Uncertainty Management in Optimal Disassembly Planning through Learning-based Strategies

Spyros A. Reveliotis, IIE Member
School of Industrial & Systems Engineering
Georgia Institute of Technology
765 Ferst Drive
Atlanta, GA 30332
Phone: 404-894-6608
Fax: 404-894-2301
Email: spyros@isye.gatech.edu

Abstract

Currently there is increasing consensus that one of the main issues differentiating the re-manufacturing from the more traditional manufacturing processes is the need to effectively model and manage the high levels of uncertainty inherent in these new processes. Hence, the work presented in this paper undertakes the issue of uncertainty modelling and management as it arises in the context of the optimal disassembly planning (ODP) problem, one of the key problems to be addressed by re-manufacturing processes. More specifically, the presented results formally establish that the theory of *reinforcement learning*, currently one of the most actively researched paradigms in the area of machine learning, constitutes a rigorous, efficient, and effectively implementable modelling framework for providing (near-)optimal solutions to the optimal disassembly problem, in the face of the aforementioned uncertainties. In addition, the proposed approach is exemplified and elucidated by application on a case study borrowed from the relevant literature.

Keywords: disassembly planning, product recovery, uncertainty management, reinforcement learning, (neuro-)dynamic programming

1 Introduction

During the last decade, the developed economies have been becoming increasingly aware of the need to handle used products in an environmentally conscious manner. The typical practices adopted in the earlier phases of industrialization, that would dispose products reaching the end of their functional life either through dumping in landfill sites or through shredding and incineration, are reckoned to be too polluting and unnecessarily wasting precious environmental resources, by failing to retrieve and reuse materials and functional components potentially available in the discarded product. Hence, under the pressure of emerging legislation in most of the developed countries, manufacturers proceed to set up additional operational networks that will retrieve their products upon reaching the end of their life, and if possible, reprocess and reuse the constituent components and materials. This new set of reclaiming, reprocessing and redistribution operations is collectively known as *reverse logistics* [5], and their design and management defines a novel and challenging technical area of production system modelling, analysis and control.

One particular theme that is emerging as a predominant issue in the current reverse logistics related literature is the need for effective modelling, analysis and management of the high levels of uncertainty inherent in the operation of these systems. For instance, three recent survey works reported in [12, 21, 5] identify the modelling and analysis of the impact of the product and environmental uncertainties underlying the operation of modern reverse logistics systems as one of the major issues to be addressed by the research community. Furthermore, the same works point out that the effective management of these uncertainties is one of the fundamental issues differentiating reverse logistics and remanufacturing-related research from the more traditional logistics and manufacturing systems theory.

Motivated by these general remarks, the work presented in this paper undertakes the problem of uncertainty modelling and management in the context of the more specific area of *"(optimal) disassembly planning (ODP)"*, which constitutes a core problem to be addressed in the operation of any reverse logistics process. A basic characterization of the ODP problem is provided by means of Figure 1, which has been adapted from [12]. As depicted in Figure 1, the disassembly of the reclaimed product units to a number of components and sub-assemblies constitutes a primary step in the entire reverse logistics process. The derived units will be subsequently directed either (i) for remanufacturing / refurbishing and reuse, or (ii) for extraction and recycling of (some of) their materials, or, finally, (iii) for disposal through dumping or incineration. In this operational context, the ODP problem seeks to determine the level of disassembly of each

returned product unit to its constituent elements, and the particular venue of disposition of the retrieved components, so that the total (monetary) value extracted during the process operation is maximized, while at the same time, various technical, legislative, environmental, and any other managerial considerations are observed.

It can be argued that the ODP problem is one of the most extensively investigated problems in the reverse logistics literature. As it was already mentioned, three recent and quite comprehensive surveys of the relevant literature are provided in [12, 21, 11]. All the works presented in these surveys address the ODP problem by (i) first formalizing in a particular representation the dynamics of the disassembly process, as constrained by the relevant technological, environmental and legislative requirements, (ii) subsequently augmenting this representation with a "cost structure" modelling the economic elements involved in the decision-making process, and (iii) finally defining and solving an optimization problem by means of the modelling framework established in steps (i) and (ii). Yet, it is also true, that with the exception of the works presented in [4, 10, 15, 7, 26, 24, 6, 18, 13], all the remaining existing works on the ODP problem are assuming a totally deterministic model for the underlying process dynamics and the applying cost structure. Furthermore, among the works that recognize the potential stochasticity of these problem elements, many of them (e.g., [4, 10, 15, 7, 26]) deal with this issue only as an after-thought, through (a) the sensitivity analysis of a solution developed according to a deterministic optimization model and/or (b) the on-line heuristical adjustment of the derived solution, in case that there exists significant deviation of the actual implementation from the normative model. On the other hand, the works of [24, 6, 13] recognize the need to explicitly address the involved uncertainty during the determination of the optimal policy, but they resort to problem representations that presume the a priori availability of some (quite sophisticated) model that provides a complete quantitative characterization of this uncertainty; only the work of [18] recognizes the potential unavailability of the information necessary to develop such a priori fully quantified models and the resulting need to derive this information in real-time.

The defining positions of our work, which are in agreement with the positions taken in [18], and also with an emerging consensus in the broader community, are that: (i) understanding the impact of the involved uncertainty and accounting for it during the development of optimized disassembly plans, is important for the effective optimization of the overall process performance; furthermore, (ii) any assumption regarding the a priori availability of a fully quantified model characterizing problem elements like the statistical distributions and/or indices modelling the randomness in the cost data and the probability distributions determining the classification of the various components and subassemblies to different quality classes, is rather unrealistic, since

(much of) the information necessary to develop such a model can be provided only through observation of the process itself. These two positions further suggest that any attempt towards developing an optimizing solution to the ODP problem, which is the focus of this work, must involve some algorithmic components that will allow the decision making process to (i) accumulate its past experience to a pertinently defined set of data structures, and, at the same time, (ii) exploit the "knowledge" captured in these data sets towards improving the overall system performance. In broader systems theory, algorithms with the aforementioned capabilities are known as *"learning"* algorithms [16]. Hence, *the main topic and the intended contribution of this paper is the design of effective and computationally efficient learning algorithms for the ODP problem*. More specifically, we are focusing on a particular class of learning algorithms known as *"reinforcement learning"* in the relevant literature [19]. We believe that these algorithms are most appropriate for the ODP problem due to (i) their strong affinity to the *dynamic programming* framework [1], which, as it will be shown in the next section, is the natural representation of the problem under consideration, and (ii) their computational simplicity and implementation flexibility, two properties which render them compatible with the conditions prevailing in the involved facilities. In addition, reinforcement learning algorithms have been extensively studied recently, and currently, there is a significant body of analytical results characterizing their convergence and dynamics.

With this basic positioning of the presented results, the rest of the paper is organized as follows: The next section provides an analytical characterization of the ODP problem that reveals the underlying problem structure but also the nature and impact of the aforementioned uncertainties on the derived solutions. Subsequently, Section 3 establishes that reinforcement learning provides an effective and computationally efficient method for generating optimized disassembly plans in the face of the aforementioned process uncertainties. Section 4 considers the implementation of the proposed algorithms in the re-manufacturing facility, providing a number of observations and suggestions that can potentially expedite the learning process and facilitate the integration of these algorithms in the overall operational context. Finally, Section 5 concludes the paper and highlights directions for future work. Throughout the paper, a case study adapted from [9] exemplifies and elucidates the primary concepts and methods introduced in this work.

2 An analytical formulation of the ODP problem

2.1 A Petri net-based modelling framework

As it was pointed out in the discussion of the introductory section, any analytical characterization of the ODP problem must be based on a formal representation of the disassembly process that will be able to express explicitly, yet compactly, all the feasible[1] disassembly sequences and their associated economics. Presently, the two most widely adopted representations for the disassembly process and the associated ODP problem are based on (i) *AND/OR graphs* [8] and (ii)*Disassembly Petri Nets (DPN)* [25] – see also [21, 11]. Although these two representations are essentially equivalent [21, 25], in this work we adopt the DPN version, since the semantics of the Petri net-based modelling framework (i) are more standardized than those of the AND/OR graphs, by now being widely accepted as a basic modelling framework in the broader systems literature, and in addition, (ii) they provide, through the notions of "place marking" and "transition firing", a well-defined mechanism for representing the disassembly process *state* and the evolution of the process dynamics. However, our work extends the basic definition of DPN's provided in [25, 20], in a way that accounts for the explicit modelling of the part condition and the relevant classification /testing process; for this reason, the adopted PN-based representation will be characterized as *Extended DPN (E-DPN)*. Next, we proceed to a detailed characterization of the E-DPN model, assuming that the reader is familiar with the basic elements of the PN theory; an excellent introductory treatment of the PN theory and its employment in manufacturing applications can be found in [23].

