The Discrete Event Modeling and Trajectory Planning of Robotic Assembly Tasks

B. J. McCarragher
Department of Engineering, Facilities,
The Australian National University,
Canberra ACT 0200 Australia

H. Asada
Department of Mechanical Engineering,
Massachusetts Institute of Technology,
Cambridge, MA 02139

1 Introduction

Task level planning, task-level process monitoring and task-level control are keys to improving the success of robotic assembly. The focus of this paper is to present a new approach to the task-level understanding of robotic assembly, modeling assembly as a discrete event dynamic system. The abstraction to discrete event modeling dissects the assembly process into small, simple tasks according to the points of contact between the workpiece and the environment. It also highlights the necessary transitions for successful assembly. Moreover, the abstraction allows for planning on a task level rather than the cumbersome process of exact trajectory planning.

A discrete event system is a dynamic system in which the state vector is discrete. Due to the discrete nature of the state vector, state changes occur at discrete points in time in response to the occurrence of certain events. Typically, discrete event dynamic system models arise from certain aspects of manufacturing systems and data network protocols. In cog programming to minimize the path length and the uncertainty during assembly, Lastly, an optimal event trajectory is calculated to demonstrate the method. This paper lays the foundation for discrete event modeling for robotic assembly. An new avenue for the analysis and synthesis of significant aspects of the assembly process is opened.

1 Introduction

Task level planning, task-level process monitoring and task-level control are keys to improving the success of robotic assembly. The focus of this paper is to present a new approach to the task-level understanding of robotic assembly, modeling assembly as a discrete event dynamic system. The abstraction to discrete event modeling dissects the assembly process into small, simple tasks according to the points of contact between the workpiece and the environment. It also highlights the necessary transitions for successful assembly. Moreover, the abstraction allows for planning on a task level rather than the cumbersome process of exact trajectory planning.

A discrete event system is a dynamic system in which the state vector is discrete. Due to the discrete nature of the state vector, state changes occur at discrete points in time in response to the occurrence of certain events. Typically, discrete event dynamic system models arise from certain aspects of manufacturing systems and data network protocols. In cog programming to minimize the path length and the uncertainty during assembly, Lastly, an optimal event trajectory is calculated to demonstrate the method. This paper lays the foundation for discrete event modeling for robotic assembly. An new avenue for the analysis and synthesis of significant aspects of the assembly process is opened.

1 Introduction

Task level planning, task-level process monitoring and task-level control are keys to improving the success of robotic assembly. The focus of this paper is to present a new approach to the task-level understanding of robotic assembly, modeling assembly as a discrete event dynamic system. The abstraction to discrete event modeling dissects the assembly process into small, simple tasks according to the points of contact between the workpiece and the environment. It also highlights the necessary transitions for successful assembly. Moreover, the abstraction allows for planning on a task level rather than the cumbersome process of exact trajectory planning.

A discrete event system is a dynamic system in which the state vector is discrete. Due to the discrete nature of the state vector, state changes occur at discrete points in time in response to the occurrence of certain events. Typically, discrete event dynamic system models arise from certain aspects of manufacturing systems and data network protocols. In cog programming to minimize the path length and the uncertainty during assembly, Lastly, an optimal event trajectory is calculated to demonstrate the method. This paper lays the foundation for discrete event modeling for robotic assembly. An new avenue for the analysis and synthesis of significant aspects of the assembly process is opened.
paper presents a new way of describing the discrete event nature of robotic assembly using Petri nets. Petri nets are a compact mathematical way of describing the geometric constraints and the admissible transitions for an assembly task. Moreover, Petri nets are a useful method for describing the undetermined nature of robotic assembly, by incorporating transitions that are possible given the uncertainties, unknowns and errors in the system. The ability to address these unknowns is one of the primary strengths of the Petri net modeling method.

