An Integrated Approach to Disassembly Planning and Demanufacturing Operation

Ying Tang, MengChu Zhou, Senior Member, IEEE, and Reggie J. Caudill

Abstract—Industrial demanufacturing is a practice of growing importance due to increasing environmental and economic pressures. However, very little research focuses on it from a system perspective. This paper presents a disassembly planning and demanufacturing scheduling method for an integrated flexible demanufacturing system. Workstation Petri net and Product Petri net are proposed for its hierarchical and modular modeling in order to derive the disassembly path with the maximal end-of-life value. Scheduling Petri net is introduced to schedule the demanufacturing resources. The proposed methodology and algorithms are demonstrated through the disassembly of personal computers in an integrated flexible demanufacturing system.

Index Terms—Demanufacturing systems, disassembly planning, Petri nets.

NOMENCLATURE

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DP</td>
<td>Disassembly path.</td>
</tr>
<tr>
<td>DPN</td>
<td>Disassembly Petri net.</td>
</tr>
<tr>
<td>EOL</td>
<td>End-of-life.</td>
</tr>
<tr>
<td>IFDS</td>
<td>Integrated flexible demanufacturing system.</td>
</tr>
<tr>
<td>PC</td>
<td>Personal computer.</td>
</tr>
<tr>
<td>PN</td>
<td>Petri net.</td>
</tr>
<tr>
<td>PPN</td>
<td>Product Petri net.</td>
</tr>
<tr>
<td>SPN</td>
<td>Scheduling Petri net.</td>
</tr>
<tr>
<td>WPN</td>
<td>Workstation Petri net.</td>
</tr>
</tbody>
</table>

I. INTRODUCTION

Due to growing concerns for material resources, energy conservation, and landfill capacity, the electronic industry has been pressured to design and manufacture environmentally benign products and demanufacture the obsolete products to maximize their end-of-life (EOL) value. Recent estimates indicate that the amount of obsolete and malfunctioning information technology equipment is expected to grow exponentially. Thus, understanding and developing techniques for EOL management of the products by means of product/material recovery are extremely crucial [19]. As an integral part of a product life cycle, demanufacturing has been introduced which is a process of disassembling products and then reusing, remanufacturing, reengineering, or disposing of them [17]. The collection of the used items and/or their packages is one of the major issues in a product recovery environment. Flow of used products back into the product demanufacturing facility is known as reverse distribution [10]. Disassembly also gains much attention in product recovery, and different methodologies to represent and find disassembly sequences have been developed. Homen de Mello and Sanderson proposed an AND/OR graph to present all feasible disassembly sequences and a heuristic-search algorithm to derive the optimal disassembly process plan [7], [8]. Based on the algebraic and graph structural properties of disassembly Petri net (DPN), the optimal disassembly process plan can be solved by the linear programming technique [18]. Considering the variations of product conditions, which in turn cause operational failure in disassembly and different EOL values imbedded in a product, a subassembly, and a part, Zussman et al. proposed a DPN linked to a Bayesian network method to provide planning algorithms to determine the desired disassembly level [24]. Zussman and Zhou also derived several planning algorithms and developed a DPN-based disassembly process planner and robotic system [25], [26]. Gupta and Taleb presented an algorithm that reverses the material requirement planning procedure and can be applied to the disassembly planning of some products where the demand is on the component level of the product structure [5]. Current demanufacturing practices include manual and automatic disassembly of discarded products. Kiril et al. proposed the dismantling system for refrigerator [9]. Kopacek et al. described a semi-automatic disassembly cell for PCs [11]. Tzafestas et al. presented the architectural and implementation issues of an autonomous car disassembly robotics system [21]. Grenchus gave a brief history and overview of the IBM demanufacturing process [4]. Lee and Bailey-Van Kuren presented a modeling and supervisory control framework of a disassembly workcell and its application to single-use cameras and PCs [12]. A recent survey of disassembly modeling and planning methodologies is presented in [22].

Although progress has been reported on the above research areas, very little effort has been focused on demanufacturing from a system perspective. In fact, the allocation of resources in a demanufacturing system is very critical to efficient operations. Many methods have been proposed for the planning of manufacturing systems. Held and Karp used a dynamical programming approach to solve the scheduling problem [6]. A heuristic algorithm was proposed in [3] to generate satisfactory schedules. Stochastic methods were presented as a promising approach to these problems [15]. Luh et al. used Lagrange multipliers to...
relax capacity and precedence constraints and obtained near-optimal solutions for single-machine and parallel-machine planning [13]. Thomas formulated a unified representation scheme to capture the assembly plan and the control level [20]. However, products for demanufacturing exhibit high uncertainty in system structures and component conditions. Furthermore, their disassembly termination goals are not necessarily fixed [22], [25], [26]. Due to many physical and operational differences between disassembly and assembly [2], this paper addresses the fundamental issues in design and operation of an integrated flexible demanufacturing system (IFDS). Moreover, the EOL value of products and the capacity of systems are considered, which makes our methodology more realistic. The rest of this paper is organized as follows. Section II describes the generic model for an IFDS. Section III focuses on disassembly modeling for products and resources using PN. Section IV presents the disassembly process planning. Section V gives a case study, and Section VI gives conclusions.

II. GENERIC MODE FOR INTEGRATED FLEXIBLE DEMANUFACTURING SYSTEMS

Demanufacturing is becoming an integral part of a product life cycle. It performs a set of functions (such as inspection and disassembly) to recover values from products and waste streams, and ships these recovered materials and components for reuse, recycling, reengineering, and remanufacturing. An IFDS, as shown in Fig. 1, consists of the following interacting and cooperating units:

- database;
- collection unit;
- inspection unit;
- disassembly workstation;
- sorting unit;
- sensory unit;
- supervision unit.