The E-DPN model is formally defined as an eight-tuple

$$\textit{E-DPN} = (P, T, F, m_0, \rho, \tau, \xi, \delta)$$

where

1. $Z = (P, T, F, m_0)$ is a *connected acyclic* Petri net presenting the following structure:

   (a) The set of *places*, P, is partitioned to three subsets: P_R, P_C and P_L. Places $p \in P_R$ model the originally retrieved product units as well as units extracted from the various disassembly steps in their **R**aw status, i.e., before their testing and classification to the various quality classes discerned by the underlying process. Places $p \in P_C$ model

[1] We remind the reader that feasibility in the ODP context should be perceived with respect to technological, as well as environmental and legislative considerations.

(**C**lassified) units after testing, categorized according to some quality attribute(s). Finally, places $p \in P_L$ model units directed to their final reprocessing operation; therefore, they constitute terminal (or **L**eaf) places in the considered disassembly sequence.

(b) The set of *transitions*, T, is also partitioned to three subsets: T_D, T_C and T_P. Transitions $t \in T_D$ model disassembly operations, while transitions $t \in T_P$ model operations corresponding to the final re-processing of the extracted units. Finally, transitions $t \in T_C$ model the classification of the different extracted artifacts, through testing and evaluation of some quality attribute(s).

(c) The net *flow relation*, F, is a function from $(P \times T) \cup (T \times P)$ to the set of nonnegative integers, Z_0^+; in the E-DPN modelling framework, F models the dynamics of a typical disassembly process by satisfying the following conditions:[2]

i. $\{p \in P : {}^\bullet p = \emptyset\} = \{p_0\} \subseteq P_R$, i.e., the whole net has a single *source* node, corresponding to the original artifact in its raw (unclassified) state. Also, $\forall p \in P_R \cup P_C$, $|p^\bullet| \geq 1$, where $|\cdot|$ denotes the cardinality of the set argument; i.e., places $p \in P_R \cup P_C$ correspond to *non*-terminal stages in the overall decision-making process.

ii. $\forall t \in T_D$, ${}^\bullet t = \{p\} \subseteq P_C$, and $t^\bullet \subseteq P_R$; i.e., a disassembly operation has as input a single classified item and it produces a number of new unclassified artifacts. Similarly, $\forall t \in T_C$, ${}^\bullet t = \{p\} \subseteq P_R$, and $t^\bullet = \{q\} \subseteq P_C$, while $\forall t \in T_P$, ${}^\bullet t = \{p\} \subseteq P_C$, and $t^\bullet = \{q\} \subseteq P_L$.

iii. Finally, F must take *binary* values on all of its domain, except for the part corresponding to $T_D \times P_R$; for pairs $(t, p) \in T_D \times P_R$, $F(t, p)$ will express the number of artifacts of the p-type that are generated through a disassembly operation of the t-type, and therefore, this value can be any nonnegative integer.

(d) The net *initial marking*, m_0, is a function from the place set P to Z_0^+; in particular, $m_0(p)$ equals 1 if $p = p_0$, and 0 otherwise, i.e., the entire disassembly process starts with a returned but still unclassified product unit. Also, to facilitate the subsequent discussion, we characterize as a *terminal* marking, any marking m reachable from m_0

[2]We remind the reader that, in the PN formalism, $F(p, t)$ denotes the number of tokens that must be available at place p for a single firing of transition t, and are consumed by this firing. Similarly, $F(t, p)$ denotes the number of tokens placed in place p by a single firing of transition t. Furthermore, ${}^\bullet p = \{t \in T : F(t, p) \geq 1\}$; $p^\bullet = \{t \in T : F(p, t) \geq 1\}$; ${}^\bullet t = \{p \in P : F(p, t) \geq 1\}$; and $t^\bullet = \{p \in P : F(t, p) \geq 1\}$.

through a sequence of transition firings, with $m(p) = 0,\ \forall p \in P_R \cup P_C$.

2. ρ is a set of discrete probability distributions, with each distribution corresponding to a place $p \in P_R$, and with its support being equal to $p^\bullet$. Given a place $p \in P_R$ and a transition $t \in p^\bullet$, $\rho(p,t)$ denotes *the probability that a unit of type p, upon testing, will be found in the condition modelled by (classifying) transition t.*

3. $\tau :\ T_D \cup T_P \rightarrow R_0^+$, where R_0^+ denotes the set of nonnegative real numbers, is a function modelling the *(expected) costs incurred by the various disassembly and (re-)processing operations.*

4. $\xi :\ P_L \rightarrow R_0^+$ is a function modelling the *(expected) returns from the various terminal operations.*

5. δ represents another set of discrete probability distributions, with each distribution corresponding to a place $p \in P_C$, and with its support being equal to $p^\bullet$. For every place $p \in P_C$ and a transition $t \in p^\bullet$, $\delta(p,t)$ denotes *the probability that a unit of type p will be disposed according to the operation modelled by transition t.* These distributions will express *the adopted disassembly plan.*

In this work, we are interested in identifying a disassembly plan, expressed by a distribution set δ^*, that will maximize the expected return from each processed item. This objective is formally expressed by introducing the *place value* function, $\pi^\delta :\ P \rightarrow R$, which is associated with the disassembly plan defined by the distribution set δ, and maps each place $p \in P$ to a real value representing *the expected return to be obtained from the processing of a unit of the artifact represented by place p, according to the disassembly plan defined by the aforementioned distribution set* δ. Then, our problem is to identify $\delta^* = \arg\max_\delta \{\pi^\delta(p),\ \forall p \in P\}$.[3] We shall also denote $\pi^*(p) \equiv \pi^{\delta^*}(p)$ across all $p \in P$. The $\pi^*(p)$ values for places $p \in P_L$ are equal to $\xi(p)$, introduced in item (4) of the E-DPN definition. For the remaining places, an optimized disassembly plan and the resulting $\pi^*(p)$ values can be computed according the dynamic programming (DP) algorithm discussed next.

[3]We remind the reader that $\arg\max_{x \in X} f(x)$ denotes any maximizer of the function $f()$ among the elements of the set X.

2.2 Computing an optimal disassembly plan through dynamic programming

In the face of the acyclic structure of the E-DPN model introduced in the previous paragraph, the maximization of $\pi^{\delta}(p),\ \forall p \in P$, can be achieved through a specialization of the broader logic of dynamic programming to the ODP problem context, that computes the maximized $\pi^*(p)$-values in a recursive manner, starting from the leaf nodes. The detailed recursion is as follows:[4]

$$\forall p \in P_L,\ \pi^*(p) := \xi(p) \tag{1}$$

$$\forall p \in P_R,\ \pi^*(p) := \sum_{t \in p^\bullet} \rho(p,t) \cdot \pi^*(t^\bullet) \tag{2}$$

$$\forall p \in P_C,\ \pi^*(p) := \max_{t \in p^\bullet} \{ \sum_{q \in t^\bullet} (F(t,q) \cdot \pi^*(q)) - \tau(t) \} \tag{3}$$

Equations 1–3 above have a very straightforward interpretation in the E-DPN context. Specifically, Equation 1 implies that the value extracted from the various artifacts in their terminal operations is determined by external factors relating to their inherent value and the prevailing market conditions. Equation 2 expresses the fact that the value of an unclassified item in some place $p \in P_R$ is defined by the values corresponding to the various classifications of this item, $\{\pi^*(t^\bullet) : t \in p^\bullet\}$, averaged according to the classification probability distribution $\rho(p, \cdot)$. Finally, Equation 3 implies that the optimal expected value to be associated with an artifact belonging to the category corresponding to a place $p \in P_C$ is the value resulting from a processing option that maximizes the resulting (expected) return, where the latter is defined as the cumulative optimal value of all the derived artifacts reduced by the corresponding processing cost. This last observation characterizes also the optimal disassembly plan: using the δ^* representation introduced above, it follows that

$$\forall p \in P_C, \forall t \in p^\bullet,$$

$$\delta^*(p,t) = \begin{cases} 1 & \text{for some } t^* \in \arg\max_{t \in p^\bullet} \{ \sum_{q \in t^\bullet} (F(t,q) \cdot \pi^*(q)) - \tau(t) \} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Notice that the optimal plan δ^*, defined in Equation 4, has $\delta^*(p,t) \in \{0,1\}$ for all pairs (p,t), i.e., it constitutes a *deterministic* (disassembly) policy, according to the DP terminology.

