Disassembly Petri Net Generation in the Presence of XOR Precedence Relationships

K. E. Moore
Information and Decision Systems
ALPHATECH, Inc.
50 Mall Road
Burlington, MA 01803, USA

A. Gungor and S. M. Gupta
Laboratory for Responsible Manufacturing
Dept. of MIME, 334 Snell Engineering
Northeastern University, 360 Huntington Ave.
Boston, MA 02115, USA

ABSTRACT

A disassembly process plan (DPP) is a sequence of disassembly tasks which begins with a product to be disassembled and terminates in a state where all the parts of interest are disconnected. Disassembly process planning is critical for minimizing the resources invested in disassembly and maximizing the level of automation of the disassembly process and the quality of the parts (or materials) recovered. In this paper, we propose an algorithm which automatically generates a disassembly PN (DPN) from a geometrically-based disassembly precedence matrix (DPM). This algorithm can be used to generate DPPs for products which contain simple AND, OR, complex AND/OR, and XOR relationships. The resulting DPN can be analyzed using the reachability tree method to generate all feasible disassembly process plans (DPPs), and cost functions can be used to determine the optimal DPP. An example is used to illustrate the procedure.

1. INTRODUCTION

As a result of regulatory and consumer pressures, there has been an increasing emphasis on environmentally conscious manufacturing (EnviCoM). EnviCoM involves the entire life cycle of products, from conceptual design to final delivery, and ultimately to the end-of-life (EOL) disposal of the products, such that environmental standards and requirements are satisfied. A major element of EOL is product recovery which includes recycling and remanufacturing. Both recycling and remanufacturing involve product disassembly in order to retrieve the desired parts and/or subassemblies.

Disassembly may be defined as a systematic method for separating a product into its constituent parts, components, subassemblies, or other groupings. Disassembly may be partial (some components are not fully disassembled) or complete (the product is fully disassembled). Disassembly process planning is critical in minimizing resources (e.g., time and money) invested in disassembling and maximizing the level of automation of the disassembly process and the quality of the parts (or materials) recovered.

A disassembly process plan is a sequence of disassembly tasks which begins with a product to be disassembled and terminates in a state where all of the parts of interest are disconnected (i.e., it includes partial and complete disassembly). We are interested in generating optimal or near-optimal DPPs, which minimize the cost of disassembly (assuming some level of disassembly is required) or obtain the best cost/benefit ratio for disassembly. In this paper, we present an algorithm that automatically generates feasible DPPs from a geometrically-based DPM. The latter can be generated from a CAD representation of the product.

In the next section, we present a review of the literature on disassembly process planning. In Section 3, we describe our technical approach and present an example to illustrate the methodology. In Section 4, we present the algorithm for generating the DPN for products with XOR precedence relationships. Section 5 summarizes the work presented here.

2. LITERATURE REVIEW

Disassembly process planning is a new area of research and there is a relatively small number of papers. One of the first known papers was published in 1991 and uses a branch-and-bound approach to minimize total disassembly cost [1]. At each stage in the disassembly, the algorithm selects a part to disassemble which has the lowest total disassembly cost. This approach generates the optimal DPP when costs are constant. Several papers utilize AND/OR graphs. For example, [2] generates the AND/OR graph from movement and interference matrices; the AND/OR graph is then analyzed to generate all feasible DPPs. References [3-5] generate disassembly process graphs (DPGs) from AND/OR graphs. The DPG incorporates cost and revenue data, which are used to stop DPG generation when further disassembly is no
longer profitable. The major drawback to this approach is that it is exhaustive. A graph theoretical approach is proposed in [6]. The graph is based on the geometric information for the product and is analyzed to generate all feasible disassembly steps. The resulting tree can be used to identify an efficient, though not necessarily optimal, sequence of steps to disassemble a specific part.

PNs have been shown to be very useful in assembly process planning [7]. To date, however, very little has been done to apply PNs to disassembly. In the first paper to use PNs in disassembly, [8] propose a method for generating PNs from a series of precedence tables, similar to the method used in [9]. The resulting PN is analyzed using the reachability method to generate all feasible DPPs. No optimization is done and the approach is exhaustive.