A. Database

The database is the information center of an IFDS. It consists of three parts: product database that stores product model, constitution, and characteristic; disassembly method database that has product disassembly sequences and treatment methods for products under faulty conditions; and resource database that keeps the knowledge of available machines, robots, and human operators.
B. Collection Unit

Discarded products are transported to the demanufacturer from various sources, which introduce uncontrolled variability in terms of product types and conditions. Different models may need different disassembly facilities, and batches of products of same disassembly families provide faster disassembly and lower setup and material handling time by keeping the same tooling and workstation setup. Considering these, an IFDS performs a preliminary screening at the collection unit to separate coming products according to their models and send them to different storage areas. In other words, the group technology concept is used to improve the system efficiency [14], [15].

C. Inspection Unit

Before disassembly, batches of products of the same model are tested for their potential reusability in primary and secondary inspection areas. In the primary inspection area, a product Petri net (PPN) with all feasible disassembly sequences and a workstation Petri net (WPN) with workstation information are introduced. During the first inspection, the system obtains information of products, such as working condition, model and constitution, and determines the disassembly path (DP) with the EOL value based on the real-time machine availability. Products in very poor condition are immediately sent for disposal (e.g., smelting, shredding, and landfill). In the secondary inspection area, a scheduling Petri net (SPN) is proposed. With the fixed DP and resource information, the system will continue to optimize the disassembly operations to obtain the maximum operational efficiency of demanufacturing facilities. The concepts and algorithms based on WPN, PPN, and SPN are discussed in Sections III and IV.

D. Disassembly Workstation

According to the DP determined in primary inspection, a product is sent to certain disassembly workstations. In an example IFDS, there are four workstations that fulfill different disassembly tasks for a PC. Workstation 1 separates case and cables from a computer product. Workstation 2 deals with the disassembly of disk drive, hard drive, and CD-ROM. Workstation 3 performs the disassembly of display adapter, audio card, and network adapter. Workstation 4 accomplishes the disassembly of power supply units. Due to the complexity of disassembly tasks, workstations are equipped with different machines and robots.

E. Sorting Unit

The sort operation segregates the disassembled components and materials into the proper commodities. Commodity containers are weighed and information on commodity type and container’s weight is sent back to the sorting controller. After that, different commodities are shipped to different vendors for reuse, remanufacturing, reengineering or disposal.

F. Sensory Unit

The sensory unit monitors the status of machines and buffers and feedback information to supervision unit. It plays an important role in decision making and maintaining excellent system performance.

G. Supervision Unit

The supervision unit consists of workstation controllers, robot controllers, sorting controllers and super-controller. Workstation controllers are responsible for controlling the operations in the corresponding machines and buffers. Robot controllers extract the optimal routes for robot movements. Once malfunction or exception happens, such as breakdown of a machine and container overflow, these controllers evaluate the errors and choose the appropriate tactics to handle them. Sorting controllers check signals from the sorting unit and appropriately remove the filled container and replace it with a new one. The super-controller supervises all the controllers. Its functions include job release, task planning and assignment, coordination, communication, and synchronization.

The rest of this paper focuses on disassembly process modeling and dynamic routing in inspection unit and disassembly workstation.

III. MODELING DISASSEMBLY PROCESSES

One of the key purposes of dealing with discarded products is to maximize recycled resources with the consideration of ecological, environmental and economic factors. Zussman et al. termed the multipurpose goal as “increasing the EOL value” [24], [25]. Since each disassembly task is processed at a certain workstation, the realizability of a disassembly task is affected by the availability of machines in its corresponding workstation. Moreover, a workstation has finite machines with different speeds. The best machine assignment is thus a key to high system throughput. In an IFDS, not only are the EOL value and the capacity of each workstation considered, but also the efficient machine assignment is emphasized. Considering these, this work proposes the following three PN models:

1) a product PN (PPN) with all feasible disassembly sequences and the EOL value of products, subassemblies, and components;
2) a workstation PN (WPN) to model the status of workstations and provide the availability information of workstations;
3) a scheduling PN (SPN) for machine scheduling.

A. Petri Nets (PNs)

Petri nets, as a graphical and mathematical tool, provide a uniform environment for modeling, formal analysis, and design of discrete event systems [23]. A PN may be identified as a particular kind of bipartite directed graph populated by three types of objects. They are places, transitions, and directed arcs connecting places to transitions and transitions to places. Pictorially, places are depicted by circles and transitions by bars. A place is an input (output) place to a transition if there exists a directed arc connecting the transition (place) to the place (transition). In order to study the dynamic behavior of the modeled system, each place may potentially hold a nonnegative number of tokens, pictured by small solid dots. Formally, a PN can be defined as follows:
Fig. 2. A simple example of a WPN with four workstations.

**Definition 3.1:** A PN is defined as a five-tuple: 
\[ P N = (P, T, I, O, M) \]

1. \( P = \{ p_1, p_2, \ldots, p_n \} \) is a finite set of places.
2. \( T = \{ t_1, t_2, \ldots, t_m \} \) is a finite set of transitions. \( P \cup T \neq \emptyset, P \cap T = \emptyset. \)
3. \( I: P \times T \to \{ 0, 1 \} \) is an input function that defines the set of directed arcs from \( P \) to \( T \), where \( I_{ij} = 1 \) if \( p_i \) is an input place of \( t_j \); otherwise 0.
4. \( O: P \times T \to \{ 0, 1 \} \) is an output function that defines the set of directed arcs from \( T \) to \( P \), where \( O_{ij} = 1 \) if \( p_i \) is an output place of \( t_j \); otherwise 0.
5. \( M: \{ 0, 1, 2, \ldots \} \) is a marking vector whose \( i \)th component represents the number of tokens in the \( i \)th place. An initial marking is denoted by \( m_0 \).