2.1 Contact State Definition. In order to explicitly describe the assembly process, we first need to define the workpiece and environment geometries. We define the geometries according to the edges and surfaces of the polygonal models [1]. The edges, or vertices, of both the workpiece and the constraint geometry are denoted generally as \( \phi \), and are given alphabetic labels as shown in Fig. 1. Additionally, the facets, or surfaces, of both the workpiece and the constraint geometry are denoted generally as \( \eta \), and are given a set of numerical labels. Possible contact pairs \( (\phi, \eta) \) are not identifiable by combinations of alphanumeric labels. A contact pair denotes the workpiece part and the constraint geometry part that are in contact with one another. In this manner, any discrete state can now be described by giving the alpha-numeric pairs designating the currently active contacts. For example, in Fig. 2, the two point contact is given by

\[
\text{(facet } 1 - \text{edge } c) \quad \text{and} \quad \text{(edge } c - \text{facet } 7) \quad (1)
\]

The justification for the discrete event modeling of robotic assembly can be seen in the natural selection of the state variables. The discrete state vector for assembly will be the discrete states of contact. The set of contact pairs given by (1) is one example of a discrete state in the assembly process. Correspondingly, a discrete event will be a change in the discrete state vector, i.e., a change in the discrete state of contact. The discrete events correspond to either a gain or a loss of a contact pair. Thus, in modeling the assembly process as a discrete event system we have reformulated the insertion problem into a higher, task-level control problem where it is desired to determine and control the changes of contact that lead to a successful insertion.

2.2 Petri Nets. A standard Petri net is composed of four parts [13]: a set of places \( P \), a set of transitions \( T \), an input function \( \delta \), and an output function \( \gamma \). In addition we define a set of discrete controls \( I \). The input function \( \delta \) is a mapping from the places to the transition, while the output function \( \gamma \) is a mapping from the transitions to the places. The number of discrete controls is identical to the number of transitions, and is denoted \( \gamma \). The number of places is \( p \). Places can be thought of as conditions of the current state that enable the transitions to occur, or fire. Likewise, controls can be thought of as conditions of the current state that enable the transitions to occur. The difference is that places are a function of the state of the system, whereas the controls are external inputs to the system chosen by the designer. It is important to note that the controls only allow or prevent a transition from occurring, they do not force the transition to occur.

In modeling the assembly process with a Petri net, we define a place to represent one contact pair \( (\phi, \eta) \); either a surface of the workpiece with an edge of the fixture or an edge of the workpiece with a surface of the fixture. Essentially, this means that each place represents only one constraint equation. Combinations of places can be used to describe cases of two-point contact (Fig. 2) where two constraints are simultaneously active. Additionally, edge-edge contact can be described as the combination of two places as shown in Fig. 3. To make the description complete there is also a place modeling the condition of no-contact, that is, the null constraint equation. Consistent with the definition of a place, we define a transition as the gaining or losing of a single contact pair (or constraint). Therefore, the occurrence of a transition is a discrete event, or change in state. The input function defines the places that must be active for a given transition to fire; that is, the contact pairs necessary for a given change of contact to occur. When the place conditions for a given transition are met, the transition is said to be place-enabled. The output function defines the contact pairs resulting from a discrete event. The set of discrete controls are defined such that there is one control variable for each transition. The controls are binary valued, with a "one" indicating that the associated transition is enabled, and a "zero" indicating that the transition is disabled. A transition with a control variable of "one" is said to be control enabled. A transition that is both place enabled, as defined by the input function, and controlled enabled is simply referred to as enabled.

A marking \( \gamma_0 \) of a Petri net is a \((p \times 1)\) vector assignment of tokens to the places of the net, where \( p \) indicates the number of places; that is, the total number of possible contact pairs. Tokens can be thought of residing in the places of the net. A token residing in a place indicates that the given edge and surface are in contact. That is, the constraint represented by that place is currently active. For our purposes here, a place can only have either zero or one token.

The execution of the Petri net is controlled by the discrete controls and the distribution of the tokens. If a token exists in each of the input places to a transition, that transition is said to be place enabled. A transition that is both place and control enabled may fire. A transition fires by removing the token from each of the input places and establishing a token in each of the output places. Note that a place may be both an input and an output of a single transition. The firing of a transition results in a next desired marking \( \gamma_t \), describing the new state of contact.

Journal of Dynamic Systems, Measurement, and Control

Fig. 1 Symbiotic representation of geometric models

Fig. 2 Two point contact

Fig. 3 Edge-edge representation using Petri nets

SEPTEMBER 1995, Vol. 117 / 365
A Petri net model of a single peg-in-the-hole assembly

represented by the distribution of tokens. By enabling various transitions and executing the net, we can direct the system through a series of discrete contact changes (events) to the desired final state.