One of the limitations present in all of the above papers is that the disassembly precedence relationships are limited to simple AND and simple OR relationships; none contain complex AND/OR or XOR relationships. An AND relationship exists between components c_1 and c_2 in relation to c_3, if both c_1 and c_2 must be removed prior to c_3. An OR relationship exists between parts c_1 and c_2 in relation to c_3, if either c_1 or c_2 must be removed prior to c_3. A complex AND/OR relationship exists between parts c_1, c_2, and c_3, in relation to c_4, if c_1 along with either c_2 or c_3 must be removed prior to c_4. An XOR relationship exists between parts c_1, c_2, and c_3, in relation to c_4, if c_1 along with c_2 or c_3, but not both, must be removed prior to c_4 (i.e., if c_1 and c_2 are removed, then c_3 must remain until c_4 is removed and, likewise, if c_1 and c_3 are removed, then c_2 must remain until c_4 is removed). [10, 11] are the only papers to address the complex AND/OR relationship. In this paper, we extend the work of [10, 11] to XOR relationships.

**APPROACH**

In our research, we are developing a method based on PNs to generate near-optimal DPPs. The methodology involves the following steps: 1) Analyze the product to generate a disassembly precedence matrix (DPM) representing the physically-based disassembly constraints [12]; (2) Generate the disassembly PN (PN) from the DPM; and (3) Generate near optimal DPPs from the DPN. In this paper, we describe the algorithm for generating a near-optimal DPP. We have successfully applied this approach to the case of complex AND/ORs [10, 11].

Throughout this paper, we use the product shown in Fig. 1 to illustrate our approach. The example product consists of seven elements (five parts and two joining elements). As shown, part 1 is attached to the fixture (the shaded base and clamps). The XOR relationship for this product exists between parts 3, 4, and 5. Either part 3 or part 4 (but not both) must be removed prior to part 5; part 5 cannot be left hanging in the air. For this discussion, we only consider movement in two dimensions; however, this approach can be extended to the three-dimensional case without loss of generality. Movement can be in direction d, d = {x, -x, y, -y}.

**Disassembly Precedence Matrix**

The DPM represents the geometrically-based precedence relationships between the components of the product. We recognize the following types of precedence relationships: AND, OR, complex AND/OR, and XOR. Due to the nature of the geometric constraints, groups of OR constraints are in the same direction, d. We can now define the DPM, B = [b_{ij}], i, j = 1, ..., k (k is the number of parts) as:

\[
b_{ij} = \begin{cases} 
1, & \text{part } i \text{ AND precedes part } j \\
0, & \text{otherwise} \\
-d, & \text{part } i \text{ OR precedes part } j \\
-1, & \text{part } i \text{ XOR precedes part } j 
\end{cases}
\]  

(1)

The DPM can be generated automatically from a CAD representation of the product [12]. The DPM for the product appears in Fig. 2.

**Petri Nets**

Petri nets (PNs) are a graphical and mathematical technique useful for modeling concurrent, asynchronous, distributed, parallel, nondeterministic, and stochastic systems. PN models can be analyzed to determine both their qualitative and quantitative properties. PNs have recently emerged as a promising approach for modeling manufacturing systems, and have been used for assembly process planning [7]. As discussed above, very little has been done to apply PNs to disassembly.

![Fig. 1. Sample Product.](image-url)
is a logical transition representing the completion of the disassembly. Additional place and transition notation will be introduced as necessary.

Observations on the Disassembly Precedence Matrix

We begin by making several observations about the DPM. These observations are explained below.

(H1) No Precedents: Column \( j \) contains only zeros (\( B_j = 0 \)). Component \( c_j \) has no predecessors; the presence of a product to disassemble is a sufficient condition for the removal of \( c_j \). Hence, \( p_j \) is an output of \( t_0 \) and the sole input to \( t_j \). The related arc weights are unity: \( w(t_0, t_j) = w(t_0, p_j) = w(p_j, t_j) = 1 \forall t_j \in B_j = 0 \).