[4] With a slight abuse of notation, $\pi^*(t^\bullet)$, in Equation 2, denotes the value of the *single* place q which is the output place of transition t (since, for the considered case, $t \in T_C$ – c.f. item (1.c.ii) in the definition of the E-DPN structure).

2.3 Example

The E-DPN modelling framework and the application of the DP-based algorithm for the computation of the optimal value function and the optimal disassembly plan, is demonstrated by means of an example case study adapted from [9]. The returned item is a particular TV model, with the basic bill of material presented in Table 1. During the disassembly, each of the extracted components is classified in two classes, "repairable" (or class 1) and "worn out" (or class 2), according to the probability distributions listed in Table 2. The disposition venues generally available for the TV sets and the extracted components are: upgrading (UP), restoration (RES), disassembly (DSBL), recycling (REC) and disposal (DISP). However, the particular options available for a certain component depend on its quality class, with the exception of recycling and disposal, which are class-independent. Furthermore, each processing option results in the generation of (monetary) value, which depends on the component, its condition, and, of course, the selected option itself. The viable processing options for each (sub-)assembly, the corresponding processing costs, and the value generated by those options that constitute terminal processing steps are also listed in Table 1. On the other hand, the expected values resulting from the various disassembly steps depend on the classification distribution(s) associated with the components generated by that step, as well as the subsequent actions taken, and therefore, they are not part of the data listed in Table 1. Instead, the *optimized* set for these values can be computed by applying the DP algorithm, expressed by Equations 1 – 4, on the E-DPN model of the considered disassembly process; the E-DPN model for this example, the optimal value function π^*, and the corresponding optimal disassembly policy are depicted in Figure 2.

In Figure 2, places in P_R, corresponding to unclassified units, including the original product unit itself, are depicted with dotted lines; places in P_C, corresponding to units classified (through testing) to one of the two recognized quality classes, are depicted with dashed lines; finally, places in P_L, corresponding to terminal operations, are depicted with solid lines. For referential purposes, each place is uniquely identified by a label annotated within the corresponding circle. In particular, the dashed-line places, that correspond to classified units, are characterized by a label with its first part being defined by the corresponding item ID # – as specified in the 2nd column of Table 1 – while its last digit is the quality class code: 1 for a repairable condition and 2 for worn-out. Dotted-line places, that correspond to unclassified units, are characterized by the concatenation of the item ID # and the label of the parental classified node, whose disassembly generated the considered item; however, in the case of item 0, that corresponds to the original product unit, the second part of the label does not exist, and it is denoted by '-'. Finally, solid-

line places, that correspond to terminal operations, are characterized by the abbreviation of the corresponding operation. The numbers attached to transitions in $T_D \cup T_P$ – that correspond, respectively, to disassembly and terminal operations for classified units – represent the (expected) cost of the corresponding operation, provided in Table 1 (first entry in the pair accompanying each operation). The numbers attached to transitions in T_C – that represent the classification of the unit associated with their input place to the quality class associated with their output place – are the relevant classification probabilities, provided by Table 2. Finally, the numbers attached to the various places represent their optimal value, $\pi^*(p)$. For places in P_L, corresponding to terminal operations, these values are provided in Table 1 (second entry in the pair accompanying each operation). For the remaining interior places, these values are obtained according to the DP logic expressed by Equations 2 and 3. For instance, the optimal value of place p_{71}, that corresponds to item 7 in repairable condition, is computed, according to Equation 3 as: $\pi^*(p_{71}) = \max\{70-40, 40-30, 56-20, 0-10\} = 36$. On the other hand, the value of place p_{741}, that corresponds to an unclassified unit of item 7, obtained from the disassembly of a unit of item 4 in condition 1 (repairable), is computed, according to Equation 2, as: $\pi^*(p_{741}) = 1.0 \cdot \pi^*(p_{71}) = 1.0 \cdot 36 = 36$. Similarly, one can find that $\pi^*(p_{81}) = \max\{80-30, 50-13, 0-10\} = 50$ and that $\pi^*(p_{841}) = 1.0 \cdot \pi^*(p_{81}) = 1.0 \cdot 50 = 50$. Then, the value of place p_{41}, that corresponds to one unit of component 4 in condition 1 (repairable), can be obtained, according to Equation 3, as: $\pi^*(p_{41}) = \max\{60-17, 36+50-10, 0-20\} = 76$. Furthermore, the logic of Equation 4 indicates that, based on the above calculations, the respective optimal options for p_{71}, p_{81} and p_{41} are RECYCLING, UPGRADING and DISASSEMBLY. Working in this fashion, from the terminal nodes of the E-DPN graph towards its source node, one can obtain the optimal value for each node and the entire optimal disassembly plan; the latter is depicted by the thicker transitions in Figure 2, and it consists of the options listed in boldface characters in Table 1. $\diamond$

The presented example, and also Equations 1–4, provide a complete characterization of the data set needed for the computation of the optimal disassembly plan through the basic DP algorithm. In particular, this data set includes the expected costs and revenues associated with the various processing stages, as well as the classification probabilities for the extracted components and sub-assemblies (c.f. Tables 1 and 2). Yet, as it was remarked in the introductory section, data like the item classification probabilities will be hard to estimate a priori, since they are determined by the process input stream, which, in turn, consists of items exposed to uncontrollable and unobservable consumer behaviors. In a similar vein, the expected costs and revenues associated with the various processing stages can be significantly affected by the quality status of the processed material, which again implies that they will not be amenable to

guesstimate mechanisms. Furthermore, many of these parameters can be varying with time, as they will be affected by shifts and drifts of the prevailing operational and the market conditions. The next section discusses how these data-related issues can be addressed by *reinforcement learning* algorithms, that augment the basic DP logic with learning and adaptive capabilities.

3 Managing the ODP uncertainty through Reinforcement Learning

3.1 Reinforcement Learning: The general framework

Reinforcement learning (RL) theory is a paradigm developed by the machine learning and the broader artificial intelligence community in an effort to design algorithms that will allow systems to "learn how to make good decisions by observing their own behavior, and use built-in mechanisms for improving their actions through a *reinforcement* mechanism" [19]. The basic structure of any RL implementation is depicted by the block diagram of Figure 3: A controlled *plant* evolves in a *discrete state space* through the execution of a sequence of *actions* commanded to it by a *learning controller*. The execution of an action at the running plant state causes the *transition* of the plant to a new state, and it also generates an *immediate reward* or *reinforcement feedback* that is a function of the state-action pair. The intention of the learning controller is to select the actions to be commanded at every plant state in a way that maximizes some *objective function* of the sequence of the collected rewards. In the most typical RL implementations, the optimal action selection scheme can be characterized by an *optimal value function* that associates an (expected) *value* with every state-action pair, such that the optimal actions for any given state are the maximizers of the restriction of this value function to that state. Hence, given a plant and an associated objective function, the RL controller tries to identify an optimal policy for them by *"learning"* the corresponding optimal value function. More specifically, the learning controller maintains an *estimate* of this value function, that is *initialized* to some arbitrary set of values, and it is subsequently *updated* every time that a new reward observation is obtained, in a way that brings the maintained value estimates closer to the value function corresponding to the observed plant behavior. On the other hand, the running estimate of the optimal value function affects the action selection process itself, since, at each *decision epoch*, actions are selected in a way that seeks to balance the conflicting objectives of (i) maximizing the resulting value, as perceived by the aforementioned estimate of the optimal value function, and (ii) enhancing the quality of this estimate through further exploration over the plant state-action space; this last

conflict is known in the relevant terminology as the *"exploration vs. exploitation"* dilemma.

