International Journal of Production Research

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tprs20

Modelling and evaluating product end-of-life options

G. Erdos , T Kis & P. Xirouchakis

Published online: 14 Nov 2010.

To cite this article: G. Erdos , T Kis & P. Xirouchakis (2001) Modelling and evaluating product end-of-life options, International Journal of Production Research, 39:6, 1203-1220, DOI: 10.1080/713845985

To link to this article: http://dx.doi.org/10.1080/713845985

INT. J. PROD. RES., 2001, VOL. 39, NO. 6, 1203–1220

Modelling and evaluating product end-of-life options

G. ERDOS†, T. KIS‡ and P. XIROUCHAKIS†*

The paper focuses on the modelling and evaluating product end-of-life options, which is the problem of representing products and determining disassembly sequences with the objective of maximizing revenue. For the problem considered here, three algorithms were developed. The first is the algorithm to generate the product recovery graph semi-automatically for a given product liaison graph. Then, using the generated product recovery graph, another algorithm is developed to obtain optimal disassembly plans that maximize revenue. This algorithm is based on the backward calculation so that the hyperedges of the recovery graph are visited only once. Finally, to cope with uncertainties of the end-of-life products, a recovery graph questioning algorithm is suggested to find the margin of allowed revenue reduction of a given target edge that maintains the same optimal plan. Application of the three algorithms is illustrated using an example.

1. Introduction

Environmentally conscious design and manufacturing (ECD&M) is a view of manufacturing that includes the social and technological aspects of the design, synthesis, processing, use and end-of-life of products in manufacturing (Zhang *et al*. 1997). The importance of ECD&M is steadily growing for product designers and manufacturers since it allows designers and manufacturers to minimize waste and to turn waste into a profitable product. This trend urges product developers to design better products from the point of view of a complete product life cycle, which is closely related to the cradle-to-grave approach. As a first step towards addressing the entire problem, the end-of-life phase is considered among the various stages of a product's life cycle. That is, this paper focuses on a subproblem of optimizing the product's end-of-life (EOL) options. During the design of a new product, there is a need for predictive planning to evaluate different EOL options and to adapt the product design for an optimized performance. The design solutions, i.e. product structure, material selection, part geometry and joint design, will then be dependent on the future EOL priorities.

Many methods have been developed for estimating product EOL. See Zussman *et al*. (1994) for description and calculation methods for various EOL options such as disposal, material recycling, reuse, separation and disassembly. Among the existing approaches, the most important ones are: (1) product structure independent approach and (2) product recovery plan approach. The main idea of the first approach is to calculate the EOL of individual components outside the context of a specific product structure. In spite of the wide applicability of this method, it does not take into account the effects of the context in which the element is embedded.

Revision received July 2000.

† Department of Mechanical Engineering and ‡Department of Mathematics, Swiss Federal Institute of Technology at Lausanne (EPFL), Lausanne, CH-1015, Switzerland.

* To whom correspondence should be addressed. e-mail: paul.xirouchakis@epfl.ch

International Journal of Production Research ISSN 0020–7543 print/ISSN 1366–588X online © 2001 Taylor & Francis Ltd
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540010009705

The second approach overcomes this shortcoming, because it specifies in detail how to disassemble the product and what are the EOL options of the resulting subassemblies and/or parts.

Most publications follow the product recovery plan approach, in which the alternative disassembly plans are represented by means of the AND/OR graph and its variations. The product recovery plan approach results in a more accurate evaluation of optimal EOL, but existing methods assume that the disassembly AND/OR graph is given. The automatic generation of disassembly sequences is rooted on the works of Bourjault (1984), Homem de Mello and Sanderson (1990, 1991) and Sanderson *et al*. (1990). Their methods are originally developed to generate assembly sequences, but they also incorporate the disassembly sequence generation. They use the liaisons between components registered in the connection diagram structure, which is called the liaison diagram, and a set of precedence rules to generate the possible disassembly sequences. Among them, Homem de Mello and Sanderson (1990) introduce the AND/OR graph to develop an algorithm that generate all assembly sequences, in which disassembly sequences are considered as the reversed ones of the corresponding assembly sequences. In particular, Kanehara *et al*. (1993) show several properties for the AND/OR graph using Petri-nets in order to use linear programming techniques for the assembly sequencing.

There are research articles on the disassembly sequencing problem. Johnson and Wang (1995) suggest an algorithm for the problem with the objective of maximizing profit. Also, they suggest an economic index to determine recovery or disposal of the product and four criteria to reduce the search space in the form of clustering. Johnson and Wang (1998) extend their model by including the two-commodity network problem to solve the disassembly sequencing. Penev and de Ron (1996) consider the problem of determining the disassembly level to maximize the revenue, in which the decision variable is whether more disassembly operations are required at each stage of disassembling of a product. Pnueli and Zussman (1997) suggest a dynamic programming algorithm for the disassembly sequencing that includes the end-of-life value of a product. Using this algorithm, they suggest several procedures for redesign-for recycling problem, in which weak spots in the design are identified and possible solutions are suggested. Kriwet *et al*. (1995) emphasized the redesign aspect of the disassembly planning problem, which is an approach for incorporating recycling considerations into product design. Zussman and Zhou (1999) modified the model using Petri nets and suggested an operational strategy to cope with the uncertainty of certain disassembly operation. Unlike the complete disassembly, Lambert (1997) presented a dynamic programming algorithm for determining the optimal disassembly sequence for selective disassembly of complex products with the objective of maximizing the revenue. Here, the selective disassembly implies incomplete disassembly sequences, i.e. an undefined final state of disassembly. For this problem, Lambert (1999a, b) suggested a linear programming formulation, which is a generalized version of Kanehara *et al*. (1993). In particular, Lambert (1999a) presented a sensitivity analysis to cope with uncertainties based on the suggested linear programming model. For other mathematical programming method, see Navin-Chandra (1994), which is based on a modification of the travelling salesman problem. For the more general problem that includes disassembly sequencing, Krikke *et al*. (1998) consider the problem of determining to what extent the products are disassembled and which recovery options are applied with the objective of maximizing net profit, and suggest a two-phased procedure using dynamic programming.

Kanai *et al.* (1999) suggested a disassembly model that considers the destructive disassembly, shredding of parts to fragments and sorting of materials. In their model, they used four kinds of graphs to represent products and processes. See Subramani and Dewhurst (1991), Lee and Kumara (1992), de Ron and Penev (1995), Gungor and Gupta (1998) and Moore *et al.* (1998) for other various disassembly models and algorithms. Also, various issues and literature survey on disassembly planning can be found in Gupta and McLean (1996) and O'Shea *et al.* (1998).