(H2) No Antecedents: Row \( i \) contains only zeros (\( B_i = 0 \)). The removal of \( c_i \) is not precedent to the removal of any other component; however, it is a precondition of complete disassembly. Hence, \( t_i \) is an input to \( p_{ni} \), \( w(t_i, p_i) = 1 \), and \( w(p_{ni}, t_i) \) is the number of zero rows in \( B \).

(H3) AND Precedents: \( B_j \) contains one or more 1s (\( b_{ij} = 1 \), for some \( i \)). The removal of \( c_j \) precedes the removal of \( c_i \). This is an AND precedent, i.e., the removal of all \( c_j \) if \( b_{ij} = 1 \) is precedent to the removal of \( c_j \). The arc weights are \( w(t_i, p_j) = 1 \) and \( w(p_j, t_i) = \) the number of 1s in \( B_j \).

(H4) OR Precedents: \( B_j \) contains \( ds \) where no two \( ds \) are the same (\( \forall b_{ij}, b_{ij} \in D, b_{ij} \neq b_{ij} \)). The removal of \( c_j \) (\( b_{ij} = d \)) is OR precedent in direction \( d \) to the removal of \( c_i \). At least one \( c_i \) (\( b_{ij} \in D \)) must be removed prior to removing \( c_j \). Let \( p_{ij} \) represent the set of OR conditions; \( w(t_i, p_{ij}) = 1 \). Since only one OR condition must be met to disassemble \( c_j \), \( w(p_{ij}, t_i) = 1 \). To ensure that only one OR condition fires \( t_i \), let \( p_j \) be a place with an input arc from \( t_i \) and an output arc to \( t_j \) where \( w(p_j, t_i) = w(p_{ij}, t_i) = 1 \). Since the remaining OR conditions are required for complete disassembly, introduce an arc from \( p_{ij} \) to \( t_i \) where \( w(p_{ij}, t_i) = 1 \) less than the total number of OR precedent conditions to \( c_j \).

(H5) AND Within OR Precedents: \( B_j \) contains \( ds \) where at least two \( ds \) are equal (\( 3 b_{ij}, b_{ij} \in D \mid b_{ij} = b_{ij} \) and \( h \neq i \)). The removal of a set of components \( \{c_i, b_{ij} = d\} \) is OR precedent in direction \( d \) to the removal of \( c_j \); i.e., all \( c_i \)'s with the same \( d \) w.r.t. \( c_j \) must be removed to satisfy a single OR condition. This is an AND precedence within an OR precedence group. This case combines (H3) and (H4). Place \( p_{aj} \) represents the AND conditions within the \( a^d \) OR precedent group for \( c_j \); a logical transition \( t_{aj} \) represents completion of the set of AND conditions within the \( a^d \) OR precedent group for \( c_j \); and \( w(t_i, p_{aj}) = 1 \) and \( w(p_{aj}, t_{aj}) = \) the number of \( \text{AND conditions within the} a^d \text{OR precedent group for} c_j \). From (H4), we include \( p_{aj} \) and \( p_j \) where \( w(t_{aj},
\( p_{o_{j,d}} = w(t_o, p_j) = w(p_j, t_j) = 1 \) and \( w(p_j, t_d) \) one less than the total number of OR precedent groups to \( c_j \).

(H6) Complex AND/OR Precedents: \( B_j \) contains \( ds \) and 1s \( (B_j = \{ b_{ij} \mid b_{ij} = \{ 0, 1, d \} \}) \). Removal of the set of components \( \{ c_i \mid b_{ij} = 1 \} \) along with at least one of the sets of components \( \{ c_h \mid b_{hi} = d \} \) is precedent to the removal of \( c_j \). This represents the complex AND/OR precedence relationship. We simply replace \( p_{c_i} \) and its input and output arcs with \( p_{j} \), where \( p_j \) represents the AND precedent conditions for removing \( c_j \). The definition of \( p_j \), its input transition(s), and input and output arc weights are the same as for (H3). (H6) combines (H3), (H4), and (H5).

