work on the problem of how to automatically generate heuristics for an arbitrary problem. In [3,7,8,33] the authors introduced respectively the related "problem similarity", "auxiliary problem", and "relaxed model" approaches to the generation of heuristics. The main deficiencies of these approaches is that they provided no way to systematically produce similar and auxiliary problems or relaxed models. Furthermore, in [39] it was proven that the approach in [7] can be computationally inefficient. Approaches similar to these have also been used in Operations Research [17,11]. As an extension to Pearl's (and the others) work the authors in [14] suggest a method for modelling a problem that will always lead to the derivation of a set of "simplified" sub-problems from which admissible and monotone heuristics can be derived algorithmically for the original problem. Their algorithm uses a problem decomposition algorithm to obtain the sub-problems and then uses exhaustive search to find the minimal cost optimal path in each sub-problem. From this, a heuristic which is admissible and monotone is generated. The problem with this approach is the reliance on an exhaustive search. While Irani and Yoo 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. Recently, it has been shown that for a class of "Vector DES", a linear integer programming approach can be used to specify

the heuristic function for $\mathbf{A}^*$ [19]. Unfortunately, there do not currently exist computationally efficient techniques to solve the linear integer programming problem.

Specifying The Heuristic Function: A Metric Space Approach

In our metric space approach to specifying the heuristic function there is no need to perform a search or use a mechanical decomposition 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. We will, however, require for some of the results below that the DES is modelled with certain DES models which have a "numerical state space", i.e. that $X \subset \mathbb{R}^n$. In this way we exploit the structure of the state space to obtain efficient solutions to optimal control problems.

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 $z \in Z$ and define $d(z,Z)=\inf\{\rho(z,z'): z' \in Z\}$. The value of d(z,Z) is called the *distance between point z and set Z*. Recall that if $x,y \in \mathbb{R}^n$, $x=[x_1\ x_2\ \cdots\ x_n]^t$, $y=[y_1\ y_2\ \cdots\ y_n]^t$, and $1 \leq p < \infty$, then $\rho_p(x,y)=[\Sigma_{i=1}^{n}|x_i-y_i|^p]^{1/p}$, $\rho_\infty(x,y)=\max\{|x_1-y_1|,|x_2-y_2|, \ldots ,|x_n-y_n|\}$, and ρ_d (*discrete metric*) where $\rho_d(x,y)=0$ if $x=y$ and $\rho_d(x,y)=1$ if $x \neq y$, are all valid metrics on $\mathbb{R}^n$ [21]. We shall frequently use these metrics in the following results and in Section 5.

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 ρ is a metric on X_a with $\rho(x,x') \leq \chi(x,x')$ for all $(x,x') \in E(A)$ then $\hat{h}(x)$ is admissible and monotone.

Proof: For admissibility let $s_{\bar{x}x''} \in \mathfrak{L}(A,\bar{x},X_{af})$ 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, $\rho(x,x'') \leq \rho(x,x')+\rho(x',x'')$. Using repeated applications of the triangle inequality along $s_{\bar{x}x''}$ we know that if $t=s_{\bar{x}x''}$

$$\rho(\bar{x},x'') \leq \sum_{(x,x') \in E_t(A)} \rho(x,x') \tag{13}$$

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

$$\sum_{(x,x') \in E_t(A)} \rho(x,x') \leq \sum_{(x,x') \in E_t(A)} \chi(x,x'). \tag{14}$$

Since this is true for any state path it is true for optimal ones also. Let $s^*_{\bar{x}x''} \in \mathfrak{L}^*(A,\bar{x},X_{af})$ (we need only consider cases where one exists). Then,

$$0 \le \rho(\bar{x},x'') \le \sum_{(x,x')\in E_t(A)} \chi(x,x') = J(s^*_{\bar{x}x''}) \qquad (15)$$

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 $s^*_{\bar{x}x''}\in \mathfrak{L}^*(A,\bar{x},X_{af})$ which guarantees the admissibility of $\hat{h}(x)$. For monotonicity let $s_{\bar{x}x''}\in \mathfrak{L}(A,\bar{x},X_{af})$ where $\bar{x}\in X_a$ and let $xx'\in s_{\bar{x}x''}$ be two successive states on $s_{\bar{x}x''}$. 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 $s\in \mathfrak{L}(A,\bar{x},X_{af})$ which guarantees the monotonicity of $\hat{h}(x)$. ■

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.

Remark 4: In [29] the authors use this same metric space approach to provide the first results on the automatic specification of $\hat{h}(x)$ for a near optimal control problem (first results for automatic specification of "semi-admissible" heuristics [34]). The result is the same as for Theorem 1 except it is required that $\rho(x,x')\le(1+\varepsilon)\chi(x,x')$ for all $(x,x')\in E(A)$ to get ε-monotonicity and hence ε-optimality (See Remark 3).

It is, however, not necessary to use the metric space notion of distance for the heuristic function as Theorem 2 shows.

Theorem 2: Let $\theta:X_a\times X_a\to\mathbb{R}_+$ and suppose that $\theta(x,x')\le\chi(x,x')$ for all $(x,x')\in E(A)$. For the DES P and A[P] there exist heuristic functions $\hat{h}(x)=\inf\{\theta(x,x'):x'\in X_{af}\}$ such that θ is not a metric on X_a, that are admissible and monotone.