A Petri net can be represented graphically, with a circle representing a place, a bar representing a transition, a box representing a discrete control, and a bullet representing a token. Directed arcs connect the places and the transitions. An arc from a place into a transition represents an input of that transition. An arc from a transition to a place represents an output of that transition. An arc from a discrete control into a transition indicates the control for that transition.

A Petri net model for a limited portion of the assembly process is shown in Fig. 4. For this model, the set of places is given by

$$\mathcal{P} = \{p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$$

(2)

where the places and their associated contact pairs are

$$p_0 \text{ no contact}$$
$$p_1 \text{ (b - 5)}$$
$$p_2 \text{ (1 - e)}$$
$$p_3 \text{ (b - 6)}$$
$$p_4 \text{ (2 - e)}$$
$$p_5 \text{ (c - 5)}$$
$$p_6 \text{ (2 - f)}$$
$$p_7 \text{ (c - 7)}$$

(3)

Thus, the number of places is $p = 8$. The set of transitions is given by

$$\mathcal{T} = \{t_1, t_2, \ldots, t_8\}$$

(4)

The number of transitions is $c = 26$. The set of controls is given by

$$\mathbb{C} = \{c_1, c_2, \ldots, c_8\}$$

(5)

The marking shown in Fig. 4 is given by a $(p \times 1)$ vector:

$$\gamma = [0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0]^\top$$

(6)

indicating that the system is in the two point contact case shown in Fig. 2.

Petri nets can be effectively represented in a matrix formulation. We define two matrices $N' \in \mathbb{R}^{p \times c}$ and $N'' \in \mathbb{R}^{c \times c}$ to represent the input and output functions. An element

$$n'_{ij} = 1$$

(7)

if the $i$th place is required as input to the $j$th transition, otherwise, it is zero. Correspondingly, an element

$$n''_{ij} = 1$$

(8)

if the $i$th place is an output place of the $j$th transition, otherwise it is zero. The following input and output matrices define the possible transitions that are considered for the example of Fig. 4. Note that some of these transitions are not shown in the figure.

$$N' = \begin{bmatrix}
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 
\end{bmatrix}$$

(9)

$$N'' = \begin{bmatrix}
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 
\end{bmatrix}$$

(10)

The input and output matrices can be combined to give the composite change matrix $N$.

$$N = N'' - N'$$

(11)

The composite change matrix describes the overall change in the marking of a net when a transition fires. If a discrete control

$$b_i = 1$$

(12)

Transactions of the ASME
then transition \( t_i \) is enabled. The transition \( t_i \) is disabled if the discrete control is
\[
b_i = 0
\]  
(13)

Additionally, we define a \((c \times c)\) matrix \( B \) to be a diagonal matrix of the discrete controls.
\[
B = \text{diag} \{ b_i \}
\]  
(14)

These matrices can be used to evaluate some of the properties of the Petri net. For example, determine if a transition \( t_i \) is enabled. Let \( \mathbf{v} \) be a unit \( c \)-dimensional vector which is zero everywhere except in the \( i \)-th component. The transition \( t_i \) is placed in a marking \( \mathbf{y} \) if there exists one token in each of the places that are inputs to \( t_i \), that is
\[
\mathbf{y} = n_i \mathbf{v} \quad \forall i \quad 1 \leq i \leq p
\]  
(15)

where \( n_i \) is the \( i \)-th component of the marking vector. A transition is control enabled if the diagonal element of the control matrix \( B \) corresponding to the transition is equal to one.

Second, we can determine the result of firing transition \( t_i \), called the next-marking \( \mathbf{y}' \), assuming the transition is enabled. The input matrix specifies the tokens that are to be removed when \( t_i \) fires, while the output matrix specifies the tokens that are to be added when \( t_i \) fires. Combining the input and output matrix changes with the current marking gives the next marking \( \mathbf{y}' \):
\[
\mathbf{y}' = \mathbf{y} - N' \mathbf{B}_i + N' \mathbf{B}_o = \mathbf{y} + N \mathbf{B}_o
\]  
(16)

where \( \Delta y \) is the change in marking. Equation (16) is the governing equation of the discrete events of the assembly task. For example, if we consider a sequence of transitions, we can use Eq. (16) repeatedly to determine what results from the execution of the Petri net, that is, firing the sequence of transitions. To perform a task, then, we wish to specify the discrete controls \( B \) that direct the system such that the desired markings occur through the occurrence of the desired discrete events. From the specified discrete controls, robot velocity commands need to be derived so that the Petri net is executed as desired. The process of determining the robot velocity commands is described in [11].