(H7) XOR Precedents: \( B_j \) contains -1s \((B_j = \{ b_{ij} \mid b_{ij} = \{ 0, -1 \} \})\). Removal of one and only one \( c_i \) \((b_{ij} = -1)\) is precedent to the removal of \( c_j \). Since only one \( c_i \) \((b_{ij} = -1)\) can be removed prior to \( c_j \), we need to prevent any other \( c_i \) \((b_{ij} = -1, i \neq k)\) from being removed prior to \( c_j \). Let \( p_{c_i} \) represent the control for the XOR precedents; \( w(p_{c_i}, t_j) = 1 \) \( \forall i \) \((b_{ij} = -1)\). To establish the initial control, we introduce an arc from \( t_o \) to \( p_{c_i} \) where \( w(t_o, p_{c_i}) = 1 \); to allow the remaining XOR precedent parts to be disassembled, we introduce an arc from \( t_o \) to \( p_{c_i} \) where \( w(t_o, p_{c_i}) \) = the number of XOR conditions to \( j \) less one. Since we do not know which XOR condition will be met first and since the remaining XOR conditions are required for complete disassembly, we introduce \( p_x \) and use it in a manner analogous to \( p_o \); i.e., \( w(p_x, t_j) = 1 \), \( w(t_o, p_x) = 1 \) \( \forall i \) \((b_{ij} = -1)\), and \( w(p_x, t_j) = one less than the total number of XOR precedent conditions to \( c_j \).

Algorithm to Generate DPN with XORs

Let \( A_G \) be the set of AND precedents to \( j \), \( O_j \) be the set of directions for which there exist OR precedence group to \( j \). \( O_{G,j} \) be the OR precedent group to \( j \) in direction \( d \), \( X_G \) be the set of components XOR precedent to \( j \), and \( NZ \) a vector of binary variables indicating whether \( B_i \) contains non-zero entries.

\[
NZ_j = \begin{cases} 
1, & B_i = 0 \\
0, & \text{otherwise}
\end{cases}
\]

The algorithm to generate the DPN \( A \) appears in Fig. 3. Fig. 4 shows the DPN generated by this algorithm for the sample product.

Step 1: Initialize Variables.

\[
P = \{ p_i, p_j, p_k \}, \ T = \{ t_o, t_i, t_j \}, i, j = 1 \to k; \ A = (k + 2) \times (k + 2) \text{ matrix (initial size),}
\]

\[
A_G = O_j = O_{G,j} = X_G = \{ \emptyset \}, j = 1 \to k, d = \{ D \}, 
\]

\[
nz_j = 1, i = 1 \to k.
\]

Step 2: Complete \( T, P, \) and \( A \).

Step 2.1: Scan \( B_j \) for places associated with AND, OR, and XOR precedence groups.

If \( b_{ij} = 1 \), add \( i \) to \( A_G \), set \( nz_i = 0 \).

If \( b_{ij} = d \), add \( i \) to \( O_{G,j} \), add \( d \) to \( O_j \),

add \( p_{o,j} \) to \( P \). set \( nz_j = 0 \).

If \( b_{ij} = -1 \), add \( i \) to \( X_G \), add \( p_x \) to \( P \),

set \( nz_j = 0 \).

Step 2.2: Generate arcs in \( A \) for places and transitions associated with \( B_j \).

Step 2.2.1: Examine \( A_G \).

If \( |A_G| = 0 \), \( a(p_o, t_i) = 1 \)

If \( |A_G| > 0 \), \( a(p_o, t_i) = -|A_G|, \)

\( a(p_o, t_i) = 1 \) for \( i \in A_G \).

Step 2.2.2: Examine \( O_{G,j} \) and \( O_j \).