The theory of RL algorithms was substantially strengthened by the realization that, in their basic definition, many of these algorithms essentially constitute stochastic approximations of some more classical DP algorithms. More specifically, under their DP-based interpretation, RL algorithms essentially seek to compute an underlying optimal value function, $\pi^*(\cdot)$, in an iterative fashion that constitutes the *Robbins-Monro stochastic approximation* of some DP recursion defined according to Bellman's equation; the reader is referred to ([1]: Chapters 4 and 5) for the more technical details. The interpretation and study of RL theory in the prism of this finding has lead to a more profound understanding of the underlying learning mechanisms, and eventually, to the broader dissemination and acceptance of the field. Next we employ this connection between DP and RL theory, in order to transform the DP recursion for the ODP problem, developed in Section 2, to a RL algorithm; this part of our work will focus on a particular class of RL algorithms known as *Q-learning* algorithms [22].

3.2 Q-learning implementation for the ODP problem

When viewed in the aforementioned DP context, the defining property of the Q-learning algorithms is that the optimal value function learned by them is not the optimal state value function, $\pi^*(\cdot)$, itself, but a refinement of it known as the (problem) Q-factors; these Q-factors are defined for each state-action pair (i, u), such that the *optimal* Q-factor values – to be denoted by $Q^*(i, u)$ – express *the expected (total) value that results by selecting action u at state i and following the optimal policy thereafter* [22, 1]. Obviously,

$$\pi^*(i) = \max_u \{Q^*(i, u)\} \tag{5}$$

In the E-DPN representation for the ODP problem developed in Section 2, the primary decision points are represented by places $p \in P_C$. Hence, a set of Q-factors adequate to implement the Q-learning algorithm in the considered problem context, is defined by the pairs $(p, t),\ p \in P_C$ and $t \in p^\bullet$. Furthermore, the above interpretation of the optimal Q-factor values, $Q^*(p, t)$, implies that, in the context of the ODP problem, they must satisfy the following Bellman equation:

$$\begin{array}{ll} \forall p \in P_C, & \forall t \in p^\bullet, \\ Q^*(p,t) & = \begin{cases} \sum_{q \in t^\bullet} (F(t,q) \cdot \pi^*(q)) - \tau(t), \\ \text{for } t \in T_P \\ \sum_{q \in t^\bullet} (F(t,q) \cdot \sum_{\psi \in q^\bullet} (\rho(q,\psi) \cdot \max_{u \in (\psi^\bullet)^\bullet} \{Q^*(\psi^\bullet, u)\})) - \tau(t), \\ \text{for } t \in T_D \end{cases} \end{array} \tag{6}$$

In plain words, Equation 6 can be interpreted as follows: The optimal Q-factor values for transitions corresponding to terminal operations for some extracted item, are equal to the expected (monetary) value resulting from those operations minus the processing costs involved. On the other hand, the optimal Q-factor value for transitions modelling disassembly operations is determined by the cumulative expected value of the derived components, where the expectation is taken with respect to the classification probabilities of the derived items, and the values of the various quality classes are determined by the optimal Q-factor values themselves, according to Equation 5.

According to the general discussion of RL algorithms provided in Section 3.1, a Q-learning implementation for the ODP problem will try to develop accurate estimates, $Q(p,t)$, of the optimal Q-factor values, $Q^*(p,t)$, by exploiting the information contained in the sequence of the immediate rewards generated by the plant. In the ODP problem context, these immediate rewards are defined by the returns generated every time that a certain artifact in some quality class $p \in P_C$ is processed through an option $t \in p^\bullet$. Upon the generation of such a reward, the algorithm will update the corresponding Q-factor estimate, $Q(p,t)$, by employing the following *Robins-Monro stochastic approximation* of Equation 6:

$$
\begin{array}{lll}
\forall p \in P_C, & \forall t \in p^\bullet, & \\
Q(p,t) & := & (1-\gamma)Q(p,t) + \\
 & & \gamma \left\{ \begin{array}{l} \sum_{q \in t^\bullet} (\hat{F}(t,q) \cdot \hat{\pi}(q)) - \hat{\tau}(t), \\ \text{for } t \in T_P \\ \sum_{q \in t^\bullet} (\hat{F}(t,q) \cdot \sum_{\psi \in q^\bullet} (\hat{\rho}(q,\psi) \cdot \max_{u \in (\psi^\bullet)^\bullet} \{Q(\psi^\bullet, u)\})) - \hat{\tau}(t), \\ \text{for } t \in T_D \end{array} \right.
\end{array}
\tag{7}
$$

The quantities $\hat{\pi}$ and $\hat{\tau}$, that appear in Equation 7, denote respectively the (observed) returned revenue and processing cost applying to that instance. In the same spirit, $\hat{F}(t,q)$ denotes the number of units of the type modelled by place q, that were actually obtained during the applied processing step, represented by transition t. In the case of a disassembly operation, $\hat{\rho}(q,\xi)$ denotes the *percentage* of the obtained $\hat{F}(t,q)$ units that were eventually classified in the class represented by transition $\xi \in q^\bullet$. Finally, γ is an implementational parameter known as the algorithm *learning rate*; it must have a value in the interval $(0,1]$, and it can be interpreted

as the "percentage" of the *"error"* term[5]

$$\left\{ \begin{array}{l} \sum_{q\in t^\bullet}(\hat{F}(t,q)\cdot\hat{\pi}(q)) - \hat{\tau}(t), \\ \text{for } t \in T_P \\ \sum_{q\in t^\bullet}(\hat{F}(t,q)\cdot\sum_{\psi\in q^\bullet}(\hat{\rho}(q,\psi)\cdot \max_{u\in(\psi^\bullet)^\bullet}\{Q(\psi^\bullet,u)\})) - \hat{\tau}(t), \\ \text{for } t \in T_D \end{array} \right\} - \; Q(p,t) \tag{8}$$

that must be added to the current value of the $Q(p,t)$ factor in order to obtain the updated estimate.

It should be clear from the above interpretation of the various parameters involved in the updating expression of Equation 7, that all the relevant data can be obtained from direct observation of the system operation at each processing cycle, and therefore, in case that the Q(p,t) estimates converge to the optimal $Q^*(p,t)$ values, the presented algorithm has indeed the potential to establish optimal operation despite the lack of an explicit model for the system behavior. The next theorem establishes that for the ODP problem version introduced in Section 2, this convergence will always take place, provided that the algorithm implementation satisfies some additional conditions.

Theorem 1 *Consider the implementation of the Q-learning algorithm for the ODP problem, defined by the updating scheme of Equation 7, and further suppose that:*

1. *For every place $p \in P_C$ and transition $t \in p^\bullet$, the sequence of learning rates $\gamma_k(p,t)$ k=1,2,..., utilized in the updates of the estimate $Q(p,t)$, satisfies the following two equations:*

$$\sum_{k=1}^{\infty} \gamma_k(p,t) \;=\; \infty \tag{9}$$

$$\sum_{k=1}^{\infty} \gamma_k^2(p,t) \;<\; \infty \tag{10}$$

2. *For every component class $p \in P_C$, the algorithm selects every processing option $t \in p^\bullet$, an* infinite *number of times.*

Then, the algorithm estimates, $Q(p,t)$, will converge to the corresponding optimal values, $Q^*(p,t)$, with probability 1, and irrespectively of the initializing values of the $Q(p,t)$ estimates.

[5]The expression of Equation 8 can be interpreted as the difference between the Q^*-value of the (p,t) pair assessed based on the currently experienced results of executing option t on an item of the quality class p, and the available $Q(p,t)$ estimate.

Proof: The proof of Theorem 1 results immediately from Proposition 5.5 of [1], when noticing that the E-DPN structure, introduced in Section 2, implies that, starting from the initial marking m_0, every execution of the learning algorithm will result in a terminal marking, m, in a finite number of steps, and therefore, in the terminology of [1], every feasible disassembly policy is *proper* (c.f. Definition 2.1 in [1]). $\diamond$

Conditions 1 and 2 of Theorem 1 are necessary in order to ensure that (i) the algorithm implementation allows for sufficient exploration, and that (ii) the algorithm convergence is not compromised by the stochasticity inherent in the observed rewards. A practical way to guarantee the requirements of the first condition, is by having the learning rates, $\gamma_k(p,t)$, decrease asymptotically to zero, according to the following schedule:

$$\gamma_k(p,t) := b/(a+k) \tag{11}$$

where a and b are positive constants. On the other hand, Condition 2 can be enforced by introducing a small positive parameter $\epsilon << 1.0$, and adopting an action selection scheme that, at every decision cycle, selects an action corresponding to a maximal $Q(p,t)$ estimate [6] with probability $1-\epsilon$, and an alternative random action with probability ϵ.[7] Hence, both Conditions 1 and 2 can be effectively satisfied during the algorithm implementation, and therefore, they do not constrain its applicability.