2.3 Advantages of Petri Net Modelling. Petri net modelling has several significant advantages for the modelling of assembly tasks. The first advantage can be seen in the definition of a place in a Petri net as a single contact pair rather than as a state of contact. By defining a place as a single contact pair we have decoupled the constraint equations, allowing us to address each constraint individually. Nonetheless, the constraint equations can be vectorially added to give the set of constraint equations that apply to any contact state. The ability to decouple the contacts plays a significant role in process monitoring using force sensing since the reaction forces in the directions normal to the surfaces of contact can be vectorially added as well [10].

Additionally, decoupling the constraint equations allows us to decouple the inadmissible motion space for the various contact pairs. This ability will significantly simplify the task of finding appropriate velocity commands [11].

A second advantage is the ability of Petri nets to represent a task level causality in the assembly process. Since we only allow the gain or loss of a single contact pair at any instance, a noncausal situation cannot arise. Each contact must be gained in its turn, and each contact must be lost in its turn. Additionally, Petri net modelling explicitly states transition conditions in terms of each contact pair. That is, before a transition may fire, it must be placed enabled by the Petri net (12). Thus, there is no ambiguity in the origin of the contact pairs. This is significantly different from the traditional methods.

For instance, consider the following five states of contact shown in Fig. 5. According to the traditional contact state network [1], the transition from \( S_3 \) to \( S_5 \) through \( S_4 \) is a valid series of transitions. Correspondingly, the transition from \( S_1 \) to \( S_3 \) through \( S_2 \) is also a valid series of transitions. However, neither of these paths is feasible. Such transitions may be geometrically admissible, but in the transition from \( S_4 \) requires a discontinuity in the rotation angle. Figure 5 also shows the rotation angle conditions for each of the states. Since \( S_3 \) is an instantaneous contact, a discontinuous jump in rotation angle is required to follow either of the paths described above.

This problem can be solved by incorporating some condition describing how the object arrived at state \( S_4 \). To get to state \( S_4 \), the preceding state must be \( S_2 \) and not \( S_1 \), based on the contact pairs. This type of causal relationship can be effectively represented using Petri nets. In the Petri net shown in Fig. 6, contact states \( S_4 \) can be described either by tokens residing in \( p_1 \) or by tokens residing in \( p_2 \). Each representing two individual contact pairs. The path from state \( S_2 \) to \( S_1 \) is then given by the firing of transitions \( t_1, t_2 \), whereas the path from state \( S_2 \) to \( S_4 \) is given by the firing of \( t_1, t_5 \). In order for \( t_1 \) to fire, a token must reside in \( p_1 \) rather than in \( p_2 \). We avoid a path from \( S_2 \) to \( S_1 \) by incorporating non-cause transitions. By requiring a token to reside in a given place for a transition to fire and allowing only one contact pair to be altered, the Petri net incorporated a task-level causality in the discrete event model of the assembly process.

Another advantage that results from defining each place to be a single contact pair is the compactness of the Petri net representation. For example, with the traditional contact state network, 21 contact state nodes are necessary to describe some of the possible discrete states [7]. By contrast, only 11 places in the Petri net are needed to describe the same 21 contact states. Moreover, with the 11 Petri net places, an additional 6 contact states are identified and incorporated. The difference between the number of required nodes and places will generally increase with the more complicated geometries. Thus, Petri net modeling allows for a more compact and complete representation than the traditional methods.

3 The Discrete Event Trajectory

Before we can determine the specific discrete control commands to specify the desired events, we need to determine a set of desired markings that trace out a discrete event trajectory [9]. The desired markings constitute the strategy for assembly. Taken together, their time history indicates the path of discrete events, that is, contact gains and losses, that the workpiece should experience enroute to the final, fully-assembled state. Given the path of discrete events, or discrete event trajectory, the discrete controls that enable the desired transitions in the
event trajectory are turned 'on'. By default, then, the controls that are associated with the transitions not included in the desired path of discrete events are turned 'off'. In this section we will develop a means to determine a discrete event trajectory and the associated discrete event controls. The methodology of mapping the discrete controls into continuous robot velocity commands and the demonstration of the effectiveness of the technique is described in [11].