The defined PN called ordinary PN can be further extended to represent various aspects of modeled systems. For instance, places may represent the status of resources. Time and cost can be introduced into ordinary PNs to analyze system performance. This paper uses and extends the ordinary PN to model workstation status, product disassembly sequences, and scheduling.

### B. Workstation Petri Net (WPN)

**Definition 3.2:** A workstation PN (WPN) is a PN where:

1. \( P = W^a \cup W^b = \{ w_{11}, w_{12}, \ldots, w_{1n} \} \cup \{ w_{21}, w_{22}, \ldots, w_{2n} \} \), where the set of places \( W^a \) represents the available machine pools and the set of places \( W^b \) stands for the busy machine pools in certain workstations;
2. \( T = T^b \cup T^e = \{ t_{1b}, t_{1e}, \ldots, t_{mb}, t_{me} \} \), where the set of transitions \( T^b \) (\( T^e \)) represents the beginning (end) of operations processed in workstations;
3. \( I = W^a \cdot I^T = \begin{pmatrix} t_{1b} & t_{1e} & t_{2b} & t_{2e} & \cdots & t_{mb} & t_{me} \\ W^a_1 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ W^a_2 & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdot & \vdots & \vdots \\ W^a_n & 0 & 0 & 0 & 0 & \cdots & 1 & 0 \\ W^b_n & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \)

is a \( 2n \times 2n \) identity matrix where each diagonal number means that each place in \( W^a \) has a directed arc to its corresponding transition in \( T^e \).

An example is given in Fig. 2 where four workstations are modeled. The present markings show that both machines are busy at Workstation 1 and idle at Workstations 2–4.

### C. Product Petri Net (PPN)

**Definition 3.3:** A Product PN (PPN) is an eight-tuple extended PN

\[ PPN = (P, T, I, O, M, \pi, h, r, \lambda, \delta) \]

where

- \( P = W \cup Q \), where the set of places \( W = \{ w_1, w_2, \ldots, w_n \} \) represents the available machine pools in certain workstations, whose information comes from the WPN. The set of places \( Q \) stands for product, subassemblies, or components.
- \( \exists p \in Q \) such that \( p = \Phi \): This place is usually denoted by \( p_1 \) and named as root or product place.
- \( \exists Q^r \subseteq Q \) such that \( Q^r \neq \Phi \) and \( \forall p \in Q^r, p^* = \Phi \).
- \( \forall p \in Q^r \) are called leaves.
- \( I(p, t) = O(p, t), \forall t \in T. \forall p \in W \).
- \( m_0(p) = 1 \) and \( m_0(p) = 0, \forall p \in Q^r \).
- Final marking \( m_f(p) = m_0(p), \forall p \in W \), and their values come from the WPN.
- \( \pi: Q \to R \) is an EOL value function assigned to a place, which is defined as \( \pi(p) = \max\{\pi_{\text{reuse}}(p), \pi_{\text{remanufacturing}}(p), \pi_{\text{reengineering}}(p), \pi_{\text{landfill}}(p)\} \), where \( \pi_{\text{reuse}}(p), \pi_{\text{remanufacturing}}(p), \pi_{\text{reengineering}}(p) \), and \( \pi_{\text{landfill}}(p) \) are the EOL values when the product, subassembly, or component represented by \( p \) is reused, remanufactured, reengineered, or landfilled [24].
- \( h: T \to R \) is a cost value function associated with transitions, which mainly depends on the accessibility and difficulty of disassembly.
• $r_i: W \rightarrow R^+$ is the time delay function associated with each workstation-available place. $r_i = 0$ when there is at least one machine available in the $i$th workstation; otherwise, $r_i$ is the waiting time for the first available machine in the $i$th workstation.

• $\lambda: T \rightarrow R$ is a conservative time function associated with a transition. It is the maximum time needed to finish the operation a transition represents.

• $\delta: T \rightarrow N$ is a decision value associated with a transition. This value is decided before the corresponding disassembly operation, or firing of the transition.

Condition 3) of a PPN assures that the token in a workstation to enable certain transitions is released after their firing. Furthermore, for reasons to be explained during the discussion of the Planning Algorithm (cf. Algorithm 4.1), in the rest of the paper we restrict our attention to the PPN (sub-) class where every transition has only one nonleaf output place, i.e., to the PPNs representing disassembly plans with no parallel tasks. Also, for the same reasons, it is assumed that no workstation is visited more than once in any disassembly plan, represented by a PPN path from the root to a leaf place. Fig. 3 gives a simple example of PPN for a PC disassembly, where letters a to k represent the components of a PC as listed in Table I.

**Table I**

**Component List for a PC**

<table>
<thead>
<tr>
<th># of products</th>
<th>Baseline method</th>
<th>Proposed method</th>
<th>Observed Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1.319</td>
<td>1.391</td>
<td>-0.072</td>
</tr>
<tr>
<td>500</td>
<td>1.333</td>
<td>1.429</td>
<td>-0.076</td>
</tr>
<tr>
<td>1000</td>
<td>1.308</td>
<td>1.389</td>
<td>-0.081</td>
</tr>
<tr>
<td>2000</td>
<td>1.342</td>
<td>1.420</td>
<td>-0.078</td>
</tr>
<tr>
<td>5000</td>
<td>1.336</td>
<td>1.417</td>
<td>-0.081</td>
</tr>
<tr>
<td>10000</td>
<td>1.323</td>
<td>1.405</td>
<td>-0.082</td>
</tr>
<tr>
<td>Sample mean</td>
<td>1.330</td>
<td>1.4085</td>
<td>-0.078</td>
</tr>
<tr>
<td>Sample variance</td>
<td>0.0008</td>
<td>0.0003</td>
<td>0.0002</td>
</tr>
</tbody>
</table>

**D. Scheduling Petri Net (SPN)**

This paper makes the following assumptions for an IFDS.