This paper focuses on the disassembly sequencing problem with EOL options, which is the problem of representing products with AND/OR graphs and determining disassembly sequences with the objective of maximizing the total profit. Here, the profit is defined as the revenues from disassembled parts minus disassembly cost. To solve the problem, one first suggests an algorithm to generate the product recovery graph semi-automatically for a given product liaison graph. Unlike the previous research, our starting point is the liaison diagram, not the (disassembly) AND/OR graph. That is, using the algorithm suggested in this paper, the AND/OR graph is derived semi-automatically from the corresponding liaison diagram and is not considered as assumed in previous research. Note that, to the best of authors' knowledge, the AND/OR graph is assumed to be given in most of the published literature. Then, based on the generated product recovery plan approach, a method capable of evaluating a product's EOL is presented. Our method does not require the pre-optimization of the EOL options of the subassemblies, since the concept of *hyperedge* was utilized. Here, each hyperedge allows to treat the EOL options in the same manner as a disassembly operation, i.e. each node in the product recovery graph is connected to all possible EOL options and not just a preselected optimal option for the connected node. The advantage of this approach is that the changes in EOL can be easily considered without changing the recovery graph. Furthermore, to cope with the uncertainty of the product, an algorithm was developed for questioning the recovery graph to analyse the value window of a target edge.

The presented algorithms have been implemented using Mathematica (Wolfram 1996), which can be the kernel of a system for product EOL options modelling and evaluation. The overall software system architecture is displayed on figure 1. As one can see, the kernel of the method can be divided into the following two main parts.

- Generate the disassembly AND/OR graph and extend it to the recovery graph.
- Find the optimal disassembly plan by searching feasible disassembly plans in the recovery graph under various optimization criteria.

The generation of the disassembly AND/OR graph is based on the liaison graph and precedence relations. The approach is similar to the method developed by Homem de Mello and Sanderson (1990), but the complementary set method was used instead of the cut-set method to generate candidate sequences. In the cut set method, all connected subgraphs with cardinality smaller than or equal to half of the total number of nodes were searched, and then a cut set was found that could divide the whole graph into two parts. However, in the complementary set method, complementary subassemblies were determined to generate the disassembly AND/OR graph. A detailed explanation is given in Section 2.2. For another approach to generate precedence graph for product structure analysis, see Danley *et al.* (1999).

The precedence relations—both geometrical and technological—are considered to be given and are used to prune the infeasible sequences. This graph is then

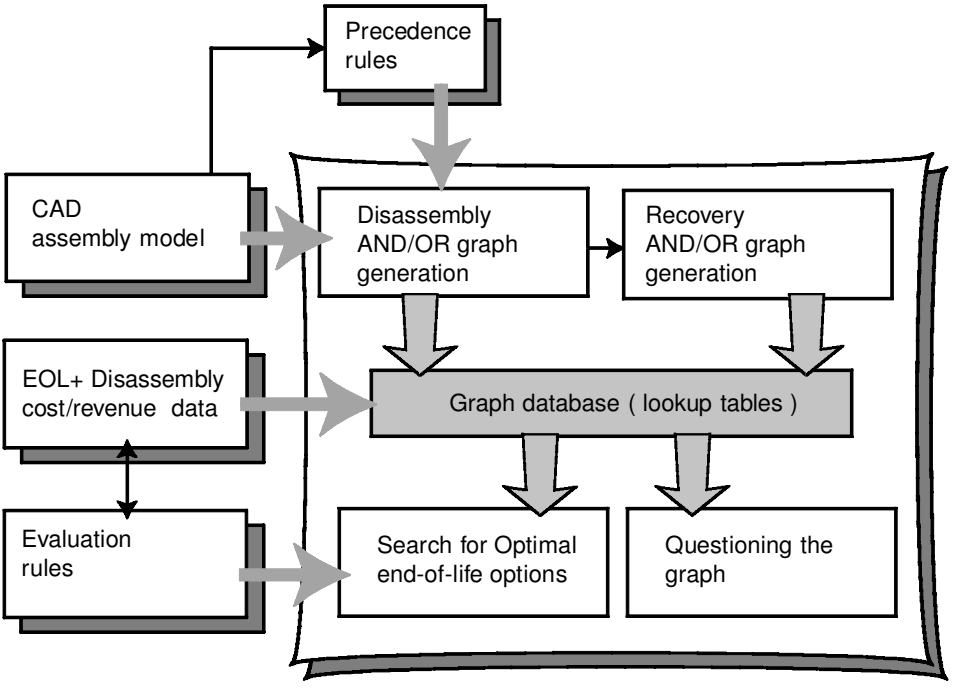

Figure 1. Software system architecture.

extended into a recovery AND/OR graph by adding final states to it. A list of weight label data sets is attached to every hyperedge of the graph during the generation. The numerical values of these weights can be changed during the search, where the weight values attached to the hyperedges represent either a disassembly cost or an EOL value. Each hyperedge value is either revenue (positive) or cost (negative).

Here, a search algorithm is used to obtain different optimal EOL plans for a product. The search algorithm utilizes the attached numerical cost/revenue values of the weight labels, which should be given by the user. This process is described schematically by the arrow connecting the 'EOL + Disassembly cost data' box with the 'Graph database (lookup tables)' box in figure 1. Since the number of attached numerical cost/revenue data sets was not limited, it is possible to search optimal EOL options from different point of views (shortest time, lowest impact on the environment, higher benefit, etc.). The optimization can be based on different criteria by simply choosing the corresponding data set of the parametric weight labels. It is also possible that the optimization be based on the weighted sum of these data sets. The utilized evaluation rule has to be provided by the user together with the corresponding cost/revenue data set, which can be obtained from an expert system. Besides the optimal plan search, an algorithm was designed to answer certain questions concerning the optimal EOL options of a target subassembly.

In Section 2, the generation of the recovery AND/OR graph and the associated data structure are discussed, as is the topological search algorithm utilized for finding the optimal EOL plan of a product is given in Section 3. Section 4 presents the algorithm of target edge analysis (questioning the graph). Section 5 concludes with recommendations for future research.

2. Recovery plan generation

2.1. *Definitions and assumptions*

The automatic or semi-automatic generation of the disassembly AND/OR graph is based on the enumeration of all possible subassemblies of a product. In general, it is difficult to take into consideration all destructive disassembly operations. Only those such as disconnecting a soldered cable can be easily included similar to non-destructive operations. Other destructive operations are usually case specific and their number is limited. Therefore, they can be added manually to the generated graphs. For example, some liquid contaminant can be removed by drilling its container. Here, it is assumed that a destructive disassembly operation is utilized if it reduces the EOL processing costs of the product. Hence, although one cannot evaluate all of the possible destructive disassembly operations, the optimization still can provide a good estimation of the upper limit for the EOL processing costs of the product.

Since the goal is to get a first estimate, i.e. the decomposition of disassembly operations into detailed disassembly steps is not considered, one takes into account only the geometric—or blocking—precedences used in the enumeration of the non-destructive disassembly sequence generation. The technological details of the operations are not considered explicitly but are included implicitly in the cost/revenue values, which are attached to the operations.

The starting point of the generation of the disassembly AND/OR graph is the product structure. In general, a product can be considered as an assembly of components. Hence, a product can be modelled as a non-oriented graph called connection graph, where each node represents component and each edge corresponds to physical connection between the components. A disassembly operation is considered as an edge cut of this non-oriented graph.