Proof: Suppose that $\theta(x,x')=0$ for all $x,x'\in X_a$. Then θ is not a metric 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. Also, if $\hat{h}(x)\le\hat{h}'(x)$ for all $x\in X_a$, where $\hat{h}'(x)$ satisfies the conditions of Theorem 4, $\hat{h}(x)$ is admissible but 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 and also clarify the relationship between monotonicity and the triangle inequality which has only, in the past, been loosely referred to [9,34].

Theorem 1 can make it easier to specify $\hat{h}(x)$ because for many problems the conditions of Theorem 1 are easier to test than the admissibility and monotonicity conditions. Theorem 1 does not however make the task of specifying $\hat{h}(x)$ entirely simple; $\hat{h}(x)$ still must be chosen so that the constraint $\rho(x,x') \leq \chi(x,x')$ is met for all $(x,x') \in E(A)$ and one must, in fact, be able to specify a metric ρ on X_a. Theorem 3 and the following discussions show several ways to overcome these difficulties.

Let ρ be any metric on X_a and

$$\rho_a(x,x') = \beta \frac{\rho(x,x')}{1+\rho(x,x')} \tag{16}$$

where

$$\beta = \inf\{\chi(x,x') : (x,x') \in E(A)\}. \tag{17}$$

Let ρ_b be a bounded metric on X_a for $(x,x') \in E(A)$, i.e., for all $(x,x') \in E(A)$ there exists $\sigma > 0$ such that $\rho_b(x,x') \leq \sigma$. Let ρ_c be a metric on X_a and assume that $\rho_c(x,x') = \gamma\chi(x,x')$ for all $(x,x') \in E(A)$ for some $\gamma > 0$. Let ρ_β be a metric on X_a such that $\rho_\beta(x,x') = \beta$ if $x \neq x'$ and $\rho_\beta(x,x') = 0$ if $x = x'$ for all $(x,x') \in E(A)$.

Theorem 3: For the DES P and A[P] the heuristic functions:

(1) $\hat{h}_1(x) = \inf\{\rho_a(x,x_f) : x_f \in X_{af}\}$ (2) $\hat{h}_2(x) = \inf\{(\beta/\sigma)\rho_b(x,x_f) : x_f \in X_{af}\}$

(3) $\hat{h}_3(x) = \inf\{(1/\gamma)\rho_c(x,x_f) : x_f \in X_{af}\}$ (4) $\hat{h}_4(x) = \inf\{\rho_\beta(x,x_f) : x_f \in X_{af}\}$

are all admissible and monotone.

Proof: Due to the fact that there exists $\delta' > 0$ such that $\chi(x,x') \geq \delta'$ for all $(x,x') \in E(A)$, $\beta > 0$; hence it is easy to show that ρ_a, $(\beta/\sigma)\rho_b$, $(1/\gamma)\rho_c$, and ρ_β are all metrics on X_a. For (1), $\rho_a(x,x') \leq \chi(x,x')$ for all $(x,x') \in E(A)$ since $\rho_a(x,x') \leq \beta$ and $\beta \leq \chi(x,x')$ for all $(x,x') \in E(A)$. For (2), since $1 \leq (1/\beta)\chi(x,x')$ and $(1/\sigma)\rho_b(x,x') \leq 1$ for all $(x,x') \in E(A)$ we know that $(\beta/\sigma)\rho_b(x,x') \leq \chi(x,x')$ for all $(x,x') \in E(A)$. Clearly for (3), $(1/\gamma)\rho_c(x,x') \leq \chi(x,x')$ and for (4) $\rho_\beta(x,x') \leq \beta$ for all $(x,x') \in E(A)$. With this, the result follows immediately from Theorem 1. ■

To choose $\hat{h}_1(x)$ determine β and specify any metric ρ on X_a; using (16) and Theorem 3, $\hat{h}_1(x)$ will be admissible and monotone. To choose $\hat{h}_2(x)$ pick a bounded metric ρ_b on

X_a and determine σ; using Theorem 3, $\hat{h}_2(x)$ will be admissible and monotone. For $\hat{h}_3(x)$ it must be the case that the costs are in a special form then $\hat{h}_3(x)$ will be admissible and monotone. For $\hat{h}_4(x)$, ρ_β can be specified for any X_a; hence this choice will always be admissible and monotone (See Theorem 5 below also).

Notice that for each of the techniques in order to specify $\hat{h}(x)$ it is necessary to be able to specify a metric on X_a. In general, this may not be an easy task but as the next Remark and the following comments explain there are a wide variety of DES for which it is easy to specify a metric on X_a.

Remark 5: There is a wide class of DES whose state space can be modelled in terms of $X \subset \mathbb{R}^n$, e.g., X comprised of n-tuples of natural numbers or integers. As evidence of this fact we turn to the many applications of the theory of Petri nets [35] (e.g. General or Extended Petri nets), the use of such models in DES-theoretic research [16,12,38], and other related "Vector DES models" [18].

Theorem 3 says that there is no difficulty in specifying $\hat{h}(x)$ for all DES that can be modelled with P in (1) provided a valid metric ρ on X_a can be specified. Remark 4 indicates that there exists many DES that can be modelled as having a state space $X \subset \mathbb{R}^n$; hence there is no trouble specifying an admissible and monotone heuristic for the wide class of DES with $X \subset \mathbb{R}^n$ because there exist many metrics on $\mathbb{R}^n$ (e.g., ρ_p, ρ_d, and ρ_∞) and any metric on $\mathbb{R}^n$ is also a metric on X, where $X \subset \mathbb{R}^n$. Note that for particular DES applications many results similar to Theorem 3 exist since for ρ_p and ρ_∞ one can weight the various terms in the sum and max respectively; hence one has flexibility in specifying the heuristic function when this metric space approach is used.