1) All the machines in a workstation have the same functions, but may have different speeds.
2) Each machine can process at most one task at a time.
3) Each task is nonpreemptive, requiring one and only one machine at a time.
4) Setup time is included in process time, and two same operations in sequence need no setup time.
5) The transportation time between workstations is negligible compared with the disassembly time.
Due to the uncertainty of disassembly, used products with the same model probably require different disassembly methods. Thus, the entire model is decomposed into several subnets according to jobs and workstations, which are stated as follows.

• The disassembly of a discarded product is called a job. \( n \) jobs to be processed are denoted by \( J = \{ J_1, J_2, \ldots, J_n \} \).
• Each job has a sequence of tasks. An operation is to process a task by one machine in a workstation. \( O_{ijkm} \) represents the \( j \)th task of the \( i \)th job being performed by the \( m \)th machine in the \( k \)th workstation.

A dynamic model for each workstation is denoted by scheduling PN (SPN). \( SPN_{ik} \) represents that the \( i \)th job being processed in the \( k \)th workstation.

**Definition 3.4:** A Scheduling PN \( SPN_{ik} \) is defined as a seven-tuple PN:

\[
SPN_{ik} = (P, T, I, O, M, \Gamma, \Lambda)
\]

where

- \( P = R \cup S \) where the set of resource places \( R \) represents the machines and the set of status places \( S \) corresponds to job status.
  1) \( \exists p \in S, \bullet p = \Phi \). This place is usually denoted by \( P_{\text{initial}} \) or \( P_1 \), which represents the initial status for the \( i \)th job in the \( k \)th workstation.
  2) \( \exists p \in S, \bullet p = \Phi \). This place is usually denoted by \( P_{\text{final}} \) corresponding to the final status for the \( i \)th job in the \( k \)th workstation.
- \( \forall t \in T \), there is one and only one resource place \( p \in R \) and \( I(p, t) = O(p, t) \).
- \( \forall t \in T \), there is one and only one pair of status places \( p_1, p_2 \), \( S \) such that \( t \in \bullet p_1 \), and \( t \in \bullet p_2 \).
- Initial marking \( m_0(p) = 1, \forall p \in R \cup \{ P_{\text{initial}} \} \) and \( m_0(p) = 0, \forall p \in S - \{ P_{\text{initial}} \} \).
- Final marking \( m_f(p) = 1, \forall p \in R \cup \{ P_{\text{final}} \} \), and \( m_f(p) = 0, \forall p \in S - \{ P_{\text{final}} \} \).
- \( \Gamma: T \rightarrow R^+ \) is a processing time function associated with each transition.
- \( \Lambda: R \rightarrow R^+ \) is a delay time function associated with a resource place, which represents the waiting time for the resource to handle a disassembly process of the current subassembly. It is updated according to the sensing and execution results of the corresponding disassembly operation performed by the supervision unit.

Fig. 4 and Table II give an example \( SPN_{31} \), which shows the PN model for the third job in the first workstation. For convenience, places and transitions are also numbered into sequences as shown on their left sides.

These extended PN models are used to facilitate the disassembly planning and machine scheduling during the primary and the secondary inspection, which we elaborate on in Section IV.

### IV. DISASSEMBLY PROCESS PLANNING AND MACHINE SCHEDULING

Zussman and Zhou define the disassembly process plan as finding the order of disassembly operations with the maximal EOL value [25], [26]. In an IFDS, both the EOL value and resource scheduling are considered, which can be implemented by two-level planning in primary and secondary inspection units. Fig. 5 summarizes the overall logic for this approach.

#### A. Planning Algorithm

In an IFDS, workstations perform disassembly tasks. In a PPN, a workstation place is introduced as an input to a transition, which means that a transition cannot fire until there is at least one machine available in the corresponding workstation to handle it. Considering the capacity of each workstation, not only are tokens used to represent the number of machines \( M(w_i) \) in the \( i \)th workstation, but also a release time \( r \) is assigned to each \( w_i \). During initialization, \( r \) is set to zero, which means that there is at least one machine available in each workstation. During operation, the workstation controller monitors the status of workstations and tracks their tokens. Once a used product is released into the facility, the workstation controller checks the current status of WPN. If all machines in the \( i \)th workstation are occupied, i.e., all tokens in \( w_i \) go to \( Q \) in WPN, workstation controller calculates the release time for the first available machine and informs the Primary Inspection Unit to update the corresponding \( r \). This \( r \)-value together with other defined function in Definition 3.3 is used to decide the disassembly plan.

In a PPN, a place \( p \in Q \) with multiple output transitions means that there are various disassembly methods available. To choose the best one, disassembly values of a place and a transition have to be defined. Due to the different units for EOL value ($$), cost value ($) and time delay (hour), the unit exchange is needed.