3.3 Example

In order to demonstrate the ability of the Q-learning algorithm, defined by Equation 7, to determine an optimal policy for the ODP problem, and to elucidate the dynamics of the learning process as it converges to the optimal value function, we applied it to the ODP example discussed in Section 2. The presented implementation satisfied Conditions 1 and Condition 2 of Theorem 1 according to the mechanisms delineated in the previous paragraph; specifically, the learning rate used for the k-th updating of the $Q(p,t)$ value, $p \in P_C,\ t \in p^\bullet$, was $\gamma_k = 0.3/(1+k/1000)$, while the value of the randomizing parameter ϵ was set to 0.2. Furthermore the initial estimates of the Q-factors were set to zero.

Figures 4 and 5 depict for two indicatively selected places, the evolution of the Q^* values learned by the algorithm. In the reported results, each trial corresponds to the processing of a single product unit. As it is expected from Theorem 1, the juxtaposition of Figures 4 and 5 with

[6] such an action selection scheme is characterized as *"greedy"* in the relevant terminology

[7] For a more extensive discussion on the "exploration versus exploitation" problem and some additional ways to address it, the reader is referred to [19, 1]

the π^* values reported in Figure 2 verifies that the implemented algorithm converges indeed to the optimal Q values for the depicted state-action pairs. At the same time, these figures reveal the significance of exploration in the learning dynamics; for instance, it can be seen in Figure 5 that it took a number of "non-greedy" selections – corresponding to the jumps taking place in the relevant curve – before recycling emerged as the dominant option for the corresponding stage. Figure 6 depicts the monetary value accrued by the disposition of the 2000 product units through the processing options selected by the algorithm. The positive impact of the underlying learning process on this quantity is revealed by the acceleration with which this value is accumulated over the different trials; for instance, while the total value accumulated during the processing of the first 1000 units is around \$85,000, the corresponding value for the next 1000 units is around \$135,000, an increase by a factor of 1.6. $\diamond$

4 Practical Considerations

The previous section modelled the ODP problem as a learning process, and established that Q-learning constitutes an effective algorithm to support the required learning. In this section we discuss some more practical issues concerning the implementation of the proposed algorithm in an actual remanufacturing facility. These issues concern: (i) the characterization of the computational and operational infrastructure necessary to support the implementation of the learning process expressed by Equation 7; (ii) the investigation of the possibility of assisting the learning task by integrating in it any a priori available, although partial, information about the costs and returns to be expected by the various processing options; and (iii) the extension of the algorithm so that it can effectively deal with potential process non-stationarity. We deal with each of these issues in a separate subsection.

4.1 Implementing Q-learning in a remanufacturing facility

We envision the underlying operational environment as a disassembly process dedicated to a particular product type. This process is continuously fed with reclaimed units of the considered part type, each of which is disassembled according to a plan that is determined on-line by the randomized action selection scheme introduced in Section 3.2, based on the currently available Q-values for the different item-option pairs. At the same time, the outcomes of the executed processing steps provide the data for the updating of the maintained Q-values, according to the logic expressed in Equation 7.

From a computational standpoint, the above mechanism is very efficient since the only thing that it requires is the regimented updating of the Q-value corresponding to every selected item-option pair according to a very simple and straightforward calculation.[8] A more pragmatic concern, however, is the extent to which the underlying disassembly process and its broader operational context are adequately flexible to support the revision of the applied disassembly plan on a unit-by-unit basis. If such an operational scheme is not deemed feasible, then the entire operational logic outlined above is still implementable on a *batch*-based mode; i.e., a constant disassembly plan is selected and applied on an entire batch of reclaimed units, while the batch sizes are selected large enough to ensure the operational stability of the facility. The Q-values of the item-option pairs appearing in the executed plan will still be updated according to the logic of Equation 7; however, this updating will take place upon the completion of the entire batch, and it will employ the *batch-means* for the various quantities appearing in Equation 7. Furthermore, to ensure a more expedient convergence for this version of the algorithm, one should perform these Q-value updates working from the leaf nodes towards the source node of the underlying E-DPN.

A last concern regards the extent to which the assumption of providing a dedicated facility to a single product unit reflects the prevailing industry practice. Based on some investigation of the industry trends reported in [2, 17], we believe that the industry will soon present the necessary economies of scale for adopting such a product-focused lay-out; the emerging trend of outsourcing the recycling function to 3rd party service providers is a major step in that direction. Furthermore, the aforementioned assumption is not strictly required for the effective implementation of the proposed learning mechanism. The algorithm is also implementable in time-shared facilities as long as (i) a different set of data structures is employed and maintained for each (re-)processed product type, and (ii) the facility provides ample capacity for the timely reprocessing of all the returned product units. On the other hand, if the various product types are contesting for the processing capacity of the remanufacturing facility, then, this effect introduces an additional resource allocation constraint in the original problem formulation, and necessitates the re-investigation of the problem under this modified set of assumptions.

We conclude this discussion on the implementability of the proposed learning framework, by noticing that it is very similar, in spirit, to the framework of *statistical process control (SPC)*

[8] Actually, a more careful consideration of this updating mechanism and of the content of Equation 7 will reveal that they present a very strong similarity to the updating mechanism employed by the *exponential smoothing*-based forecasting [14]; hence, all the advantages and computational efficiency that are typically associated with that forecasting methodology can also be attributed to the proposed implementation of Q-learning.

[3], that has been applied to the more traditional manufacturing processes. In both cases, the ultimate objective is to obtain a higher value from the underlying process, by applying tighter and more systematic control on it, that is enabled by the information provided through an effective monitoring function. As established by the experience of the SPC revolution of the late 80's / early 90's, the effective deployment of such a control capability is a challenging task, since it requires a very disciplined operation and a well-understood and managed process, but it can result in very substantial gains!

4.2 Eliminating sub-optimal options based on partial cost / return information

It should be obvious from the previous discussion that the number of trials required for the eventual convergence of the Q-learning algorithm to the optimal policy depends strongly on the size of the underlying E-DPN structure. Therefore, any effort to reduce the size of this E-DPN structure by identifying and eliminating processing options that are going to be suboptimal can have a considerable pay-off in terms of the convergence rate of the applied algorithm and the total value extracted by the underlying disassembly process. Indeed, in many cases it is reasonable to assume some a priori knowledge regarding the cost of the various processing options and the expected returns, expressed by a set of reliable lower and upper bounds for each of these quantities. More formally, we can assume that for each cost element $\tau(t),\ t \in T_P \cup T_D$, we are given a lower and an upper bound, respectively denoted by $\underline{\tau}(t)$ and $\overline{\tau}(t)$. Similarly, for each processing option $p \in P_L$, we are given a lower and an upper bound for the expected return, respectively denoted by $\underline{\xi}(p)$ and $\overline{\xi}(p)$. This information can subsequently enable an E-DPN reduction scheme that will (i) compute lower and upper bounds, $\underline{Q}(p,t)$ and $\overline{Q}(p,t)$, for all $p \in P_C,\ t \in p^\bullet$, and (ii) eliminate from the E-DPN structure those actions $t \in p^\bullet,\ p \in P_C$, for which $\exists t' \in p^\bullet$ s.t. $\underline{Q}(p,t') > \overline{Q}(p,t)$.