Good Heuristic Functions

Theorems 1 and 3 provide an automatic procedure to specify $\hat{h}(x)$ for a wide class of DES; the use of such $\hat{h}(x)$ will allow $\mathbf{A}^*$ to produce solutions to optimal control problems in a computationally efficient manner. Next we seek to show how to make $\hat{h}(x)$ large so that even more computational savings can be obtained, i.e. fewer states will be expanded in finding the optimal state trajectory.

Consider the DES model $P'=(X,Q,\delta,\chi',x_0,X_f)$ defined as in (1) except $\chi':X\times X\rightarrow\mathbb{R}_+$ where χ' is a metric on 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. Notice that, in general, we are requiring that χ' be defined on some (x,x') such that $(x,x')\notin E(A)$. The allowable behavior $A'=(X_a,Q_a,\delta_a,\chi'_a,x_{a0},X_{af})$ such that A'[P'] as in (5). We call a

heuristic function *good* if $\hat{h}(x)=\inf\{\chi'(x,x_f):x_f\in X_{af}\}$ for all $x\in X_a$. The motivation for our definition of this new class of heuristic functions lies in the desire to choose $\hat{h}(x)$ as large as possible to get efficient search.

Theorem 4: 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 $\chi'(x,y)\geq\delta'$ for every $x,y\in X$ such that $x\neq y$. Since A^* prunes cycles it will not repeatedly investigate any single $(x,x')\in E(A')$ with $x=x'$ and $\chi'(x,x')=0$; hence if $\hat{h}(x)$ is good then $A^*(\hat{h}(x))$ is complete. By Theorem 1, $\hat{h}(x)$ is admissible and monotone. ■

This indicates that if we have a plant P' without costs for the events or a plant where it is not known how to specify the costs then Theorem 4 offers a method to assign the costs so that an efficient search for a solution to several optimal control problems is possible. In fact, for any P such that all the costs are equal (or where this can be assumed) the following result provides an admissible and monotone heuristic function.

Theorem 5: If $\chi'(x,x')=\gamma\rho_d(x,x')$ for all $(x,x')\in E(A)$ for some $\gamma>0$ then $\hat{h}(x)=\inf\{\chi'(x,x_f):x_f\in X_{af}\}$ is an admissible and monotone heuristic function for finding the solution to the OCP in the case where the costs are all equal to some γ where $\gamma>0$.

Proof: Even though $\chi'(x,x')=0$ when $x=x'$ such self-loops will be pruned by A^* so it does not matter that χ' doesn't precisely model the fact that all the costs are equal. The $(x,x')\in E(A)$ such that $\chi(x,x')\neq\gamma$ cannot be on any optimal state trajectory. The result follows directly from Theorem 4 since χ' is a metric. ■

Theorem 5 is quite useful in practice since often a solution is sought which will minimize the length of the input sequence. Theorems 4 and 5 illustrate how information from the plant (the knowledge that the costs were modelled with a metric) is used to focus 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 $A^*(\hat{h}(x))$ to more narrowly focus its search.

Theorem 6: For the DES P' and A'[P'] if $\hat{h}(x)=\inf\{\chi'(x,x_f):x_f\in X_{af}\}$ for all $x\in X_a$ then $|\hat{h}(x)-\hat{h}(x')|\leq\chi'(x,x')$ for all $(x,x')\in E(A')$.

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

We see that 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 A^* will not get side-tracked too much from finding an optimal solution.

Theorems 4-6 support the results in [37] where the authors show that if the costs can be defined by a metric then A^* has average complexity $O(|X|)$ for a wide class of randomly generated graphs and thus, on the average, far outperforms conventional algorithms in solving the shortest path from state to set problem. 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 hence optimal control problems can be solved efficiently.

5. DISCRETE EVENT SYSTEM APPLICATIONS

In this Section we apply the method in Section 3 and results in Section 4 to three DES applications: (i) an automated factory, (ii) an optimal part distribution problem in flexible manufacturing systems, and (ii) artificial intelligence (AI) planning problems. In each case we specify the model P for the problem, the allowable behavior A, and state the particular OCP. Then, using the results of Section 4 we specify admissible and monotone heuristic functions so that A^* can find solutions to the OCPs in a computationally efficient manner. A^* and the generalized Dijkstra's algorithm were implemented and ran to compare the complexity of the two algorithms. For all cases in the three examples A^* using a heuristic function chosen via the results in Section 4 significantly outperformed the generalized Dijkstra's algorithm.

Automated Factory Example

This first example is used to clarify (i) how the A^* algorithm operates to find optimal state trajectories, and (ii) how the results of Section 4 can be used to specify a wide variety

of heuristics for a simple route-planning problem. Consider an automated factory where there is a vehicle which is given the task of delivering parts to various locations in the factory while minimizing the distance it travels on a grid. Some "master controller" provides the vehicle with its task description which consists of:

(i) A set of positions of machines in the factory that need parts,

(ii) A map of the factory floor containing the locations of walls, pillars, etc.

The vehicle is to develop a plan of how to get from its starting position to the closest machine to which it is to deliver parts without colliding with the walls, pillars, etc. This is a standard route-planning problem which has been studied in many areas.