**Definition 4.1:** The cost value for the time delay caused by a workstation \( w \) is denoted as \( \sigma(w) \), and

\[
\sigma(w) = r(w) \times R
\]  

where \( r(w) \) is the time delay for workstation \( w \) and \( R \) is the exchange rate between cost and time delay, e.g., \( R = \$20/h \).
Definition 4.2: The disassembly values of place \( \pi \) and transition \( \lambda \) are denoted as \( d(\pi) \) and \( d(\lambda) \), respectively, and calculated in a recursive manner using the following equations:

1) \( \forall \pi \in Q', d(\pi) = \pi(\pi) \)
2) \( \forall \lambda \in T, d(\lambda) = \sum_{q \in \lambda} d(q) - h(t) \)
3) if \( \pi \in Q - Q' \), \( d(\pi) = \max_{\lambda \in \lambda^*} \{ \pi(\pi), \max_{\lambda \in \lambda^*}(d(\lambda)) \} \).

After calculating all the disassembly values, the EOL value of each place \( \pi \in Q \) is compared with that of its corresponding transitions. \( \forall t \in \pi^* \) if \( d(\lambda) < \pi(\pi) \), i.e., \( t \)'s disassembly value is lower than its input place \( \pi \)'s EOL value, \( t \) is pruned from a PPN.

With the information of EOL values of a product, subassemblies, and components, the cost values of disassembly methods, and costs caused by time delays of workstations, a disassembly path of a product can be obtained by Algorithm 4.1. It is a heuristic algorithm that tends to maximize the EOL value of the product while observing the current commitments of the facility resources.

Algorithm 4.1 Disassembly Plan:

Step 1) Apply Procedure \texttt{reman\_value} to generate \( d(x) \), \( x \in Q \cup T \) in a PPN.

Step 2) Set \( Z = \{p_1\} \) (assuming \( p_1 \) is the root node for every incoming product).

Step 3) Set \( \text{DP} = \Phi, C = \Phi, \) and \( U = \Phi \) (DP represents final disassembly path, \( C \) represents the last components after disassembly and \( U \) is the set of workstations the product will go through).

Step 4) While \( (Z \neq \Phi) \) do:

For each node \( p \) in \( Z \):

Apply Procedure \texttt{Optimal\_Path} to obtain the best disassembly method at each place, i.e., \( T(p) \). \( \forall \pi \in Q - Q' \):

if \( T(p) \neq \Phi \)

Select the first transition \( t \) in \( T(p) \).

\( \text{DP} = \text{DP} \cup \{ t \} \);

\( U = U \cup \{ t \in W \} \);

Update time delay vector \( r \):

\( r(w_i) = \max\{r(w_i) - \lambda(t), 0\}, \)

\( i = 1, 2, \ldots, n \);

\( Z = Z \cup \{ p \} \);

else \( C = C \cup \{ p \} \);

\( Z = Z - \{ p \} \);

End

Procedure \texttt{reman\_value} (PPN) [25]:

For \( p \in Q' \), set \( d(p) = \pi(p) \);

Set \( L = Q - Q' \);

While \( (L \neq \Phi) \) do:

Find a place \( p \in L \) such that \( \forall q \in (j^*), q \notin L \), i.e., \( d(q) \) is known.

Calculate:

\( d(t) = \sum_{q \in \lambda^*} d(q) - h(t); \)

\( d(p) = \max_{\lambda \in \lambda^*} \{ \pi(\pi), \max_{\lambda \in \lambda^*} d(\lambda) \} \).

Let \( L = L - \{ p \} \), i.e., remove \( p \) from \( L \);

End

---

**Table II**

<table>
<thead>
<tr>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p^\text{initial}_{i,k} )</td>
</tr>
<tr>
<td>( p^\text{source}_{i,j} )</td>
</tr>
<tr>
<td>( p^\text{inter}_{i,k,j} )</td>
</tr>
<tr>
<td>( p^\text{final}_{i,k} )</td>
</tr>
<tr>
<td>( t_{i,j,k} )</td>
</tr>
</tbody>
</table>
Procedure Optimal Path (p):

Set $T(p) = \Phi$;
if $\pi(p) < d(t)$
for all $t \in P$ and $d(t) \geq \pi(p)$, then $T(p) = T(p) \cup \{t\}$;
Order transitions in $T(p)$.

Apply (4.1) and transfer time delay vector $r$ into cost value vector $\sigma$.
Calculate decision values $\delta(t) = i$ if $d(t) - \sigma(g)$ is the $i$th smallest in \{$d(t) - \sigma(g), g \in t \cap W, t \in T(p)$\}.
Order transitions in $T(p)$ in a decreasing order of their decision values.
End

Since this paper assumes every transition in a PPN has only one nonleaf output place and the number of visitation to any single workstation in a disassembly plan is one, no concurrent execution of various operations exists in the plan. Thus, the components of vector $r$ are decreased by the estimated transition duration, every time that a new operation is introduced in the present plan. The assumptions guarantees the correct estimation of the delay function $r$. Due to the introduction of $r$ and its estimation errors, the $DP$ obtained through Algorithm 4.1 is not globally optimal as the original one did [25]. However, the consideration of machine availability and waiting time in this algorithm makes it more suitable for realistic applications.

B. Scheduling Algorithm

After a product or subassembly enters a workstation, it goes through a secondary inspection. In this area, an SPN is developed to model machines and use a token to represent the availability of each machine place. During operation, the workstation controller always monitors the status of machines and track their tokens. Once the station is occupied, a waiting time for this machine is associated with the corresponding token and updated as time passes. Then machines are assigned based on their speeds and availability. Algorithm 4.2 shown below fulfills this task and obtains the execution time, $\theta_{ik}$, to perform the disassembly process for the $i$th job in the $k$th workstation. $\eta_k$ denotes the total disassembly time for the disassembly of the $i$th job.