The automatic construction of the disassembly AND/OR graph is based on two assumptions: (1) a disassembly operation results in two separate connected subassemblies; and (2) the set of feasible disassembly operations is restricted by the precedence rules. As a consequence of the first assumption, only one piece, which can be a subassembly or a component, can be separated with one operation. This assumption does not conflict with the more realistic case, where disassembly operations can result in more than two connected subassemblies because this can be modelled as a sequence of two disassembly operations. For example, a screw may hold three or more components together. The cost and revenue values attached to such operations naturally have to be distributed to the resulting operations. The intuition behind the second assumption is that the non-destructive disassembly operations have to obey the geometric blocking constraints. These precedence constrains can be deduced from the detailed knowledge of the geometry of the product. The automatic extraction of the precedence rules is not considered in this article, and is considered as a part of the input.

2.2. *Recovery AND/OR graph*

Explained here is the algorithm for generating the recovery AND/OR graph that iteratively generates all the possible subassemblies of a given product. Here, the subassemblies are generated on non-destructive disassembly operations. A detailed description of the algorithm is given in the pseudo code of figure 2.

The kernel of the iteration is the construction of the feasible subassembly pairs based on the input product. A feasible subassembly pair is created by an operation

```
program Disassembly AND/OR Graph Generation

    product          ← connection graph of the product
    precedence       ← list of geometric precedence rules
    subassembly      ← (product, precedence)
    quelist          ← subassembly
    graph-database   ← Initialization with subassembly

    while quelist ≠ 0 loop
        subassembly      ← first(quelist); remove subassembly from quelist
        neighbour-list   ← GetNeighbourSubassembly(subassembly)
        feasible-list    ← GetFeasibleSubassembly(neighbour-list, subassembly)
        new-precedence   ← UpdatePrecedence(feasible-list, subassembly)
        sub-subassembly-list ← CreateProductList(feasible-list, new-
        precedence)
        append sub-subassembly-list to graph-database
        append sub-subassembly-list to quelist
    endloop

end
```

Figure 2. Algorithm for the recovery AND/OR graph generation.

that does not have any precedence edge in either of the subassembly pairs. The generated subassemblies are then recursively treated by the kernel until no further pairs can be generated. The kernel is decomposed into the following two basic parts.

- Generating the neighbouring subassemblies, i.e. all possible connected subassembly pairs that complement each other, and hence their union is equal to the input subassembly. The set of neighbouring subassemblies is calculated by the GetNeighbourSubassembly(*subassembly*) function in figure 2.
- Filtering out the subassembly-pairs that do not satisfy the precedence rules by using the GetFeasibleSubassembly(*neighbour-list, subassembly*) function in figure 2.

The GetNeighbourSubassembly function in figure 2 uses the complementary set method. That is, one first enumerates all connected subassemblies of the input product and these are generated level by level, i.e. the *i*th level subassemblies consist of all the sets of *i*th connected parts. Here, a subassembly with level *i* has *i* connected parts. Then, this set is divided into a *lower half set* and an *upper half set*, depending on whether the cardinality of components in the subassemblies is not greater or greater than half the cardinality of the components in the product. For every subassembly in the lower half set, its complementary set was calculated. The question is then posed as to whether each complementary set is a member of the upper half-set. If the answer is positive, then the subassembly and its complementary are moved into

the neighbouring subassembly set. After the neighbouring subassemblies are defined, those pairs that violate the precedence rules have to be filtered out (GetFeasibleSubassembly in figure 2). The precedence rules define precedence relations between disassembly operations, i.e. a disassembly operation can be used only if certain connections have already been disassembled. The resulting set of subassemblies is called feasible subassemblies. The final step in the kernel is the updating of the precedence rules. This is done by the UpdatePrecedence function, which removes the already disassembled connections from the precedence list. See Section 5 for an example of the recovery AND/OR graph generation.

The feasible subassembly pairs are recorded into the graph-database. This graph-database represents an oriented AND/OR graph with the following properties: the nodes of the graph are the feasible subassemblies and the edges are the disassembly operations used to divide a parent-assembly into two child subassemblies. One disassembly operation is represented by two edges emanating from one parent-assembly pointing into two subassemblies. Therefore, this pair of edges is called a hyperedge. During the generation of the disassembly AND/OR graph, a list of weight labels (or costs) is attached to every hyperedge, which will be used in the optimal plan search algorithm. After the generation of the disassembly AND/OR graph, the user can examine the result and manually add some nodes and hyperedges, which correspond to some specific destructive disassembly operations.

Finally, by adding terminal nodes to the disassembly AND/OR graph, it is converted into the recovery AND/OR graph. A terminal node represents an EOL option of a subassembly. The conversion of the disassembly AND/OR graph into a recovery AND/OR graph is done by adding hyperedges from every node of the original graph to the terminal nodes. The resulting graph is an acyclic oriented hypergraph that has to be searched for the shortest hyperpath that corresponds to the optimal EOL of the product.

3. Finding the optimal recovery plan

The problem of finding the optimal recovery plan in the disassembly AND/OR graph is addressed, which corresponds to the shortest path in the acyclic directed hypergraph. A directed hypergraph consists of a set of nodes and hyperedges between the nodes. A hyperedge is an ordered pair of two sets of nodes, which are called the *tail* and *head* of the edge, respectively, here. For example, the [{(Clip, Pentop), Clip}, {(Clip, Pentop), Pentop}] edges in figure 6 represent a hyperedge defined by the (Clip, Pentop) *tail* node and (Clip), (Pentop) head nodes. Note that an ordinary directed graph is a special hypergraph having one single node in both tail and head of each edge.

The application suggests the following restrictions: one only considers hypergraphs with edges whose tail consists of one single node. Furthermore, there is a *root* node that is not the head of any hyperedge. One does not consider graphs in which there is a closed chain of hyperedges, i.e. cycle. The *leaves* of a hypergraph will be frequently referred to, which are those nodes that are not the tails of any hyperedges.

After weighting the hyperedges of the graph, it is the wish to find the shortest hyperpath from the *root* to the leaves of the graph. In our domain, such a path represents a *recovery plan*. By summing up the weights of the hyperedges of a path, one can obtain the profit (revenue minus cost) for choosing that path provided that weights are assigned to hyperedges appropriately.

Our method is basically an application of the well-known topological sort algorithm to the hypergraph at hand Aho *et al*. (1987). However, it is modified to find the shortest hyperpath from the root to the leaves. First, the following trivial observation is made: for each node n in the graph described above, the length of the shortest path from n to the leaves depends only on the weights of hyperedges along a path from n to the leaves. Consequently, one can compute the shortest path from the leaves to each node of the graph *backwards*, so that eventually the shortest path to the root is also obtained. This is the well-known *Bellman's principle* for dynamic programming. Since the length of a hyperpath does not depend on which direction it is traversed, what one gets is really the shortest hyperpath from the root to the leaves. The advantage of the backwards calculation of the shortest hyperpath is that the hyperedges are only visited once since the computation of the node profits and of the optimal hyperplan proceed simultaneously.