The computation of the lower and upper bounds, $\underline{Q}(p,t)$ and $\overline{Q}(p,t)$, for all $p \in P_C,\ t \in p^\bullet$, that is involved in Step (i) above, can be performed through the following recursion that proceeds from the E-DPN leaf nodes towards its source node, and can be derived straightforwardly from Equations 1–6, that constitute the DP characterization of ODP problem:

$$\forall p \in P_L,\quad \underline{\pi}(p) := \underline{\xi}(p);\quad \overline{\pi}(p) := \overline{\xi}(p) \tag{12}$$

$$\forall p \in P_R,\quad \underline{\pi}(p) := \min_{t \in p^\bullet}\{\underline{\pi}(t^\bullet)\};\quad \overline{\pi}(p) := \max_{t \in p^\bullet}\{\overline{\pi}(t^\bullet)\} \tag{13}$$

$$\forall p \in P_C,\ \forall t \in p^\bullet, \quad \underline{Q}(p,t) := \sum_{q \in t^\bullet} (F(t,q)\underline{\pi}(q)) - \overline{\tau}(t); \quad \overline{Q}(p,t) := \sum_{q \in t^\bullet} (F(t,q)\overline{\pi}(q)) - \underline{\tau}(t) \quad (14)$$

$$\forall p \in P_C, \quad p^\bullet_{eff} := \{t \in p^\bullet : \nexists t' \in p^\bullet \ s.t.\ \underline{Q}(p,t') > \overline{Q}(p,t)\} \quad (15)$$

$$\forall p \in P_C, \quad \underline{\pi}(p) := \max_{t \in p^\bullet_{eff}} \{\underline{Q}(p,t)\}; \quad \overline{\pi}(p) := \max_{t \in p^\bullet_{eff}} \{\overline{Q}(p,t)\} \quad (16)$$

4.3 Dealing with non-stationary processes

Throughout the previous discussion it has been assumed that the unknown parameters will remain constant during the entire operation of the ODP process. In reality, however, it is possible that (some of) these parameters will present significant variation during the process life cycle. For instance, the various cost and return parameters will be determined by the prevailing market conditions, which might significantly evolve during the process life cycle. Similarly, the classification probabilities for the various components and sub-assemblies might be different for different input batches, obtained from different sources. A sound implementation of the proposed algorithm must be aware of the various non-stationarities that can potentially arise in the considered operational environment, and provide the mechanisms to control the impact of these non-stationarities on the process performance. Hence, in the subsequent discussion, we provide some "rules-of-thumb" that will minimize the adversarial impact of these non-stationarities on the process performance.

When dealing with the potential process non-stationarities, it is pertinent to discriminate between (i) major abrupt *"shifts"*, and (ii) slow, yet considerable, *"drifts"* for (some of) the process parameters. Regarding the former, we expect that (almost all of) these changes will be caused by some specific source event(s) taking place in the process operational or business environment, and therefore, they should be immediately foreseen and anticipated by an alert process management team. In that case, the problem reduces to the effective and efficient re-learning of an optimal policy, under the new prevailing conditions. This can be readily done by *"re-setting"* the entire learning process, either to some "default" initial Q values, or to some Q values that are deemed to reflect the experienced situation. In some other cases, these parameter shifts can result from a systematic rotation of the process through a number of operational modes; for instance, the process might systematically rotate among the processing of batches coming from different input streams with different quality classification probabilities.

In this case, it is advisable that the algorithm maintains a *"bank"* of different sets of Q values, each applicable to a specific operational mode. Finally, in order to guard against any major parametric shifts that can occur in an unexpected and otherwise undetected fashion, one can incorporate in the process some *"error-tracking"* mechanism that will monitor the estimation errors generated by the various item-option pairs according to Equation 8, and provide an *"alert"* signal in case that these errors are unexpectedly and persistently increased. Foregoing the technical details due to space limitations, we notice that this idea can be implemented through some statistical analysis of these errors and the creation of appropriate confidence intervals for them, similar to the case of exponential smoothing models.

The second type of non-stationarity mentioned above is more insidious and therefore more difficult to detect based on external input. To deal effectively with it, the adopted implementation of the proposed learning algorithm must present a high degree of *alertness to change* and *learning flexibility*. This conceptual requirement suggests, in turn, an increased value for the exploration-controlling parameter, ϵ, and the preservation of a high learning rate, γ_k; a practical way to satisfy this last requirement is by eliminating the dependence of the learning rate on the trial number k, and setting it on a fairly high constant level.

5 Conclusions

The starting point and the major motivation for the work presented in this paper was the observation that the effective management of the uncertainties inherent in the emerging remanufacturing processes has not been adequately addressed in the relevant literature, even though it is currently recognized as an essential issue for the process viability. Hence, the presented work undertook the problem of uncertainty management, as it arises in the context of the optimal disassembly planning, one of the key tasks to be resolved for the efficient operation of the aforementioned processes. Using recently emerged results from (approximate) dynamic programming and machine learning, this work provided a rigorous framework for the modelling and analysis of the ODP problem in the face of the aforementioned uncertainties, and an effective and easily implementable computational algorithm for obtaining optimal disassembly policies. The last part of the paper considered some additional practical issues that need to be addressed for the effective implementation of the proposed algorithm in any "real-life" remanufacturing facility.

As part of our future work, we shall seek to implement the presented algorithm on an industrial-scale remanufacturing process. On the more theoretical side, we are currently inves-

tigating a number of issues that can lead to (i) more expedient convergence of the Q-learning algorithm in the considered application context and (ii) more enhanced responsiveness to any experienced changes in the process parameters. Some mechanisms that can lead to such enhanced performance include (i) the exploitation of the *acyclic* E-DPN structure in the management of the "exploration vs. exploitation dilemma", and (ii) the integration to the learning framework developed in this work of some further modelling capability that will seek to explicitly learn an approximating model of the underlying process dynamics and the associated cost structure. Finally, as it was mentioned in Section 4.1, another interesting problem is the extension of the developed methodology so that it applies to *resource-constrained, multi-product* disassembly processes, i.e., processes that disassemble two or more product types which contest for a shared set of resources. The work of [15] could be a good starting point for this task; the systematic understanding and the formal characterization of the process economics and dynamics underlying this new operational setting, in a way that enables the development of an on-line / incremental process optimization algorithm, are also part of our research agenda.

Acknowledgement

This work was partially supported by NSF grant DMII-0318657 and the Logistics Institute - Asia Pacific (TLI-AP). The author would also like to acknowledge the help of Gopal Jayaram and Sagar Kalyankar in the development of the MATLAB code and the execution of the simulated experiment reported in this work.

References

[1] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, Belmont, MA, 1996.

[2] D. Cotton and S. A. Reveliotis. An overview of the legislation and applications of reverse logistics relating to product take back. Technical report, School of Industrial & Systems Eng., Georgia Tech, 2003.

[3] R. E. De Vor, T-H. Chang, and J. W. Sutherland. *Statistical Quality, Design and Control.* Prentice Hall, 1992.

[4] G. Erdos, T. Kis, and P. Xirouchakis. Modeling and evaluating product end-of-life options. *IJPR*, (to appear).

[5] M. Fleischmann, J. M. Bloemhof-Ruwaard, R. Dekker, E. Van der Laan, J. A. E. E. Van Nunen, and L. N. Van Wassenhove. Quantitative models for reverse logistics: A review. *EJOR*, 103:1–17, 1997.

[6] D. Geiger and E. Zussman. Probabilistic reactive disassembly planning. *Annals of CIRP*, 45:49–52, 1996.

[7] A. Gungor and S. M. Gupta. Disassembly sequence planning for products with defective parts in product recovery. *Computers & Industrial Eng.*, 35:161–164, 1998.

[8] L. S. Homen de Mello and A. C. Sanderson. And/or graph representation of assembly plans. *IEEE Trans. on R&A*, 6:188–199, 1990.

[9] H. R. Krikke, A. Van Harten, and P. C. Schuur. On a medium term product recovery and disposal strategy for durable assembly products. *IJPR*, 36:111–139, 1998.

[10] A. Lambert. Determining optimum disassembly sequences in electronic equipment. *Computers & Industrial Engineering*, 43:553–575, 2002.

[11] A. Lambert. Disassembly sequencing: a survey. *Intl. Jrnl of Prod. Res.*, 41:3721–3759, 2003.

[12] D.-H. Lee, J.-G. Kang, and P. Xirouchakis. Disassembly planning and scheduling: Review and further research. *Proc. IMechE*, 215B:695–709, 2001.

[13] C. G. Looney. Fuzzy petri nets for rule-based decision making. *IEEE Trans. on SMC – Part B*, 18:178–183, 1988.