Algorithm 4.2 Scheduling and Execution ($SPN_{ik}$):

Set $V = \{p_k\}$, and tag $p_k$ “new.”
Set $\theta_{ik} = 0$.
While ($V \neq \Phi$) do:

Find a $t_k \in V^\bullet$ such that $\Gamma(t_k) + \Lambda(q_k), q_k \in (t_k^\bullet \cap R)$ is the smallest in \{$\Gamma(t) + \Lambda(q), q \in (t^\bullet \cap R), t \in V^\bullet$\}.
and set $\theta_{ik} = \theta_{ik} + \Gamma(t_k) + \Lambda(q_k)$.
Update waiting time vector $\Lambda$: $\forall p \in R, \Lambda(p) = \max\{\Lambda(p) - \Gamma(t_k) - \Lambda(q_k), 0\}$.
Set $V = \{p, p \in (\bullet) \cap S\}$.
End

From Algorithm 4.2, the total disassembly time for the $i$th job is $\eta_k = \sum \theta_{ik}$ (the $k$th workstation $\in U$).

The complexity of Algorithm 4.2 is $n \times m$, where $n$ and $m$ denote the number of tasks and machines in $SPN_{ik}$. The algorithm guarantees that each disassembly task for the $i$th job is processed as quickly as possible based on the present state.

V. Case Study

To fully understand the above concepts and algorithms, the disassembly of a batch of obsolete PCs in an example system is used to demonstrate the disassembly process planning and scheduling in both primary inspection and secondary inspection units. Section V-A gives the implementation method; Section V-B presents the experimental results.

A. Implementation Method

To fulfill the inspection function for each incoming product, this paper proposes the following implementation steps.
1) Construct WPN and update the initial markings.
2) Construct PPN for an incoming product given the product information and all feasible disassembly choices.
3) Collect and associate all the data with places and transitions in the PPN.
4) Apply Algorithm 4.1 to the Primary Inspection for this product.
5) Construct an SPN for each task of the product being processed.
6) Collect and associate all the data with places and transitions in the SPN.
7) Execute Algorithm 4.2 in the IFDS.
When a discarded product comes to the IFDS facility and waits for its processing, the first four steps are executed. Based on the results of the primary inspection, the rest of steps are performed when each task of this used product reaches a workstation in its process plan.

A discarded PC consists of eleven main components. Its layout without case and cables is given in Fig. 6, which shows how nine components are connected to each other.

First, check the workstations’ condition and model a WPN, where dynamically updated markings represent the number of idle and busy machines.

Second, to construct a PPN for an incoming PC, start with the entire product labeled by \( p_1 \). Identify all possible ways to disassemble it into the next-level subassemblies or components. For instance, at \( p_1 \), only one method is identified and represented by \( t_1 \), which means that other components can be disassembled only after opening the case and taking out all cables. This process is repeated for each subassembly until no subassembly left. Then workstation places are added. Finally the complete PPN is obtained as shown in Fig. 3. The dotted arrows show the parallel disassemblies in a real-time system. For clarity, workstation places \( w_{d_1} (t = 1 \text{ to } 4) \) are shown more than once in Fig. 3.

Then, the relevant information associated with places and transitions is collected. The detailed data is given in the next subsection.

Based on the result of Primary Inspection, an SPN is modeled for each job. To construct SPN\(_{i,j} \), start with the initial status of the \( i \)-th job in the \( j \)-th workstation labeled by \( p_2 \). Identify all feasible ways to process its disassembly into the next-level subassemblies/components. Repeat this procedure until the final status of the \( i \)-th job in the \( j \)-th workstation is reached. Taking SPN\(_{34} \) as an example, at \( p_1 \), there are two machines that can process the disassembly of case and cables, which are represented by two transitions as shown in Fig. 4. The detailed data associated with places and transitions are also presented in Section V-B.

Finally, the proposed algorithms are implemented into the IFDS. This work has developed a simulation module that runs under Windows 98/NT operating system written in Visual C++. The software architecture includes Database, Process Planner and Controller. Database provides the relevant information of each product and workstation, including EOL and cost values in PPN, process time in SPN, and idle/busy markings in WPN. Based on the information from Database, Process Planner obtains the DP, decides machine assignments, and transfers it to Controller. Then, Controller assigns jobs to machines according to the results from Process Planner, updates workstation information, and feedbacks it to Database.

B. Experimental Results

In this section, a value gain in the disassembly plan DP is defined first. Two cases are considered in the simulation software. In the baseline case, a coming PC is completely disassembled through Workstations 1 to 4, no matter what condition it has. The disassembly sequence is fixed. A job immediately takes an idle machine based on their speeds when both are available. In the proposed case, each coming PC is first inspected based on its potential disassemblability and system condition. Then it is disassembled following its disassembly sequence derived by Algorithms 4.1 and 4.2.

To show the significant difference between two cases, a sample mean, a sample variance, confidence interval estimation, and standard error are introduced and a correlated sampling statistical technique is used [1]. Finally, the execution of Algorithms 4.1 and 4.2 is presented. Two alternatives in real-time procedures are compared through a set of experiments.

Definition 5.1: The value gain in a disassembly plan DP \( f(DP) \) is

\[
f(DP) = \sum_{p \in C} \pi(p) - \sum_{t \in DP} h(t).
\] (5.1)

In IFDS, there are various parameters. Some of them are the input data of the Process Planner (i.e., \( \pi, h, \gamma \)); and others are calculated during the operations (i.e., \( \delta \)). In the real-time situation, there are two kinds of input data, one of which is decided according to the previous benchmark experience and the market price (i.e., EOL value \( \pi \), cost value \( h \)); the other is dynamically obtained through sensors (i.e., \( \gamma \)). During the implementation of the simulation software, the input data are created as follows and used to both cases.