We begin by modelling the vehicle with the model $P=(X,Q,\delta,\chi,x_0,X_f)$ where,

(i) $X=\mathbb{N}^2$, $x_k=[x_1\ x_2]^t$ and $x_{k+1}=[x'_1\ x'_2]^t$ are the current and next states,

(ii) $Q=\{q_n,q_s,q_e,q_w\}$, (q_n-move north, q_s-move south, q_e-move east, q_w-move west)

(iii) $\delta(q_n,x_k)=x_k+[0\ 1]^t$, $\delta(q_s,x_k)=x_k-[0\ 1]^t$, $\delta(q_e,x_k)=x_k+[1\ 0]^t$, $\delta(q_w,x_k)=x_k-[1\ 0]^t$,

(iv) $\chi(x_k,x_{k+1})=1$ for all $x_{k+1}=\delta(q,x_k)$,

(v) $x_0=[0\ 0]^t$, and $X_f=\mathbb{N}^2$.

We suppose that the map (See Figure 1), which consists of the positions of walls (shaded regions) and the machines which need parts, is generated by the master controller and communicated to the vehicle. In terms of our formulation the map specifies the allowable DES behavior and hence we characterize it with A such that A[P] as follows: From Figure 1, let $X_b\subset X$ be given by $X_b=\{[0\ 3]^t, [1\ 3]^t, [1\ 1]^t, [3\ 3]^t, [3\ 2]^t, [3\ 1]^t, [4\ 1]^t, [5\ 1]^t, [6\ 1]^t, [6\ 2]^t, [6\ 3]^t, [7\ 3]^t, [8\ 3]^t, [9\ 3]^t, [10\ 3]^t, [8\ 0]^t, [8\ 1]^t, [9\ 1]^t, [10\ 1]^t\}$, and $X_o\subset X$ be given by $X_o=\{x_k\in X: x_i<0\}$. The sets X_b and X_o specify the positions of the walls and pillars. With this and the positions of the two machines we specify the allowable behavior as $A=(X_a,Q_a,\delta_a,\chi_a,x_{a0},X_{af})$ where

(i) $X_a=(X-X_b)-X_o$, $Q_a=Q$, and

(ii) $x_{a0}=x_0$, and $X_{af}=\{[4\ 2]^t, [5\ 2]^t\}$.

The functions δ_a and χ_a can easily be obtained from these definitions. The solution to the OCP for this problem will be a sequence of moves by the vehicle that will result in it delivering parts to the nearest machine in the least number of moves (i.e., the minimum distance on the grid). This example was first studied in [28].

Next we discuss how the A^* algorithm constructs an optimal state trajectory from x_0 to X_{af}. Clearly A is (x_0,X_{af})-reachable so we know that A^* will be able to find a solution. It is easy to specify a heuristic function. We use ρ_p with p=1 and $\hat{h}_1(x_k)=\min\{\rho_1(x_k,x_f): x_f\in X_{af}\}$; via Theorem 4 since $\rho_1(x_k,x_{k+1})=\chi(x_k,x_{k+1})$ for all $(x_k,x_{k+1})\in E(A)$, $\hat{h}_1(x)$ is good and hence admissible and monotone. The use of the "air distance" ρ_p with p=2, or ρ_∞ would also result in good heuristic functions

$\hat{h}_2(x_k)=\min\{\rho_2(x_k,x_f):x_f\in X_{af}\}$ and $\hat{h}_\infty(x_k)=\min\{\rho_\infty(x_k,x_f):x_f\in X_{af}\}$ but for congested factories where the robot stays on a grid of tracks $\hat{h}_1(x_k)$ is a better heuristic as it is shown for our example in Figure 1. Next, we explain how the A* algorithm operates.