[14] S. G. Makridakis and S. C. Wheelwright. *The Handbook of Forecasting*. John Wiley & Sons, NY,NY, 1985.

[15] A. Meacham, R. Uzsoy, and U. Venkatadri. Optimal disassembly configurations for single and multiple products. *Journal of Manufacturing Systems*, 18:311–322, 1999.

[16] T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[17] S. A. Reveliotis. Uncertainty management in optimal disassembly planning through learning-based strategies. In *Proceedings of the NSF–IEEE–ORSI Intl. Workshop on IT-enabled Manufacturing, Logistics and Supply Chain Management*, pages –. NSF/IEEE/ORSI, 2003.

[18] N. Salomonski and E. Zussman. On-line predictive model for disassembly process planning adaptation. *Robotics and Computer Integrated Manufacturing*, 15:211–220, 1999.

[19] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, Cambridge, MA, 2000.

[20] Y. Tang, M. Zhou, and R.b Caudill. An integrated approach to disassembly planning and demanufacturing operation. *IEEE Trans. on R&A*, 17:773–784, 2001.

[21] Y. Tang, M. Zhou, E. Zussman, and R.b Caudill. Disassembly modeling, planning and application. *Journal of Manufacturing Systems*, 21:200–217, 2002.

[22] C.J.C.H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, UK, 1989.

[23] M. Zhou and K. Venkatesh. *Modeling, Simulation and Control of Flexible Manufacturing Systems: A Petri Net Approach*. World Scientific, Singapore, 1998.

[24] E. Zussman, A. Kriwet, and G. Seliger. Disassembly-oriented assessment methodology to support design for recycling. *Annals of CIRP*, 43:9–14, 1994.

[25] E. Zussman and M. Zhou. A methodology for modeling and adaptive planning of disassembly processes. *IEEE Trans. on R&A*, 15:190–194, 1999.

[26] E. Zussman and M. Zhou. Design and implementation of an adaptive process planner for disassembly processes. *IEEE Trans. on R&A*, 16:171–179, 2000.

Spyros A. Reveliotis received his Ph.D. in Industrial Engineering from the University of Illinois at Urbana-Champaign in 1996. He also holds an MS degree in Electrical & Computer Systems Engineering. Currently, he is an Associate Professor of the School of Industrial & Systems Engineering, at the Georgia Institute of Technology.

Dr. Reveliotis' research interests are in the area of Discrete Event Systems theory and its applications. His current research focuses on: (i) the logical / structural control of flexibly automated production systems, workflow management systems and guidepath-based traffic systems; (ii) the development of effective scheduling policies for these environments; and (iii) the development of customized algorithms and implementations of approximate dynamic programming and machine learning theory for the aforementioned application contexts.

Dr. Reveliotis is a senior member of IEEE and a member of IIE and INFORMS. He is also a member of the IFAC technical committee on Discrete Event Systems, and a member of the College-Industry Council on Material Handling Education. He has been an Associate Editor for the IEEE Trans. on Robotics & Automation, and he is currently serving as an Associate Editor for the IEEE Trans. on Automation Science & Engineering. Finally, he was the recipient of the Kayamori Best Paper Award in the 1998 IEEE Intl. Conference on Robotics & Automation.

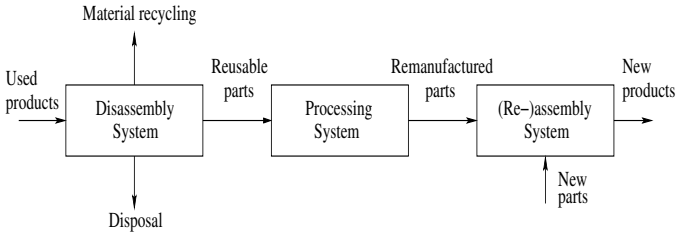

Figure 1: The typical material flow in reverse logistics systems – adapted from [12]

.

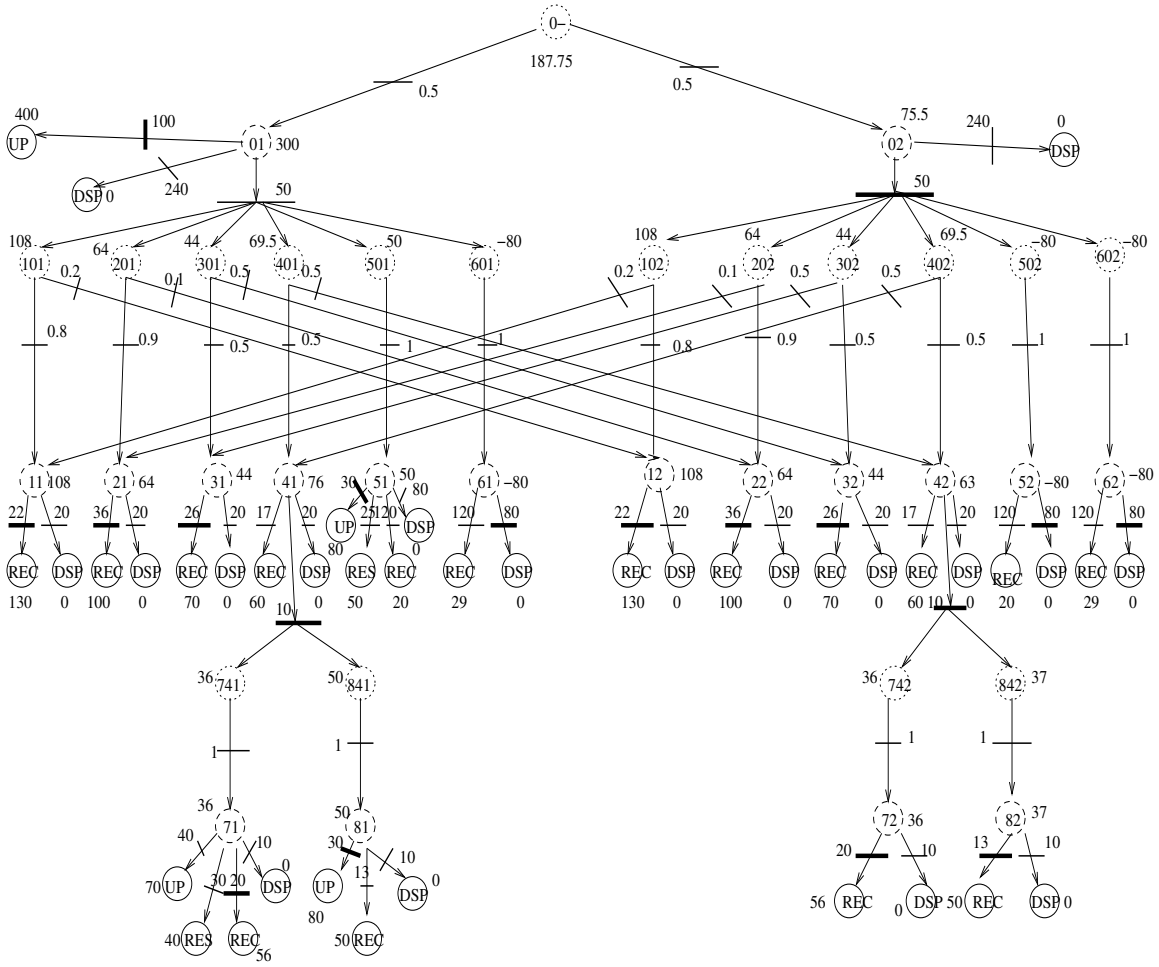

Figure 2: The E-DPN model and the optimal value function for the considered case study.