If \( |O_j| > 0 \), \( a(p_{o,j}, t_i) = 1 \)

\( a(p_{o,j}, t_i) = |O_j| - 1 \).

If \( |O_{G,j}| = 1 \), for \( d \in O_j \),

\( a(p_{o,j}, t_i) = 1 \) for \( i \in O_{G,j} \).

If \( |O_{G,j}| > 1 \), for \( d \in O_j \),

\( T = (T \cup t_{a,j}), P = (P \cup p_{a,j} \cup p_{o,j}) \)

\( a(p_{o,j}, t_i) = 1 \), \( a(p_{a,j}, t_a,j) = |O_{G,j}| - 1 \)

\( a(p_{o,j}, t_i) = 1 \) for \( i \in X_G \), \( a(p_{x,j}, t_i) = 1 \) for \( i \in X_G \)

\( a(p_{x,j}, t_i) = 1 \), \( a(p_{x,j}, t_i) = -|X_G| - 1 \).

Step 2.2.3: Examine \( X_G \).

If \( |X_G| > 0 \), \( P = (P \cup p_x \cup p_x) \)

\( a(p_{x,j}, t_i) = 1 \), \( a(p_{x,j}, t_i) = |X_G| - 1 \)

\( a(p_{x,j}, t_i) = 1 \) for \( i \in X_G \), \( a(p_{x,j}, t_i) = 1 \) for \( i \in X_G \)

\( a(p_{x,j}, t_i) = 1 \), \( a(p_{x,j}, t_i) = -|X_G| - 1 \).

Step 3: Finalize \( A \).

Step 3.1: Scan \( NZ \) to generate arcs for parts with no antecedents.

\( a(p_{o}, t_i) = nz_i \).

Step 3.2: Sum \( NZ \) to generate arcs for final place.

If \( \sum_{i=1}^{k} nz_i > 0 \), \( a(p_{o}, t_i) = -\sum_{i=1}^{k} nz_i \)

If \( \sum_{i=1}^{k} nz_i = 0 \), \( a(p_{o}, t_i) = 1 \).

Fig. 3. Algorithm for Generating DPN with XOR.
Properties of the DPN

With a small number of restrictions, this algorithm guarantees that the resulting DPN is live, bounded, and reversible. These restrictions include:

- In complex AND/OR cases, a part cannot be OR precedent in more than one direction.
- We do not consider direction in the case of XOR.
- We assume that there are a finite number of parts, that the parts can be removed without destructive disassembly, parts are removed individually, and parts are removed in a single direction.

The proof of these properties, which is too long for inclusion here, is based on the work of [13, 14]. The proof begins with a set of submodels that capture the observations made on the DPN. Each submodel is supplemented by $p_b$, $t_b$, $p_b$, and $t_b$, resulting in a live, bounded, and reversible DPN submodel. The resulting models are merged via shared paths, and then reduced by deleting redundant paths. The rules for merging and reduction preserve the desired behavior. This is simply a graphical version of the algorithm presented in Fig. 3. Further, the DPN is constructed such that every transition in the net must fire, before the final transition can fire.

The DPN can be analyzed using the reachability tree method [15] to generate feasible DPPs. The algorithm for generating the DPN guarantees that the reachability tree contains only feasible DPPs. Once the DPPs are identified, cost functions can be applied to determine the best DPP.

**SUMMARY**

In this paper, we presented a methodology based on Petri nets (PNs) which can be used to generate disassembly process plans (DPPs). We presented an algorithm to automatically generate geometric-based disassembly precedence matrix (DPM). The product may contain AND, OR, complex AND/OR, and XOR precedence relationships. To our knowledge, this is the first paper which addresses the XOR case.

The DPN is constructed in such a way as to guarantee that it will generate only feasible DPPs, using the reachability tree method. Once the set of all feasible DPPs is generated, cost functions can be applied to determine the optimal DPP. Since the reachability tree method is NP-
complete, for complex products, a heuristic approach can be applied to generate near-optimal DPPs.

REFERENCES