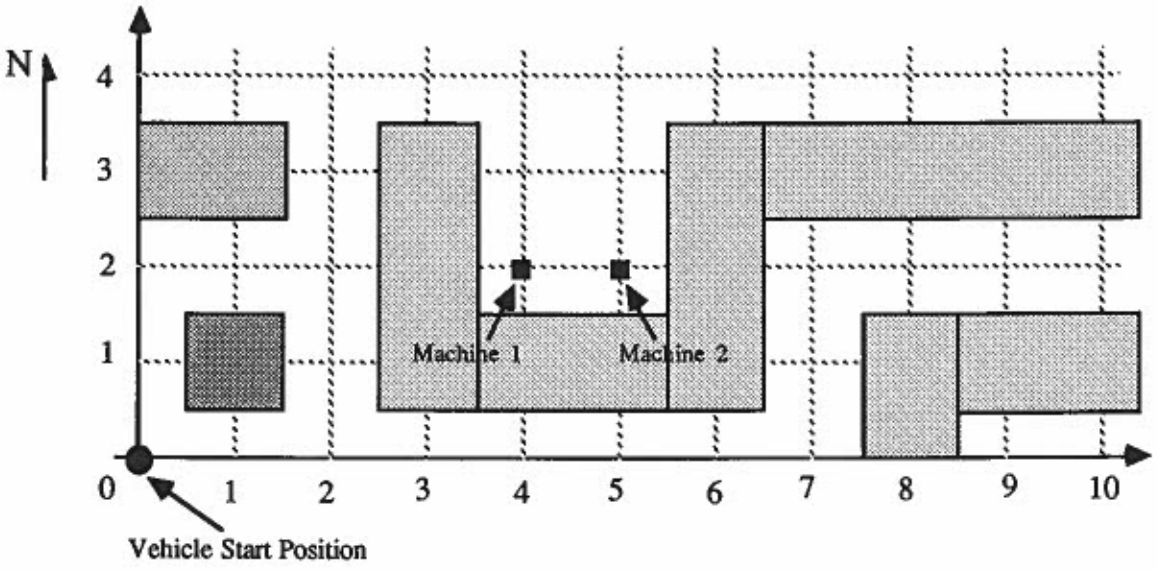

Figure 1. Automated Factory Floor

First consider $\hat{h}_1(x_k)=\min\{\rho_1(x_k,x_f):x_f\in X_{af}\}$. Since $\hat{h}_1(x_k)$ is monotone, Step 5 (iii) can be removed from A*. At the end of the first iteration of A* (at Step 6) O={([0 0]^t, [0 1]^t), ([0 0]^t, [1 0]^t)} and C={(x_d, [0 0]^t)}. At the end of iteration 7, O={([2 2]^t, [2 3]^t), ([2 0]^t, [2 1]^t), ([2 0]^t, [3 0]^t)} (assuming that for ties at Step 3 the order of states expanded was [0 0]^t, [1 0]^t, [0 1]^t, [0 2]^t, [1 2]^t, [2 2]^t, [2 0]^t). For iteration 8, Step 3, the values of $\hat{f}$ for each of the three pointers in O are 8, 6, and 6 respectively. Assume that we choose to expand [2 1]^t. At Step 5, $\mathcal{E}$([2 1]^t)={[2 2]^t} and there exists ([1 2]^t,[2 2]^t)$\in$ C; since $\hat{f}$ for ([2 1]^t,[2 2]^t) is not less than $\hat{f}$ for ([1 2]^t,[2 2]^t) we see that Step 5 (iii) will not be executed and hence at the end of iteration 8, O={([2 2]^t, [2 3]^t), ([2 0]^t, [3 0]^t)}. Next, A* expands [3 0]^t, [4 0]^t, [5 0]^t, [2 3]^t, and [6 0]^t. (Note that at iteration 11, $\hat{h}_1(x_k)$ measures machine 2 as closer than machine 1.) By the end of iteration 13, A* will expand only states on the optimal path, so it expands [2 4]^t, [3 4]^t, [4 4]^t, and [4 3]^t, then it finds that [4 2]$^t\in X_{af}$ so it terminates. The optimal trajectory is found by tracing back through pointers in C.

With $\hat{h}_1(x_k)$, A* expanded 17 states in finding the optimal path which is: [0 0]^t, [0 1]^t, [0 2]^t, [1 2]^t, [2 2]^t, [2 3]^t, [2 4]^t, [3 4]^t, [4 4]^t, [4 3]^t, [4 2]^t. Clearly, another optimal path would be for the vehicle to navigate south rather than north around the pillar at [1 1]^t. When A* is used with $\hat{h}_2(x_k)$ the operation is similar but 18 states are expanded ([7 0]^t is

also expanded); hence $\hat{h}_1(x_k)$ is a better heuristic. Moreover, if the generalized Dijkstra's algorithm is used many more states are expanded. For instance, $[7\ 1]^t$, $[7\ 2]^t$, $[2\ 5]^t$, and $[0\ 4]^t$ are expanded. In all, to solve the same OCP, the generalized Dijkstra's algorithm will expand at least 32 states before finding an optimal path. We see that A^* using either monotone $\hat{h}_1(x_k)$ and $\hat{h}_2(x_k)$ performs significantly better than the generalized Dijkstra's algorithm even for this simple example.

Optimal Parts Distribution Problem in Flexible Manufacturing Systems

A Flexible Manufacturing System (FMS) that is composed of a set of identical machines connected by a transportation system is described by a directed graph (M,T) where $M=\{1,2, ..., N\}$ represents a set of machines numbered with $i\in M$ and $T\subset M\times M$ is the set of transportation tracks between the machines. We assume that (M,T) is *strongly connected*, i.e., that for any $i\in M$ there exists a path from i to every other $j\in M$. This ensures that no machine is isolated from any other machine in the FMS. Each machine has a queue which holds parts that can be processed by any machine in the system (with proper set-up). Let the number of parts in the queue of machine $i\in M$ be given by $x_i\geq 0$. There is a robotic transporter that travels on the tracks represented by $(i,j)\in T$ and moves parts between the queues of various machines. The robot can transfer parts from any $i\in M$ to any other $j\in M$ on any path between i and j (it is assumed that the robot knows the path to take, if not A^* could be used to find it). The robot can transfer no more than $\beta\in N-\{0\}$ parts at one time between two machines. It is assumed that the robot knows the initial distribution of parts, the graph (M,T), and we wish to find the sequence of inputs to the robot of the form "move α ($\alpha\leq\beta$) parts from machine i to machine j" that will achieve an even distribution of parts in the FMS. In this way, we ensure that every machine in the FMS is fully utilized. It is assumed that no new parts arrive from outside the FMS and that no parts are processed by the machines while the redistribution takes place. Our example is similar to the "load balancing problem" in Computer Science except that we require that a minimum number of parts be moved to achieve an even distribution. Next, we specify the model P in (1) of this FMS.