Figure 3 describes the algorithm with a pseudo code. The algorithm maintains an *agenda* of nodes that initially consists of the leaves of the hypergraph. A node is on the *agenda* if and only if all of its successors have been processed. Here, a successor of a node is defined as the heads of the hyperedges emanating from the node. The array *OutEdges*(n) indicates the number of edges emanating from node n that have been processed; when it is zero for a node m, then m is placed on the agenda. For a particular node on the hypergraph, its value is calculated as the maximum of the sum

```
program Shortest Hyperpath

    agenda ← leaves of the hypergraph
    forall n ∈ nodes of the hypergraph loop
        OutEdges(n) ← number of edges whose tail is n
        value(n) ← − ∞
    forall n ∈ leaves of the hypergraph loop
        value(n) ← 0
    while agenda ≠ 0 loop
        n ← first(agenda); remove n from agenda
        forall e ∈ hyperedges whose tail is n loop
            value(e) ← weight(e) + Σ_{k∈head(e)} value(k)
            if value(n) <value(e) then
                value(n) ← value(e)
                next(n) ← e
        forall e ∈ hyperedges whose head contains n loop
            m ← tail(e)
            OutEdges(m) ← OutEdges(m) − 1
            if OutEdges(m) =0 then append m to agenda

end
```

Figure 3. Algorithm that for obtaining the optimal recovery plan.

of the weight of a hyperedge emanating from it plus its heads values. After running the algorithm on a hypergraph defined above, the *value* of each node is maximized and *next*(n) indicates the hyperedge emanating from node n that lies on the optimal plan. By following the *next* directions in the hypergraph from the *root*, the shortest hyperpath to the leaves can easily be read out.

4. Sensitivity analysis

The problem of finding the lowest value of a target edge is dealt with here provided that the optimal plan remains unchanged. That is, by how much one can lower the value (by lowering the value of a specified edge), one gets from the calculated optimal plan without changing the optimality of the current plan. One is essentially looking for the value distance between the current optimal plan and the next suboptimal plan. From the practical viewpoint, this information can be useful to estimate the benefit margins from the sale of a used component or subassembly, while maintaining the optimal recovery plan. To this end, it is assumed that one has a hypergraph and its shortest hyperpath from the root to the leaves. Note that this hyperpath can be obtained from the algorithm shown in figure 3.

Suppose that the optimal path joins the target component n to an end-of-life option k. That is, there is a hyperedge (target edge) in the hypergraph whose tail is n, i.e. the target component, and whose head is k and this hyperedge is a member of the shortest hyperpath from the *root* to the leaves found by the *shortest hyperpath algorithm*. Here, the target component is always the tail node of the target edge. Certainly, this hyperedge has a weight that is the value of choosing end-of-life option k for component n. One wants to find out whether it is possible to decrease this value without changing the optimal path and if the answer is affirmative, to what extent?

To answer the question posed above, the hypergraph and edge respectively and node values determined by our algorithm are used (figure 4). If n is the target component, then one examine the nodes starting with n up to the *root* along the shortest path. For each node i, the difference of the actual value and the second best value, if any, are determined and represented by a hyperedge emanating from the node but not selected in the shortest path. Certainly, if one decreases the weight of the hyperedge from n to the selected EOL option k by at most the minimum of these differences *target_diff*, then the optimal path remains the same. The reason is that all values along this path (from n up to the root) will decrease by this amount and by the definition of the minimum difference, the new values of the influenced nodes can attain only the second best value, but not less. The algorithm for calculating this value window is shown in figure 4.

However, the sketched method only gives the most cautious estimate. It is also possible in certain hypergraph structures that further decrease of the value still does not change the shortest hyperpath from the *root* to the leaves. This can happen when the propagation towards the root of the weight reduction of the target hyperedge affects other hyperedges beyond those on the optimal path. This case can be handled by iterating the above computation, using a new target edge weight reduced by the amount of the value window. The iteration stops when the optimal plan changes.

5. An example

To demonstrate the application of the developed algorithms, a simple ballpoint pen example has been chosen, which is the same as Lambert (1997). The assembly of the pen consists of 10 parts and 11 edges between them. The parts and the connec-

```
program Target Edge Analysis
repeat
    optplan ← shortest hyperpath in the recovery graph
    tedge ← target edge
    while tedge ≠ ∅ loop
        tedge_tail_node ← tail node of tedge
        value(tedge) ← weight(tedge) + Σ_{k∈head(tedge)} value(k)
        second_max ← − ∞
            forall e ∈ hyperedges whose tail is tedge_tail_node loop
                value(e) ← weight(e) + Σ_{k∈head(e)} value(k)
                if e ≠ tedge ∧ value(e) >second_max then
                second_max ← value(e)
            end loop
        append (value(tedge) − second_max) to diff_list
        tedge ← predecessor hyperedge whose head is tedge_tail_node in the
        optimal plan
    end loop
    target_diff ← min(diff_list)
    weight(tedge) ← weight(tedge) − target_diff
until(optimal plan changes)
end
```

Figure 4. Algorithm for obtaining the target hyperedge value window.

tions are shown in figure 5. Here, a pair of connected parts defines the corresponding connection. Therefore, the precedence constraints are defined as an ordered list of connections, where every restricted connection is paired with its precedence connections. All the precedence connections have to be disassembled prior to the restricted connection. The utilized precedence constraints of the disassembly operations are displayed in table 1.

The recovery AND/OR graph generation described in the pseudo code of figure 2 starts with the initialization. To illustrate the algorithm, the generation of feasible subassembly pairs that can be obtained from the liaison diagram of the ballpoint pen are explained. To determine the *neighbour-list* of the ballpoint pen, 10 levels of the connected subassemblies are enumerated (table 2). The complementer node set for each subassembly in the lower half-set is calculated. Then for each complementer node set one searches for subassemblies having the same node set in the upper-half set. If such an element is found, the subassembly pair from the lower and upper-half set is a neighbour subassembly pair. The calculated neighbouring subassembly pairs are listed in table 3. To calculate the feasible subassembly pairs out of the neighbour subassemblies, one has to check whether the removed edges satisfy the precedence constraints. To generate the neighbouring subassembly pairs, certain connection (separation edges) of the original assembly have to be removed. Table 4 shows the

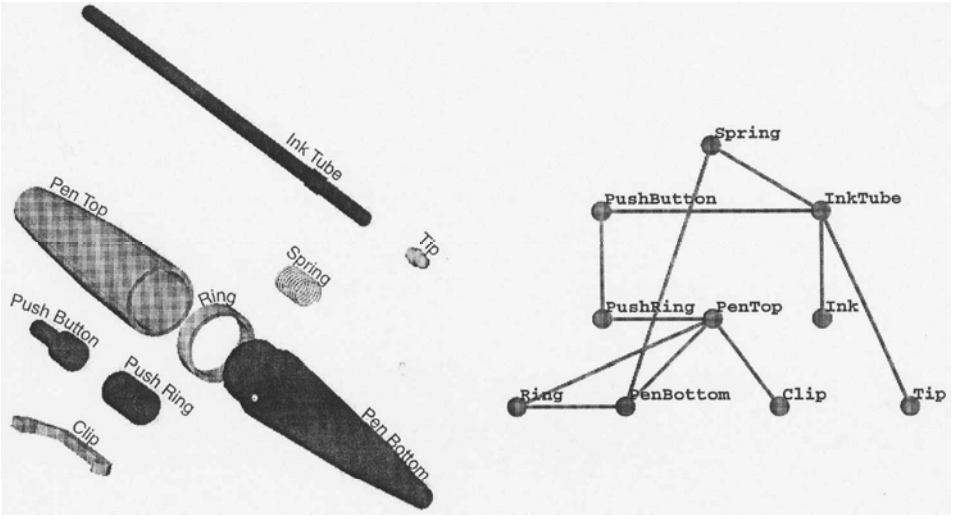

Figure 5. Parts and liaisons of the pen.

separation edges of every neighbouring subassembly pair in table 3. One can check whether the separation edges satisfies the precedence constraints (table 1). The feasible subassembly pairs are then those neighbouring subassembly pairs that satisfy the precedence constraints (table 5). The precedence rule can be updated after generating the feasible subassembly pairs by removing the separation edges from the precedence list. The remaining nodes of the disassembly AND/OR graph are calculated by iteratively applying the feasible subassembly pairs generation algorithm for every generated subassembly.