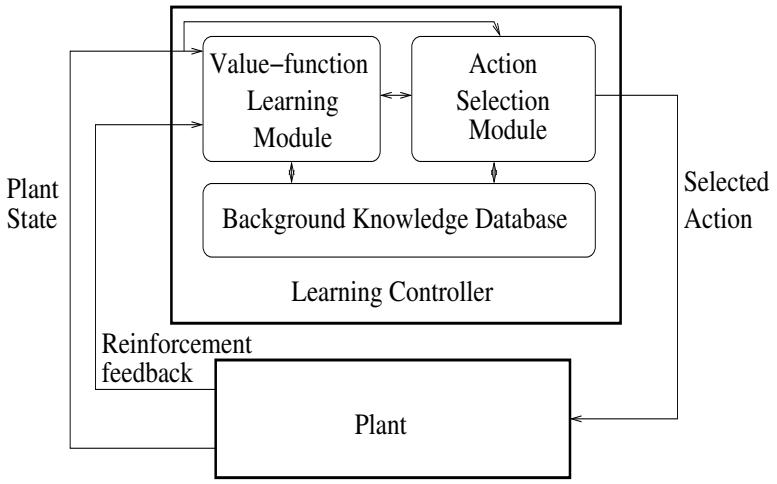

Figure 3: The basic architecture of a RL-based controller

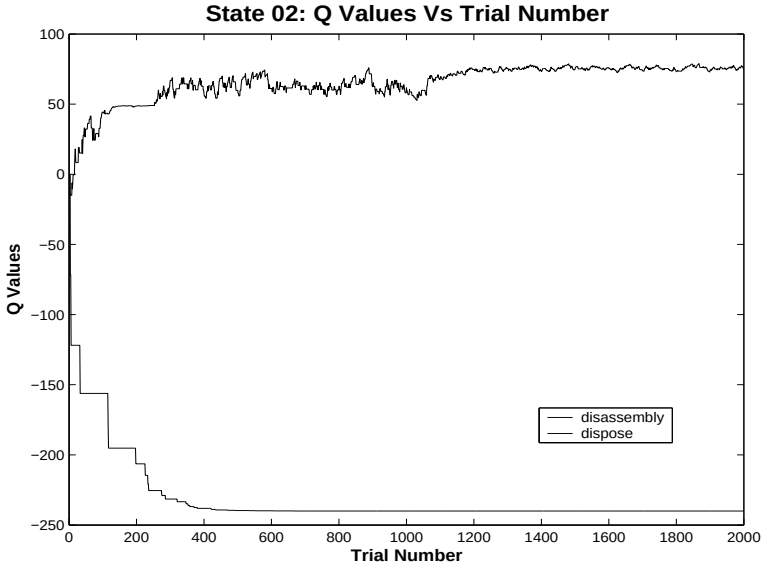

Figure 4: Example: Learning the $Q^*(p,t)$ values for place p_{02} ($\epsilon = 0.2$; $\gamma_k = 0.3/(1 + k/1000)$; $Q^0(p,t) = 0,\ \forall p \in P_C,\ \forall t \in p^\bullet$)

.

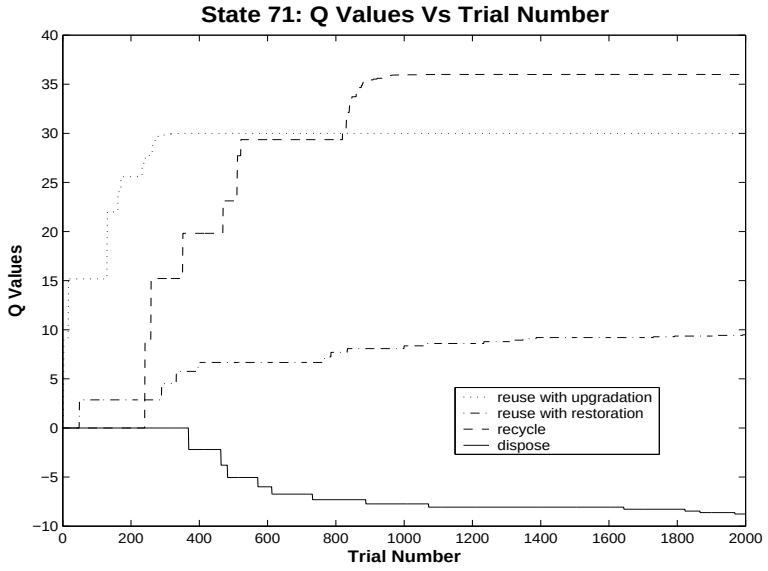

Figure 5: Example: Learning the $Q^*(p,t)$ values for place p_{71} ($\epsilon = 0.2$; $\gamma_k = 0.3/(1 + k/1000)$; $Q^0(p,t) = 0,\ \forall p \in P_C,\ \forall t \in p^\bullet$)

.

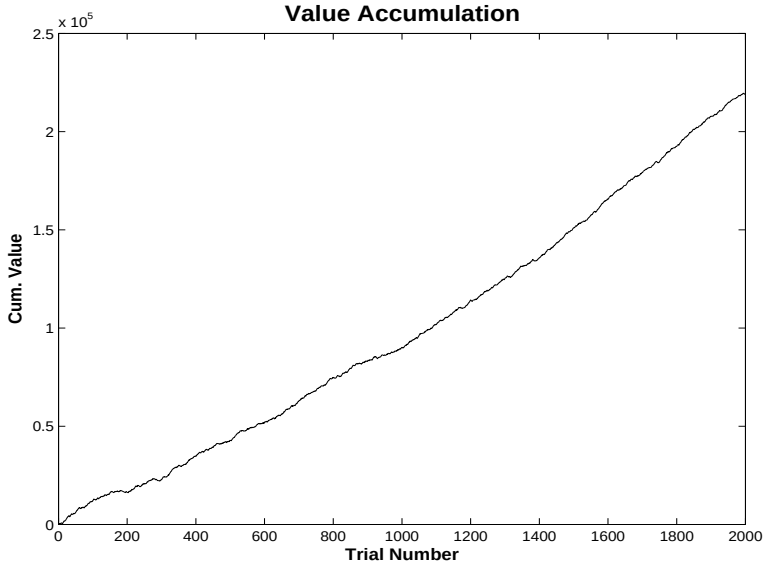

Figure 6: Example: The accumulation of the extracted value ($\epsilon = 0.2$; $\gamma_k = 0.3/(1 + k/1000)$; $Q^0(p,t) = 0,\ \forall p \in P_C,\ \forall t \in p^\bullet$)

.

Table 1: The example data; processing options annotated in boldface represent the optimal disassembly plan.

BOM Level	Component ID #	Component Name	Quality Class (Code)	Viable Options (proc. cost, value generated)
0	0	TV	Repairable (1)	DSBL(50,·), **UP**(100,400), DISP(240,0)
			Worn Out (2)	**DSBL**(50,·), DISP(240,0)
1	1	Casing	Repairable (1)	**REC**(22,130), DISP(20,0)
			Worn Out (2)	**REC**(22,130), DISP(20,0)
1	2	Wiring	Repairable (1)	**REC**(36,100), DISP(20,0)
			Worn Out (2)	**REC**(36,100), DISP(20,0)
1	3	Trafo	Repairable (1)	**REC**(26,70), DISP(20,0)
			Worn Out (2)	**REC**(26,70), DISP(20,0)
1	4	PCB	Repairable (1)	**DSBL**(10,·), REC(17,60), DISP(20,0)
			Worn Out (2)	**DSBL**(10,·), REC(17,60), DISP(20,0)
2	7	CPU	Repairable (1)	UP(40,70), RES(30,40), **REC**(20,56), DISP(10,0)
			Worn Out (2)	**REC**(20,56), DISP(10,0)
2	8	Chip	Repairable (1)	**UP**(30,80), REC(13,50), DISP(10,0)
			Worn Out (2)	**REC**(13,50), DISP(10,0)
1	5	Battery	Repairable (1)	**UP**(30,80), RES(25,50), REC(120,20), DISP(80,0)
			Worn Out (2)	REC(120,20), **DISP**(80,0)
1	6	Tube	Repairable (1)	REC(120,29), **DISP**(80,0)
			Worn Out (2)	REC(120,29), **DISP**(80,0)

Table 2: Item Classification Probabilities: $\rho(i|j)$ denotes the probability that the considered item will be in class i given that its parent item was in class j; for the case of the original product, i.e., level 0 item, the corresponding classification probabilities are unconditional.

Level	Item ID #	$\rho(1\|-)$	$\rho(2\|-)$		
0	0	0.5	0.5		
Level	Item ID #	$\rho(1\|1)$	$\rho(2\|1)$	$\rho(1\|2)$	$\rho(2\|2)$
1	1	0.8	0.2	0.2	0.8
	2	0.9	0.1	0.1	0.9
	3	0.5	0.5	0.5	0.5
	4	0.5	0.5	0.5	0.5
	5	1	0	0	1
	6	1	0	0	1
2	7	1	0	0	1
	8	1	0	0	1