Let $X=N^N$ denote the set of states and $x_k=[x_1\ x_2\ \cdots\ x_N]^t$ and $x_{k+1}=[x'_1\ x'_2\ \cdots\ x'_N]^t$ denote the current and next state respectively. Let $Q=\{u_{ij}^{\alpha}:\alpha\in N-\{0\}\}$ be the set of inputs where u_{ij}^{α} denotes the command to the robot to move α parts from machine i to machine j. The state transition function is given by $\delta(u_{ij}^{\alpha},x_k)=[x_1\ x_2\ \cdots\ x_i-\alpha\ \cdots\ x_j+\alpha\ \cdots\ x_N]^t$, the event cost function by $\chi(x_k,x_{k+1})=\alpha$, and $x_0=[x_{01}\ x_{02}\ \cdots\ x_{0N}]^t$. The set X_f characterizes the state (or states) for which we consider the the parts in the FMS to be at an even

distribution. Let int(x) denote the integer part of x (e.g. int(3.14)=3) and "mod" denote modulo. Let

$$L=\text{int}\left(\sum_{i=1}^{N}\frac{x_{0i}}{N}\right) \quad \text{and} \quad L_e=\left(\sum_{i=1}^{N}\frac{x_{0i}}{N}\right) \text{mod } N.$$

The value of L represents the amount of parts each machine would have if the parts could be evenly distributed and L_e represents the number of extra parts that we seek to, for instance, distribute across the first L_e machines. With this intent we let $\tilde{x}=[\tilde{x}_1\ \tilde{x}_2\ \cdots\ \tilde{x}_N]^t$ where $\tilde{x}_i=L+1$ for i, $i\leq L_e$ and $\tilde{x}_j=L$ for j, $L_e<j\leq N$ (other states where the parts are considered to be evenly distributed can be specified in a similar manner - an example of this is given below). We often let $X_{af}=\{\tilde{x}\}$, hence $\hat{f}(s_x)$ is easy to compute. Also note that for each $x\in X$ there are at most αN next states which will clearly be much less than |X|.

We let A=P, i.e., all valid DES behavior is allowable, but note that our solution will work for any allowable behavior A so long as A is (x_0,X_{af})-reachable. This is very important since it shows that if the robot is informed that some machine or transportation track is out of order, the robot can still evenly distribute the load for the remaining machines that are still appropriately connected to the FMS. It is in this sense that our solution is "fault tolerant".

The OCP for this optimal parts distribution problem involves finding a sequence of inputs u_{ij}^{α} to the robot which will result in it moving the least number of parts to achieve an even distribution, i.e., $x_k\in X_{af}$. By Proposition 1, if we can find an $\hat{h}(x_k)$ that is admissible then A^* will solve the OCP (possibly inefficiently). Here, we show that the metric space approach developed in Section 4 can be used to specify a monotone $\hat{h}(x_k)$ (and hence admissible) so that the OCP can be solved efficiently. First, consider using the metric ρ_p with p=1. Notice that $\rho_1(x_k,x_{k+1})=2\alpha$ for all $(x_k,x_{k+1})\in E(A)$. Hence, by Theorems 3 and 1, $\hat{h}_1(x_k)=(1/2)\rho_1(x_k,\tilde{x})$ ($\tilde{x}$ defined above) is admissible and monotone so we get an efficient solution to the OCP. Theorems 4 and 6 offer another possibility. Consider the metric ρ_∞. Notice that $\rho_\infty(x_k,x_{k+1})=\alpha$ for all $(x_k,x_{k+1})\in E(A)$, all $x_k\in X$ are isolated points, and hence $\hat{h}_\infty(x_k)=\rho_\infty(x_k,\tilde{x})$ ($\tilde{x}$ defined above) is a good heuristic function. By Theorem 4 it is admissible and monotone.

Consider the FMS with 3, 4, and 6 machines and track topologies shown in Figure 2. For the 3-machine FMS in Figure 2 let $\beta=1$ and $x_0=[10\ 0\ 4]^t$; then L=4 and $L_e=2$ and we choose $X_{af}=\{[5\ 5\ 4]^t\}$. $A^*(\hat{h}_1(x_k))$ and $A^*(\hat{h}_\infty(x_k))$ both expand 5 states and result in a optimal state trajectory of cost 5 (i.e., 5 parts is the minimum number of parts that must be moved to achieve a even distribution). The generalized Dijkstra's algorithm expands 36 states to find a solution. If we let $x_0=[11\ 3\ 2]^t$ then L=5 and $L_e=1$. If we choose

$X_{af}=\{[6\ 5\ 5]^t\}$, $A^*(\hat{h}_1(x_k))$ and $A^*(\hat{h}_\infty(x_k))$ both expand 11 states and result in a optimal state trajectory of cost 5. The generalized Dijkstra's algorithm expands 51 states to find a solution; hence we see that for the 3-machine FMS A^* using the heuristic functions specified via the results of Section 4 far outperforms the generalized Dijkstra's algorithm.