To model the EOL options of the product, two terminal nodes were added to the disassembly AND/OR graph. These are the LandFill and Reuse nodes. They are the leaf nodes for every path in the recovery graph because they are connected with every node of the disassembly graph. The generated recovery AND/OR graph is displayed in figure 6. To increase the legibility, one does not display the two terminal nodes (Landfill and Reuse), since they are connected to all nodes of the disassembly AND/OR graph.

Also, to find the optimal EOL plan of the product, revenue and cost are attached to the hyperedges and are obtained from Lambert (1997). If the values of the weight labels are positive, they are considered as revenue, while in case of negative values, they represent the cost of the operations. The assigned values are displayed in table

Connection	Precedence connections
{PushButton, PushRing}	{{penToP, PenBottom}, {PenTop, Ring}, {PushButton, InkTube}, {PushRing, PenTop}}
{PushRing, PenTop}	{{PenTop, PenBottom}, {PenTop, Ring}, {PushButton, InkTube}}
{RingPen, Bottom}	{{PenTop, PenBottom}, {PenTop, Ring}, {PushButton, InTube}}
{PenBottom, Spring}	{{PenTop, PenBottom}, {PenTop, Ring}, {PushButton, InkTube}}
{Spring, InkTube}	{{PenTop, PenBottom}, {PenTop, Ring}, {Pushbutton, InkTube}}
({InkTube, Tip}	{{PenTop, PenBottom}, {PenTop, Ring}, {PushButton, InkTube}, {Spring, InkTube}}
{InkTube, Ink}	{{PenTop, PenBottom}, {PenTop, Ring}, {PushButton, InkTube}, {Spring, InkTube}, {InkTube, Tip}}

Table 1. Precedence constraints of the disassembly operations.

			Lower half set				
NodeLevel [1]	PushButton	PushRing	PenTop	Clip	Ink	InkTube	...
NodeLevel [2]	Clip	Ink	InkTube	InkTube	InkTube	PenBottom	...
	PenTop	InkTube	PushButton	Spring	Tip	PenTop	
NodeLevel [3]	Clip	Clip	Clip	Ink	Ink	Ink	
	PenBottom	PenTop	PenTop	InkTube	InkTube	InkTube	...
	PenTop	PushRing	Ring	PushButton	Spring	Tip	
NodeLevel [4]	Clip	Clip	Clip	Clip	Clip	Ink	
	PenBottom	PenBottom	PenBottom	PenTop	PenTop	InkTube	
	PenTop	PenTop	PenTop	PushButton	PushRing	PenBottom	...
	PushRing	Ring	Spring	PushRing	Ring	Spring	
NodeLevel [5]	Clip	Clip	Clip	Clip	Clip	Clip	
	InkTube	InkTube	PenBottom	PenBottom	PenBottom	PenBottom	
	PenBottom	PenTop	PenTop	PenTop	PenTop	PenTop	...
	PenTop	PushButton	PushButton	PushRing	PushRing	Ring	
	Spring	PushRing	PushRing	Ring	Spring	Spring	
			Upper half set				
NodeLevel [6]	Clip	Clip	Clip	Clip	Clip	Clip	
	Ink	Ink	InkTube	InkTube	InkTube	InkTube	
	InkTube	InkTube	PenBottom	PenBottom	PenBottom	PenBottom	
	PenBottom	PenTop	PenTop	PenTop	PenTop	PenTop	...
	Pentop	PushButton	PushButton	PushButton	PushRing	Ring	
	Spring	PushRing	PushRing	Spring	Spring	Spring	
NodeLevel [7] Clip	Clip	Clip	Clip	Clip	Clip		
	Ink	Ink	Ink	Ink	Ink	Ink	
	InkTube	InkTube	InkTube	InkTube	InkTube	InkTube	
	PenBottom	PenBottom	PenBottom	PenBottom	PenBottom	PenTop	...
	PenTop	PenTop	PenTop	PenTop	PenTop	PushButton	
	PushButton	PushButton	PushRing	Ring	Spring	PushRing	
	PushRing	Spring	Spring	Spring	Tip	Ring	
Nodelevel [8]	Clip	Clip	Clip	Clip	Clip	Clip	
	Ink	Ink	Ink	Ink	Ink	Ink	
	InkTube	InkTube	InkTube	InkTube	InkTube	InkTube	
	PenBottom	PenBottom	PenBottom	PenBottom	PenBottom	PenBottom	
	PenTop	PenTop	PenTop	PenTop	PenTop	PenTop	...
	PushButton	PushButton	PushButton	PushButton	PushButton	PushRing	
	PushRing	PushRing	PushRing	Ring	Spring	Ring	
	Ring	Spring	Tip	Spring	Tip	Spring	
NodeLevel [9]	Clip	Clip	Clip	Clip	Clip	Clip	
	Ink	Ink	Ink	Ink	Ink	Ink	
	InkTube	InkTube	InkTube	InkTube	InkTube	InkTube	
	PenBottom	PenBottom	PenBottom	PenBottom	PenBottom	PenTop	
	PenTop	PenTop	PenTop	PenTop	PenTop	PushButton	...
	PushButton	PushButton	PushButton	PushButton	PushRing	PushRing	
	PushRing	PushRing	PushRing	Ring	Ring	Ring	
	Ring	Ring	Spring	Spring	Spring	Spring	
	Spring	Tip	Tip	Tip	Tip	Tip	
NodeLevel [10]	Clip						
	Ink						
	InkTube						
	PenBottom						
	PushButton						
	PenTop						
	PushRing						
	Ring						
	Spring						
	Tip						

Table 2. Precedence constraints of the disassembly operations.