- Products come into the IFDS with an arrival interval, which has a uniform distribution between [0.01, 1] minute.
- The process time for tasks is randomly chosen, which also has a uniform distribution between [0.1, 5] minutes.
- In a PPN, the EOL value of each place and the cost value of each transition are both randomly chosen with a uniform distribution, which have the range [1.2, 1] and [0.001, 1], respectively.
- Users can define the number of workstations, machines, and tasks in each workstation. In our example, the IFDS has four workstations in total. Each workstation is equipped with two machines.

1) Illustration of Disassembly Sequences: In this experiment, \( J_3 \) is chosen as an example. The detailed execution of Algorithm 4.1 is presented as follows.

1) When it comes into the IFDS, its PPN and WPN are modeled and the corresponding data is collected from Database

\[
\pi = (-1.40, -0.90, -0.80, -0.40, -0.30, 0.10, 0.20, 0.50, 0.70, 0.70, 0.90, 0.90, 0.90)
\]
\[
h = (0.013, 0.079, 0.068, 0.062, 0.057, 0.089, 0.025, 0.067, 0.071, 0.001, 0.026, 0.019, 0.027)
\]
\[
\lambda = (0.94, 0.38, 0.22, 0.32, 0.82, 0.42, 0.16, 0.87, 1.25, 0.38, 0.16, 0.87, 1.25, 0.87).
\]

This implies that the maximum disassembly times are 9.44, 16.87, 12.59 and 3.82 for the processes in Workstations 1 to 4, respectively. Note that, if a transition in DPN implies several tasks, its disassembly time is the sum of these tasks’ processing times. The time unit is minute.
2) The workstation information is transferred from the WPN: $r = (2.73, 0, 0, 0)$ where 2.73 indicates that the earliest machine in $W_1$ will be ready in 2.73 min.

3) Calculate the disassembly values of places and transitions using procedure `reman_value`:

$$d(p) = (3.89, 3.20, 2.54, 2.31, 2.30, 1.61, 1.67, 1.43, 0.70, 0.70, 0.80, 0.90, 0.90)$$

$$d(t) = (3.89, 3.17, 2.44, 2.31, 2.25, 1.61, 1.67, 1.43, 3.14, 3.20, 2.54, 2.31, 2.30).$$

4) Apply procedure `Optimal_Path` to obtain the best disassembly method at each place

$$T(p_1) = \{t_1\}, \quad T(p_2) = \{t_{10}, t_2, t_3\}, \quad T(p_3) = \{t_{11}, t_3\}, \quad T(p_4) = \{t_7\}, \quad T(p_5) = \{t_5\}.$$

5) The sequence of operations for $J_3$ is obtained as $\{t_{12}, t_{30}, t_{23}, t_3\}$. $U = \{W_1, W_3, W_2, W_4\}$, and $f(DP) = 3.89$.

Note that the EOL value of the entire product without disassembly is $\pi(p_1) = -1.4$. At the end of the disassembly process, the EOL value of the PC increases to $3.89$. Fig. 7 gives the disassembly value as a function of the disassembly steps. Following the decided disassembly sequence, $J_3$ goes through Workstations 1 and 3, then 2 and 4. Once $J_3$ enters the 4th workstation, $SPN_{33}$ is modeled. The relevant information associated with places and transitions is collected. Taking $SPN_{33}$ as an example, the execution of Algorithm 4.2 is presented as follows.

1) Collect the corresponding data for $SPN_{33}$

$$\Gamma = (4.52, 3.46, 4.32, 4.92)$$

$$\Lambda = (1.45, 0)$$

where 1.45 indicates Machine 1 ($p_2$) will be available in 1.45 min.

2) The execution plan is decided and the execution time for the third job in the first workstation is obtained: $\theta_{31} = 7.78$ min.

2) Comparison Results: In this set of experiments, as the batch size of PCs changes from 100 to 10,000, the average disassembly time and value gain are obtained for and compared between baseline and proposed methods that are shown in Figs. 8 and 9. Using the data in Table III, 95% confidence intervals for the true mean differences in average disassembly time and value gain of products are calculated. The approximate $100(1 - \alpha)%$ confidence interval for $\theta$ is defined as

$$\hat{\theta} \pm g_{(n-1)} \hat{V}(\hat{\theta})$$

or

$$\theta - g_{(n-1)} \hat{V}(\hat{\theta}) \leq \theta \leq \hat{\theta} + g_{(n-1)} \hat{V}(\hat{\theta})$$

(5.2)

where $\hat{\theta}$ is a sample mean of $\theta$ based on a sample of size $n$; $\hat{V}(\hat{\theta})$ is the standard error of $\hat{\theta}$; $g = (\hat{\theta} - \theta)/\hat{V}(\hat{\theta})$ and $g_{(n-1)}$ is the $100(1 - \alpha)%$ percentage point of a t-distributed with $n - 1$ degrees of freedom. The value of $g_{(n-1)} = g_{0.025, 5} = 2.57$ is obtained from t-distribution table [1].

A 95% confidence interval for average disassembly time is given by

$$1.111 \pm (2.57)0.029$$

or

$$1.036 \leq \theta_1 - \theta_2 \leq 1.186.$$

A 95% confidence interval for value gain of products is given by

$$-0.078 \pm (2.57)0.002$$

or

$$-0.083 \leq \theta_1 - \theta_2 \leq -0.073.$$

The 95% confidence interval for average disassembly time lies completely above zero, which provides strong evidence that $\theta_1 < \theta_2 > 0$—that is, the proposed method is better than the baseline one, because its average disassembly time is smaller. Another convincing evidence that the proposed methodology is better, is the hypothesis $\theta_1 < \theta_2 < 0$ for the value gain of products, which shows that the EOL values of products are increased in the proposed case.