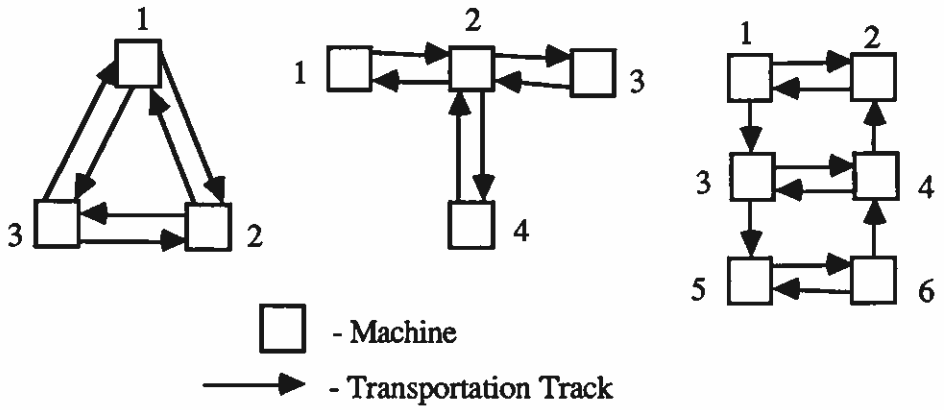

Figure 2. Example Flexible Manufacturing System Topologies

For the 4-machine FMS in Figure 2 let $\beta=1$ and $x_0=[0\ 5\ 2\ 6]^t$ so that L=3 and $L_e=1$. Choose $X_{af}=\{[4\ 3\ 3\ 3]^t, [3\ 3\ 3\ 4]^t\}$. $A^*(\hat{h}_1(x_k))$ and $A^*(\hat{h}_\infty(x_k))$ expand 38 and 53 states respectively and result in a optimal state trajectory of cost 6 that ends in $[3\ 3\ 3\ 4]^t$. The generalized Dijkstra's algorithm expands 141 states to find a solution to the OCP for the 4-machine FMS.

For the 6-machine FMS in Figure 2 let $\beta=1$ and $x_0=[4\ 0\ 1\ 2\ 0\ 5]^t$ so that L=2 and $L_e=0$. Let $X_{af}=\{[2\ 2\ 2\ 2\ 2\ 2]^t\}$. $A^*(\hat{h}_1(x_k))$ expands 82 and results in a optimal state trajectory of cost 6. The generalized Dijkstra's algorithm expanded 798 states to produce the same solution.

Note that if we had allowed $\beta>1$ for the above examples then the computational savings obtained by using A^* over the generalized Dijkstra's algorithm would even be more pronounced. This is the case since A^* would exploit the fact that the robot could move multiple parts so that an even distribution could be achieved quicker. For the generalized Dijkstra's algorithm large β will drastically increase the number of states it visits in finding an optimal state trajectory. Also note that for large N and total number of parts initially in the FMS, for many FMS track topologies the OCP can easily become too difficult to solve via any method due to combinatorial explosion. We have, however, illustrated that for typical FMS systems the A^* algorithm, with the appropriate heuristic function, can solve the optimal parts distribution problem efficiently - and with significantly fewer computations than conventional techniques.

Artificial Intelligence Planning Problems

Several fundamental relationships between AI planning systems and control systems have recently been identified in [31]. Here we show that a class of AI planning problems falls into our DES theoretic framework and that the results of Section 4 provide a method to specify heuristic functions so that OCPs can be solved efficiently for AI planning problems. The A^* algorithm has already been used for the solution to many AI planning problems such as tic-tac-toe, the 8 and 15 puzzle, etc. [34]. The extensions to the theory of heuristic search in this paper allow for a wider variety of such problems to be studied. For instance, in [27] the authors showed that the metric space approach could be used to specify the standard heuristic functions for the 8-puzzle and discovered several new heuristics for this problem that also work for the more general N-puzzle. In [26] heuristic functions were specified for a "triangle and peg" problem and a simple robotics problem ("blocks world"). Here we study the Missionaries and Cannibals Problem as in [26], an AI planning problem for which there currently exist no admissible and monotone heuristic functions. In this way we illustrate that the results of Section 4 facilitate the discovery of new heuristics.

The problem statement is as follows: Three missionaries and three cannibals are trying to cross a north-south river by crossing from east to west. As their only means of navigation, they have a small boat, which can hold one or two people. If the cannibals outnumber the missionaries on either side of the river, the missionaries will be eaten; this is to be avoided. Find a way to get them all across the river which minimizes the number of boat trips taken.

First we model this problem with the DES model P. Let $X=\mathbb{N}^6$ and $\mathbf{x}_k=[x_1\ x_2\ \cdots\ x_6]^t$ and $\mathbf{x}_{k+1}=[x'_1\ x'_2\ \cdots\ x'_6]^t$ denote the current and next state respectively. Let x_1 (x_4) and x_3 (x_6) denote the number of cannibals and missionaries on the east (west) side of the river respectively. Let "E" and "W" denote the east and west side of the river respectively. Let "C" and "M" denote cannibals and missionaries. Let $Q=\{q_i:i=1,2,\ ...,\ 10\}$ where q_1=2 C W$\rightarrow$E (move 2 cannibals from the west side of the river to the east side of the river); q_2=2 C E$\rightarrow$W; q_3=1 C W$\rightarrow$E; q_4=1 C E$\rightarrow$W; q_5=1 C 1M W$\rightarrow$E (move 1 cannibal and 1 missionary from the west side of the river to the east side of the river); q_6=1 C 1M E$\rightarrow$W; q_7=1M E$\rightarrow$W; q_8=1 M E$\rightarrow$W; q_9=2 M E$\rightarrow$W; q_{10}=2 M W$\rightarrow$E. Of course the boat moves in the indicated direction also. For the state transition function we have $\delta(q_2,[3\ 1\ 3\ 0\ 0\ 0]^t)$ = $[1\ 0\ 3\ 2\ 1\ 0]^t$; the other cases are defined similarly. Let $\chi(\mathbf{x}_k,\mathbf{x}_{k+1})=1$ for all $(\mathbf{x}_k,\mathbf{x}_{k+1})\in E(P)$, $\mathbf{x}_0=[3\ 1\ 3\ 0\ 0\ 0]^t$, and $X_f=\{[0\ 0\ 0\ 3\ 1\ 3]^t\}$.