No.	Complementer connected subassembly pairs
1	{{**Clip**}, {Ink, InkTube, PenBottom, PenTop, PushButton, PushRing, Ring, Spring, Tip}}
2	{{**Ink**}, {Clip, InkTube, PenBottom, PenTop, PushButton, PushRing, Ring, Spring, Tip}}
3	{{**PenBottom**}, {Clip, Ink, InkTube, PenTop, PushButton, PushRing, Ring, Spring, Tip}}
4	{{**PushButton**}, {Clip, Ink, InkTube, PenBottom, PenTop, PushRing, Ring, Spring, Tip}}
5	{{**PushRing**}, {Clip, Ink, InkTube, PenBottom, PenTop, PushButton, Ring, Spring, Tip}}
6	{{**Ring**}, {Clip, Ink, InkTube, PenBottom, PenTop, PushButton, PushRing, Spring, Tip}}
7	{{**Spring**}, Clip, Ink, InkTube, PenBottom, PenTop, PushButton, PushRing, Ring, Tip}}
8	{{**Tip**}, {Clip, Ink, InkTube, PenBottom, PenTop, PushButton, PushRing, Ring, Spring}}
9	{{**Clip**, **PenTop**}, {Ink, InkTube, PenBottom, PushButton, PushRing, Ring, Spring, Tip}}
10	{{**Penbottom**, **Ring**}, {Clip, Ink, InkTube, PenTop, PushButton, PushRing, Spring, Tip}}
11	{{**Penbottom**, **Spring**}, {Clip, Ink, InkTube, PenTop, PushButton, PushRing, Ring, Tip}}
12	{{**Penbottom**, **PushRing**}, {Clip, Ink, InkTube, PenBottom, PenTop, Ring, Spring, Tip}}
13	{{**Clip**, **PenTop**, **PushRing**}, {Ink, InkTube, PenBottom, PushButton, Ring, Spring, Tip}}
14	{{**Clip**, **PenTop**, **Ring**}, {Ink, InkTube, PenBottom, PushButton, PushRing, Spring, Tip}}
15	{{**Ink**, **InkTube**, **Tip**}, {Clip, PenBottom, PenTop, PushButton, PushRing, Ring, Spring}}
16	{{**Penbottom**, **Ring**, **Spring**}, {Clip, Ink, InkTube, PenTop, PushButton, PushRing, Tip}}.
17	{{**Clip**, **Penbottom**, **PenTop**, **Ring**}, {Ink, InkTube, PushButton, PushRing, Spring, Tip}}
18	{{**Clip**, **PenTop**, **PushButton**, **PushRing**}, {Ink, InkTube, PenBottom, Ring, Spring, Tip}}
19	{{**Clip**, **PenTop**, **PushRing**, **Ring**}, {Ink, InkTube, PenBottom, PushButton, Spring, Tip}}
20	{{**Ink**, **InkTube**, **PushButton**, **Tip**}, {Clip, PenBottom, PenTop, PushRing, Ring, Spring}}
21	{{**Ink**, **InkTube**, **Spring**, **Tip**}, {Clip, PenBottom, PenTop, PushButton, PushRing, Ring}}
22	{{**Clip**, **Penbottom**, **Pentop PushRing**, **Ring**}, {Ink, InkTube, PushButton, Spring, Tip}}
23	{{**Clip**, **Penbottom**, **PenTop**, **Ring**, **Spring**}, {Ink, InkTube, PushButton, PushRing, Tip}}
24	{{**Clip**, **PenTop**, **PushButton**, **PushRing**, **Ring**}, {Ink, InkTube, PenBottom, Spring, Tip}}

Table 3. Complementer conected subassembly pairs.

No.	Separation edges		
1	{Clip, PenTop}		
2	{Ink, InkTube}		
3	{PenBottom, PenTop}	[PenBottom, Ring]	{PenBottom, Spring}
4	{InkTube, PushButton}	{PushButton, PushRing}	
5	{PenTop, PushRing}	{PushButton, PushRing}	
6	{PenBottom, Ring}	{PenTop, Ring}	
7	{InkTube, Spring}	{PenBottom, Spring}	
8	{InkTube, Tip}		
9	{PenBottom, PenTop}	{PenTop, PushRing}	{PenTop, Ring}
10	{PenBottom, PenTop}	{PenBottom, Spring}	{PenTop, Ring}
11	{InkTube, Spring}	{PenBottom, PenTop}	{PenBottom, Ring}
12	{InkTube, PushButton}	{PenTop, PushRing}	
13	{PenBottom, PenTop}	{PenTop, Ring}	{PushButton, PushRing}
14	{PenBottom, PenTop}	{PenBottom, Ring}	{PenTop, PushRing}
15	{InkTube, PushButton}	{InkTube, Spring}	
16	{InkTube, Spring}	{PenBottom, PenTop}	{PenTop, Ring}
17	{PenBottom, Spring}	{PenTop, PushRing}	
18	{InkTube, PushButton}	[PenBottom, PenTop}	{PenTop, Ring}
19	{PenBottom, PenTop}	{PenBottom, Ring}	{PushButton, PushRing}
20	{InkTube, Spring}	{PushButton, PushRing}	
21	{InkTube, PushButton}	{PenBottom, Spring}	
22	{PenBottom, Spring}	{PushButton, PushRing}	
23	{InkTube, Spring}	[PenTop, PushRing}	
24	{InkTube, PushButton}	{PenBottom, PenTop}	{PenBottom, Ring}

Table 4. Separation edges of neighbouring subassemblies.

No.	Feasible subassembly pairs
1	{{Clip}, {Ink, InkTube, PenBottom, PenTop, PushButton, PushRing, Ring, Spring, Tip}}
2	{{Clip, PenTop, PushButton, PushRing}, {Ink, InkTube, PenBottom, Ring, Spring, Tip}}

Table 5. Feasible subassembly pairs.

6. Computing the shortest hyperpath in the recovery AND/OR graph results in the cost optimal hyperplan displayed in figure 7. The total revenue of this plan is 1.3502$.

We have selected the $(\{PenTop, PushButton, PushRing\}, \{Reuse\})$ target edge for the sensitivity analysis. The currently assigned value of this edge is $weight[6, 25] = 0.099\$$. The calculated window value of this target edge is 0.0002$. By decreasing the value of the target edge with the window value, namely, setting $weight[6, 25] = 0.0988\$$, the optimal plan remains the same as it displayed in figure 7, but the total revenue of this plan is decreased to 1.35$. Any further decreasing of the value of the target edge, for example by assigning $weight[6, 25] = 0.097999$, results in the structural changing of the optimal plan, as it is shown in figure 8. Therefore, the value of the target edge [6,25] can be decreased by 0.0002$ without changing the structure of the optimal plan.

6. Concluding remarks

In this paper, the modelling and evaluating of product end-of-life options has been considered. For the entire problem, three algorithms are developed: (1) the semi-automatic generation of the product recovery graph, given the product liaison graph; (2) the backwards calculation (so that the recovery graph hyperedges are only visited once) of the optimum hyperplan that maximizes the plan's revenue; and (3) a sensitivity analysis, which is called the recovery graph questioning algorithm, that finds the margin of allowed revenue reduction of a given target edge that maintains the same optimal plan. In this research, the first algorithm is based on the complementary-set method instead of the cut-set method employed previously in the literature with regard to the study of assembly sequences. For the question of disassembly sequence generation, the assumption is usually made in previous publications that the AND/OR disassembly graph is given while in this paper the AND/OR disassembly graph is derived from the product's liaison graph using the first algorithm. The second algorithm calculates the optimum plan of the recovery graph that optimizes some given criterion such as maximum revenue. The advantage of the algorithm presented (which is a modification of the well-known topological sort algorithm) is that the maximum revenue is computed backwards so that the hyperedges are only visited once. The third algorithm allows one to find the answer to the following question on the recovery graph: by how much one can lower the value (by lowering the value of a specified target edge) to get from the calculated optimal plan without changing the current plan optimality? This is useful to estimate the benefit margins from the sale of a used component or subassembly, while maintaining the optimal recovery plan.