3.1 Performance Measure. The desired markings could be specified by the engineer designing the assembly station. However, we propose a method for determining a desired marking path through the network that is based on the dynamics of the network. Here we consider two overriding principles of robotic assembly—minimum path and minimum uncertainty—and how they can be used to determine a discrete event trajectory.

As an objective measure of path length, we will use the number of transitions required to bring the system to the final marking configuration. There are several reasons why a minimum number of transitions is desirable. First and most important, every transition represents a change in the state of contact and thus a change in the type of constraints. Changing the constraint affects the desired velocity and often the desired force commands as well. The manipulator is required to alter its trajectory to accommodate for the change in constraint. Minimizing the number of times the control commands are changed will reduce many of the problems associated with differences in the constraint geometry. Second, the number of transitions roughly correlates to the speed at which the assembly is accomplished. Transitions indicate changes in momentum that slow the assembly process. Third, a minimum number of transitions will inhibit the system from repeating motions. The cyclic motion of bouncing between the sides of the hole should not occur in the desired path.

It was shown earlier that the governing equation for the Petri net is

\[ \gamma_t = \gamma + N(B_t) \]  

(17)

By repeatedly applying this equation we derive an equation for the final marking of the Petri net \( \gamma_f \).

\[ \gamma_f = \gamma + N(B_f) \]  

(18)

where \( t_i \) represents the required transitions to get from the current marking to the final marking, assuming the transitions are enabled. Therefore, we define our path length measurement to be the squared magnitude of the transition vector \( t \).

\[ \text{squared path length} = t_i^2 \]  

(19)

Path length alone will yield an event trajectory that is very risky and difficult to execute because contact states and transitions may be chosen with no consideration for how reliable they may be. Therefore, in addition to minimizing the path length,

\[ \text{uncertainty} = -\eta^T \gamma \]  

(20)

Combining Eqs. (19) and (20) we have our overall performance index, \( P_f \). Note, however, that the two terms in the performance index may have contradictory effects. For example, a transition that gets us one step closer to the final state may require a contact to be lost. In this case, the path length term is 1, but the uncertainty term is \(-1\), for a net change in the performance index of 0. Thus, the performance index indicates that no change in contact state should be taken. However, the change is necessary for a successful insertion. This dilemma is overcome by weighting the path length term greater than the uncertainty term. The path length term is weighted greater because it is more important to get to the final state than to be in the more certain state. Thus, the performance index is

\[ P_f = t_i^2 - \eta^T \gamma \]  

(21)

where \( 0 < \eta < 1 \) is a relative weight between the path length term and the uncertainty term.

3.2 Discrete Event Trajectory Optimization. Now that the performance function has been specified, the next objective is to determine the path in the network that minimizes this criterion. Because the initial and final conditions are known and because the search space is already discretized, the search for the minimum is directly amenable to the method of dynamic programming. Dynamic programming repeatedly applies the principle of optimality to sequences of decisions to define the optimal trajectory. Essentially, the routine steps backward through the discretized space, minimizing the performance measure at each step.

As an example of determining the time series of desired markings, we again use the peg-in-the-hole example of Fig. 4. Note that the algorithm was executed using the entire set of transitions. Figure 4 shows only a limited number of these transitions for ease of presentation and understanding. Both of the weighting matrices in the performance index are taken to be identity. Consider the initial marking of

\[ \gamma_0 = [0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]^T \]  

(22)

which is edge contact on the left-hand side of the hole. We define the final marking to be

\[ \gamma_f = [0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0]^T \]  

(23)

which is an inserted peg on the left-hand side of the hole.

The problem now is, given the initial and final conditions, find a series of transitions that take the system from the initial

Transactions of the ASME
Table 1 Paths which minimize the performance measure

<table>
<thead>
<tr>
<th>Number</th>
<th>Suggested path</th>
<th>Marking sequence</th>
<th>Performance measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path 1</td>
<td>(t₁, t₂, t₃)</td>
<td>(1) (1, 2) (2, 3)</td>
<td>0</td>
</tr>
</tbody>
</table>

to the final markings when these transitions are enabled and executed. An analytical solution to this problem remains elusive because of the integer-valued nature of the transition vector. Therefore, we use dynamic programming. We will begin at the final marking and work our way to the initial marking, examining the performance measure along the way.