Notice that we have not represented the part of the problem which states that "the cannibals cannot outnumber the missionaries". We will consider this to be included in the design objectives using the allowable behavior A such that A[P]. Let $X_b=\{x_k \in X: x_1>x_3$ or $x_4>x_6\}$ and $X_a=X-X_b$, $Q_a=Q$, and the definition of A follows immediately. The OCP for the missionaries and cannibals problem is to find the minimum length sequence of inputs (loads of passengers) that will result in all persons on the west side of the river.

Currently, there does not exist any monotone $\hat{h}(x_k)$ for this problem. We now show that the results of Section 4 allow for the specification of several such $\hat{h}(x)$. First consider ρ_p where p=2 and notice that $\rho_2(x_k,x_{k+1}) \leq \sqrt{10}$ and $\chi(x_k,x_{k+1})=1$ for all $(x_k,x_{k+1}) \in E(A)$ so by Theorems 3 and 1 $\hat{h}(x_k)=(1/\sqrt{10})\rho_2(x_k,\bar{x})$ where $\bar{x}=[0\ 0\ 0\ 3\ 1\ 3]^t$ is an admissible and monotone heuristic function. Also notice that $\rho_\infty(x_k,x_{k+1}) \leq 2$ so by Theorems 3 and 1 $\hat{h}(x_k)=(1/2)\rho_\infty(x_k,\bar{x})$ where $\bar{x}=[0\ 0\ 0\ 3\ 1\ 3]^t$ is an admissible and monotone heuristic function. When these heuristic functions are used with A^* to find the solution to the OCP, the minimum length sequence of inputs found was: $q_6, q_8, q_2, q_3, q_9, q_5, q_9, q_3, q_2, q_8, q_6$. The solution involves 11 boat trips, the minimum number of trips needed to solve the problem.

These AI planning problems 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 DES control community.

6. CONCLUSIONS

We showed how to adapt the A^* algorithm for the efficient solution of several optimal control problems for a wide class of DES. It was shown that for the class of DES modelled by P as defined in (1), Theorem 1 offers a method to specify admissible and monotone heuristic functions. In the case where $X \subset \mathbb{R}^n$ (e.g. for Extended Petri nets), via Theorem 3 and Remark 4 we showed that our metric space approach can be used to *automatically* specify admissible and monotone heuristic functions. It was shown that if this heuristic function is subsequently used by A^*, it would, in a computationally efficient manner return a solution to a variety of optimal control problems for a wide class of DES. We showed via Theorems 4-6 that if the costs of the events could be modelled with a metric then further computational savings can be expected. We applied the results to an automated factory, an optimal parts distribution problem in flexible manufacturing systems, and an AI planning problem. In each case we showed that our main results in Section 4 provided a technique to automatically specify an admissible and monotone $\hat{h}(x)$ and that when A^* uses this $\hat{h}(x)$ there is a significant reduction in the complexity of finding solutions to the OCPs.

Our results help to set the foundation for the development of a complete theory of optimal control for DES.

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] 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.

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

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

[5] Golden B.L., Ball M., "Shortest Paths with Euclidean Distances: An Explanatory Model", Networks, Vol. 8, pp. 297-314, 1978.

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

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

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

[9] 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.

[10] 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.

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

[12] Holloway L.E., Krogh B.H., "Synthesis of Feedback Control Logic for a Class of Controlled Petri Nets", IEEE Trans. on Automatic Control, Vol. 35, No. 5, pp. 514-523, May 1990.

[13] 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.

[14] 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, pp. 676-686, Sept. 1988.

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

[16] Krogh B.H., "Controlled Petri Nets and Maximally Permissive Feedback Logic", Proc. of the Allerton Conf. on Communication, Control, and Computing, Univ. of Illinois, pp. 317-326, Oct. 1987

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

[18] 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, pp. 1140-1149, Sept. 1988.

[19] Li Y., Wonham W.M., "A^* Algorithm for Vector Discrete-Event Systems", Systems Control Group Technical Note 891006, Oct. 1989.

[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] 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.

[27] 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, pp. 350-355, Arlington VA, August 1988.

[28] 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.

[29] Passino K.M., Antsaklis P.J., "Near-Optimal Control of Discrete Event Systems", Proc. of the Allerton Conf. on Communication, Control, and Computing, pp. 915-924, Univ. of Illinois, Sept. 1989.

[30] Passino K.M., Antsaklis P.J., "On the Optimal Control of Discrete Event Systems", Proc. of the Conference on Decision and Control, Tampa, Florida, pp. 2713-2718, Dec. 1989.

[31] 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.

[32] Passino K.M., Antsaklis P.J., "Optimal Stabilization of Discrete Event Systems", To appear in the Proc. of the IEEE Conf. on Dec. and Control, Hawaii, Dec. 1990.

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

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

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

[36] Ramadge P.J., Wonham W.M., "Supervisory Control of a Class of Discrete Event Processes", SIAM J. Control and Optimization, Vol. 25, No. 1, pp. 206-230, Jan. 1987.

[37] Sedgewick R., Vitter J.S., "Shortest Paths in Euclidean Graphs", Algorithmica, Vol. 1, pp. 31-48, 1986.

[38] Ushio T., "Maximally Permissive Feedback and Modular Control Synthesis in Petri Nets with External Input Places", IEEE Trans. on Automatic Control, Vol. 35, No. 7, pp. 844-848, July 1990.

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

[40] White D.J., Dynamic Programming, Holden-Day, San Francisco CA, 1969.