Future work should include the modelling of more EOL options structures, which consider technological alternatives. Furthermore the extension of this work to the associated scheduling problem should also be part of future research. One has assumed only simple blocking type precedence relation in the disassembly graph; to

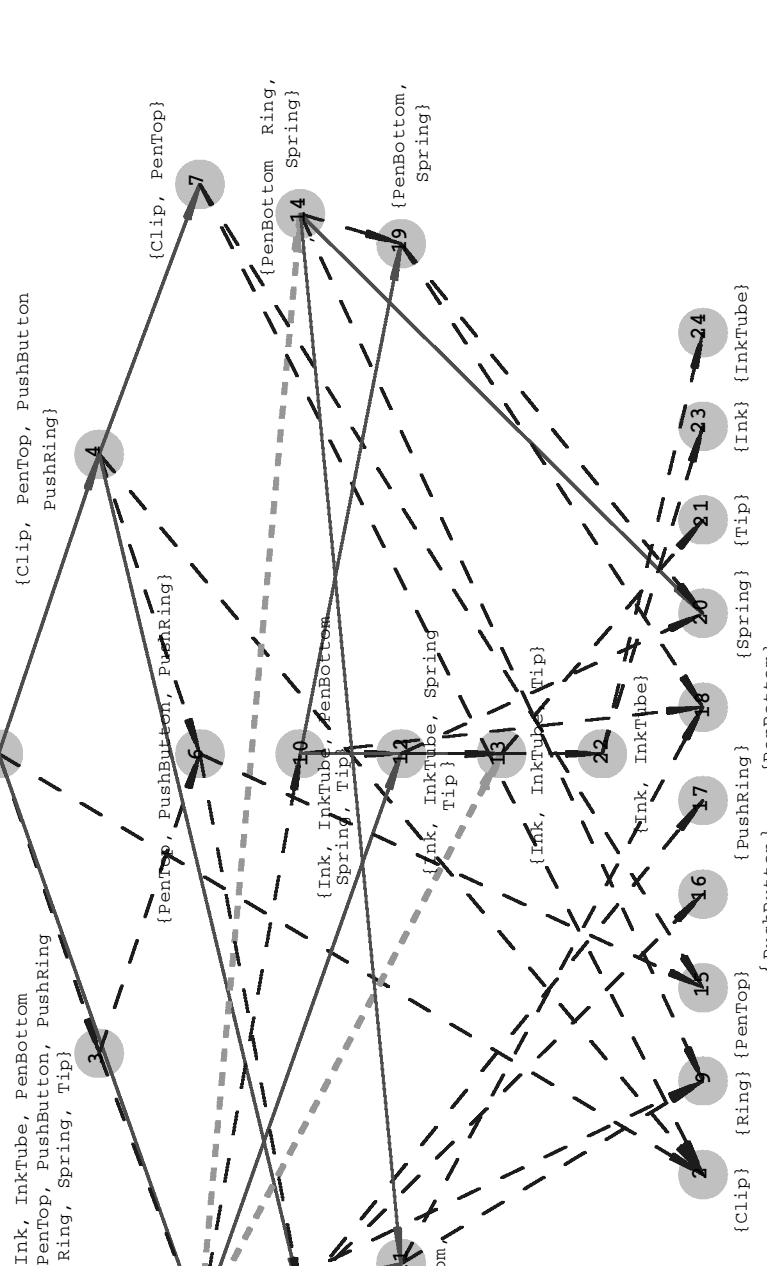

Figure 6. Recovery AND/OR graph of the pen.

Value of part ($/item)	Value of subassembly ($/item)	Disassembly cost ($/item)	
weight[2, 25] = 1.59	weight13, 25] = −1.7809	weight[1, 2, 3}] = −0.15	weight[10, {19, 13}] = −0.45
weight[9, 25] = 0.135	weight[4, 25] = 0	weight[1, {4, 5}] = −0.1	weight[11, {18.9}] = −0.4
weight[15, 25] = 0.149	weight[5, 25] = −0.629	weight[3, {6, 5}] = −0.2	weight[12, {20, 13}] = −0.5
weight[16, 25] = 0.19	weight[6, 25] = 0.099	weight[4, {2, 6}] = −0.55	weight[13, {21, 22}] = −0.75
weight[17, 25] = 0.055	weight[7, 25] = 0.204	weight[4, {7, 8}] = −0.6	weight[14, {9, 19}] = −0.7
weight[18, 25] = 0.059	weight[8, 25] = 0.147	weight[5, {9, 10}1 = −0.25	weight[14, {20, 11}] = −0.65
weight[20, 25] = 0.29	weight[10, 25] = −0.04672	weight[5, {11, 12}1 = −0.3	weight[19, {18, 20}] = −0.45
weight[21, 25] = 0.95	weight[11, 25] = 0.0832	weight[5, {13, 14}] = −0.35	weight[22, {23, 24}] = −0.5
weight[23, 25] = 0	weight[12, 25] = −0.315	weight[6, {15, 8}] = −0.8	
weight[24, 25] = 0.095	weight[13, 25] = −0.152	weight[7, {2, 15}] = −0.55	
	weight[14, 25] = 0.1992	weight[8, {16, 17}] = −0.6	
	weight[19, 25] = 0.1632	weight[10, {18, 12}] = −0.4	
	weight[22, 25] = −0.038		

Table 6. Input revenue and cost data of the ball point pen.

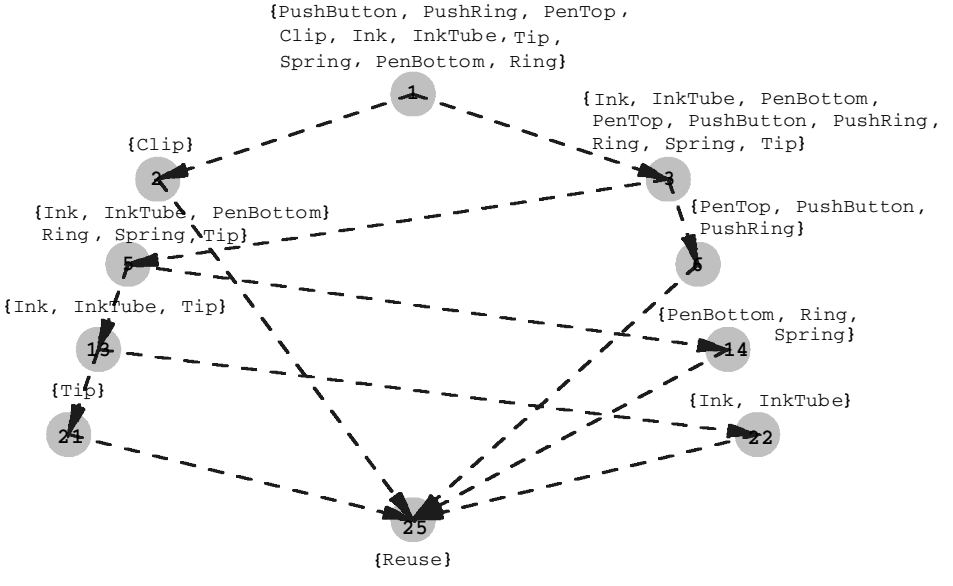

Figure 7. Optimal EOL hyperplan.

be able to treat more complex product future work should address more complex precedence relations.