The simulation output analysis represents a significant improvement in terms of system throughput and overall EOL values for the proposed methodology. The experiments with input data that have truncated normal distribution are also tested and the similar results are obtained.

VI. CONCLUSION

Demanufacture of products for component and material recovery is an emerging field of research. This paper addresses algorithmic issues to design and implementation of an integrated flexible demanufacturing system (IFDS). Three PN models (WPN, PPN, and SPN) are designed and implemented in real time. To deal with the unique features in disassembly planning, the EOL and cost value functions are introduced to places and transitions in a PPN, respectively. To incorporate the time delay caused by availability of machines in workstations,
the delay time function associated with each workstation place is introduced in a PPN. The information of each workstation place comes from a WPN. In a SN, processing and delay time functions assigned to transitions are used to deal with machine assignment in each workstation. Thus, the proposed heuristic methodology accommodates a disassembly job to the running schedule of an IFDS and aims to maximize the EOL value of the considered product and minimize the job cycle time, while at the same time it observes the current commitments of the facility resources.

To the authors’ knowledge, no paper has comprehensively dealt with a problem of maximizing both the EOL value of products and the demanufacturing system’s throughput. This work successfully overcomes a deficiency in [25], [26] that ignores real-time resource capacity and availability, thereby making the proposed methodology more applicable to real industrial settings. The main benefit of this methodology is that a reliable and environmentally friendly execution of robotic disassembly task sequences is guaranteed through modeling, planning and demanufacturing control.

Although the proposed algorithms can handle different products with different batch sizes, the efficiency issue and sensitivity analysis remain open. Both group technology and arrival intervals are necessary to be used in order to optimize the system productivity and EOL values. Moreover, the expression of updating delay time function τ in primary inspection is accurate only when the derived disassembly plan for a product does not contain parallel tasks. In order to use the heuristic methodology successfully in practical applications, an accurate method to estimate delay time function τ considering concurrent processes in the disassembly path for a product deserves research effort. More factory data need to be used to test our methodology in the future. Adaptive algorithms considering the operational failure and their industrial scale implementation need to be developed.

ACKNOWLEDGMENT

The authors would like to thank the comments provided by anonymous reviewers and the Associate Editor, which help greatly improve this paper.

REFERENCES

Ying Tang received the B.S. and M.S. degrees from Northeastern University, China, in 1996 and 1998, respectively, and the Ph.D. degree from the New Jersey Institute of Technology, Newark, in 2001. She is an Assistant Professor of computer science and computer engineering at Pacific Lutheran University, Tacoma, WA. Her research interests include modeling, design and scheduling of computer-integrated manufacturing and demanufacturing systems, Petri net applications, and discrete event systems. Dr. Tang received Best Student Paper Award at the 2000 IEEE International Conference on Systems, Man and Cybernetics, and the first prize at the 2001 North New Jersey IEEE Section Graduate Category Student Paper Presentation Contest. She is a member of the IEEE Systems, Man and Cybernetics Society and Sigma Xi.

Mengchu Zhou (S’88–M’90–SM’93) received B.S. degree from Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree from Beijing Institute of Technology, Beijing, China, in 1986, and the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1990. He joined New Jersey Institute of Technology, Newark, in 1990, and is currently a Professor of electrical and computer engineering and the Director of Discrete-Event Systems Laboratory. He is interested in computer-integrated robotic systems, disassembly automation, Petri nets, Internet-based manufacturing, wireless network security, and intelligent control. He has more than 170 publications including four books, 40 journal papers, and 10 book chapters. He was invited to lecture in Australia, Canada, China, France, Germany, Hong Kong, Italy, Japan, Korea, Taiwan, and the U.S. He organized and chaired more than 50 technical sessions and served on program committees for many conferences. He has been the PI/Co-PI for 20+ research and education projects with budget over $2M by National Science Foundation, Department of Defense, Engineering Foundation, New Jersey Science and Technology Commission, and industry. Dr. Zhou was the recipient of the National Science Foundation’s Research Initiation Award, the CIM University—LEAD Award by the Society of Manufacturing Engineers, the Perlis Research Award, the Humboldt Research Award for US Senior Scientists, the Asian American Achievement Award by Asian American Heritage Council of New Jersey, and the Leadership Award by the Chinese Association for Science and Technology—USA. He has been an editor of the International Journal of Intelligent Control and Systems since 1996. He was Program Chair of the 1998 and 2001 IEEE International Conferences on Systems, Man, and Cybernetics and the 1997 IEEE International Conference on Emerging Technologies and Factory Automation. He has served as Guest Editor for the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, IEEE TRANSACTIONS ON SEMICONDUCTOR MANUFACTURING, and Journal of Intelligent Manufacturing.

Ying Tang received the B.S. and M.S. degrees from Northeastern University, China, in 1996 and 1998, respectively, and the Ph.D. degree from the New Jersey Institute of Technology, Newark, in 2001. She is an Assistant Professor of computer science and computer engineering at Pacific Lutheran University, Tacoma, WA. Her research interests include modeling, design and scheduling of computer-integrated manufacturing and demanufacturing systems, Petri net applications, and discrete event systems. Dr. Tang received Best Student Paper Award at the 2000 IEEE International Conference on Systems, Man and Cybernetics, and the first prize at the 2001 North New Jersey IEEE Section Graduate Category Student Paper Presentation Contest. She is a member of the IEEE Systems, Man and Cybernetics Society and Sigma Xi.