The general algorithm is as follows:

1. Determine desired final marking.
2. Using output matrix, Eq. (10), determine which transitions could result in the given marking if fired. Evaluate path length measure.
3. Using the input matrix, Eq. (9), determine which markings are required to fire the transitions found in #2. Evaluate uncertainty measure.
4. Repeat steps #2 and #3 until the desired initial marking is reached. Evaluate the measures at each step, choosing the path of lowest cost.

For the given markings, the dynamic programming algorithm yields the sequence of desired markings and associated performance index given in Table 1. Additionally, Figure 7 shows the discrete event path that was determined using dynamic programming. Path 1 describes the path of minimum performance index in which the desired contacts are gained as soon as possible without losing certainty. Note that the edge-edge contact is considered less certain than a single edge-surface contact for the purposes of planning an event trajectory since the edge-edge contact is a combination of two edge-surface contacts. If the contact is to be maintained, a workpiece in edge-edge contact is only allowed to rotate about the edge in contact. On the other hand, a workpiece in edge-surface contact can both rotate about the point of contact and translate along the surface of contact. This additional direction of motion makes the edge-surface less certain.

Given the desired trajectory and the current marking, it is simple to determine the appropriate B that describes the desired transitions. Those transitions that are included in the discrete event trajectory have control values of ‘one’, while all other place enabled transitions have control values of ‘zero’. Transitions that are not place enabled are immaterial.

3.3 Contingency Performance and Error Recovery. One of the foremost advantages of the Petri net approach to robotic assembly is the ability to detect and recover from unwanted situations. The ability to recover from an unwanted situation increases the reliability of the process and the quality of the product. Due to uncertainties, unexpected, nonoptimal transitions may occur. A common cause of unwanted contact states is the inaccuracy of the robot motion. Such errors may result either from specifying an inaccurate trajectory or from errors of the robot in following the specified trajectory. A second cause is a mismatch or misalignment between the model and the actual physical system. For instance, the optimal velocity commands could be calculated according to a model angle of 30° while the actual angle may be 35°. This mismatch can easily result in a suboptimal event trajectory consisting of undesired contact states.

The Petri net modeling allows for error recovery capabilities since it incorporates all possible contact pairs individually. Each gain or loss of contact can be monitored and recognized individually [10]. Given an unexpected event resulting in a contingent contact state, we can generate an optimal recovery trajectory through the Petri net to the final desired state by applying the same optimization technique described in Section 3. Moreover, the necessary information for determining contingency paths is readily available through steps 2 and 3 of the dynamic programming algorithm. Once an unexpected event occurs, the control can be adjusted to maintain the new state of contact by incorporating the change in constraint indicated by the change in marking, Eq. (16). Lastly, the control can be adjusted to direct the system along its updated path through the net. Figure 8 gives a schematic of how the error recovery system works. Thus, Petri net modeling accomplishes error recovery through detection and identification of the unexpected transition, and then through reconfiguration of the discrete event trajectory and control.

4 Conclusions

This paper established a new understanding of robotic assembly, modeling the process as a discrete event dynamic system. A discrete event in assembly is defined as a discrete change in the state of contact. This new way of approaching the problem used Petri nets to model the assembly system. The abstraction to Petri net modeling highlighted the key components of the synthesis and task-level planning problems. A method for determining a trajectory through the discrete event space using dynamic programming was given. Based on the discrete event trajectory, discrete control variables were derived. Each discrete control variable, from which continuous control commands will be derived, either enables or disables a discrete event.

Several advantages to the discrete event modeling of assembly have been noted. First, it highlights the most significant times in the assembly process when the system becomes dynamic due to a change in the state of contact. It is at these discrete events that the trajectory and control commands need to be modified. Second, discrete event modeling effectively decouples states of contact into their individual control pairs. The decoupling simplifies the monitoring and control of the process. Third, discrete event modeling with Petri nets also proved to be more compact and flexible than traditional contact state modeling, incorporating event causality.

Fig. 7 Minimum cost discrete event trajectory

Journal of Dynamic Systems, Measurement, and Control

Fig. 8 Schematic of contingency performance routine
The modeling of the assembly process as a discrete event system significantly advances the state of the art in robotic assembly. An understanding of the process as dynamic with time varying contacts with the environment has been gained. With this modeling framework several advanced areas are open for study including error detection and recovery, convergence and task-level adaptive control.

References