Acknowledgements

The authors thank Dr Dong-Ho Lee for cooperation and valuable advice. They also gratefully acknowledge the partial support from FNRS 'Integrated dynamical modelling for process and production planning' project.

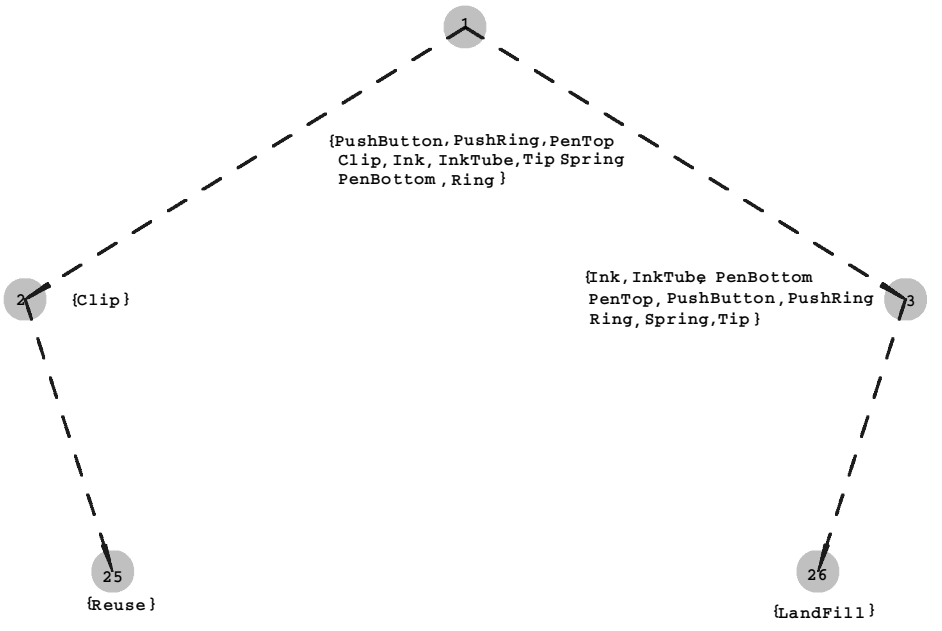

Figure 8. Optimal EOL hyperplan for modified target edge value.

References

Aho, V. A., Hopcroft, E. J., and Ullman, D. J., 1987, *Data Structures and Algorithms* (Addison-Wesley).

Boothroyd, G., and Dewhurst, P., 1996, *Design for Environment Software* (Wakefield: Boothroyd & Dewhurst).

Bourjault, A., 1984, Contribution une Approche Mèthodologique de l'Assemblage Automatisé: èlaboration Automatique des Sèquence Opèratoires. PhD thesis, Université de France'Compte.

Danley, J., Petit, F., Leroy, A., de Lit, P., and Rekiek, B., 1999, A pragmatic approach for precedence graph generation. *Proceedings of the IEEE International Symposium on Assembly and Task Planning*, 387–392.

De Ron, A. J., and Penev, K. D., 1995, Disassembly and recycling of electronic consumer products: an overview. *Technovation*, **15**, 363–374.#

Gungor, A., and Gupta, S. M., 1998, Disassembly sequence planning for products with defective parts in product recovery. *Computers and Industrial Engineering*, **35**, 161–164.

Gupta, S. M., and McLean, C. R., 1996, Disassembly of products. *Computers and Industrial Engineering*, **31**, 225–228.

Homem De Mello, L. S., and Sanderson, A. C., 1990, AND/OR graph representation of assembly plans. *IEEE Transactions on Robotics and Automation*, **6**, 188–199.

Homem De Mello, L. S., and Sanderson, A. C., 1991, A correct and compete algorithm for the generation of mechanical assembly sequences. *IEEE Transactions on Robotics and Automation*, **7**, 228–240.

Johnson, M. R., and Wang, M. H., 1995, Planning product disassembly for material recovery opportunities. *International Journal of Production Research*, **33**, 3119–3142.

Johnson, M. R., and Wang, M. H., 1998, Economical evaluation of disassembly operations for recycling, remanufacturing and reuse. *International Journal of Production Research*, **36**, 3227–3252.

Kanai, S., Sasaki, R., and Kishinami, T., 1999, Representation of product and processes for planning disassembly, shredding, and material sorting based on graphs. *Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning*, 123–128.

KANEHARA, T., SUZUKI, T., INABA, A., and OKUMAKUMA, S., 1993, On algebraic and graph structural properties of assembly Petri net—searching by linear programming. *Proceedings of the 1993 IEEE/RSJ International Symposium on Intelligent Robots and Systems*, 2286–2293.

KRIKKE, H. R., VAN HARTEN, A., and SCHUUR, P. C., 1998, On a medium term product recovery and disposal strategy for durable assembly products. *International Journal of Production Research*, **36**, 111–139.

KRIWET, A., ZUSSMAN, E., and SELIGER, G., 1995, Systematic integration of design-for-recycling into product design. *International Journal of Production Economics*, **38**, 15–22.

LAMBERT, A. J. D., 1997, Optimal disassembly of complex products. *International Journal of Production Research*, **35**, 2509–2523.

LAMBERT, A. J. D., 1999a, Optimal disassembly sequence generation for combined material recycling a part reuse. *Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning*, 146–151.

LAMBERT, A. J. D., 1999b, Linear programming in disassembly/clustering sequence generation. *Computers and Industrial Engineering*, **36**, 723–738.

LEE, Y.-Q., and KUMARA, S. R. T., 1992, Individual and group disassembly sequence generation through freedom and interface spaces. *Journal of Design and Manufacturing*, **2**, 143–154.

MOORE, K. E., GUNGOR, A., and GUPTA, S. M., 1998, A Petri net approach to disassembly process planning. *Computers and Industrial Engineering*, **35**, 165–168.

NAVIN-CHANDRA, D., 1994, The recovery problem in product design. *Journal of Engineering Design*, **5**, 65–86.

O'SHEA, B., GREWAL, S. S., and KAEBERNICK, H., 1998, State of the art literature survey on disassembly planning. *Concurrent Engineering: Research and Applications*, **6**, 345–357.

PENEV, K. D., and DE RON, A. J., 1996, Determination of a disassembly strategy, opportunities. *International Journal of Production Research*, **34**, 495–506.

PNUELI, Y., and ZUSSMAN, E., 1997, Evaluating the end-of-life value of a product and improving it by redesign. *International Journal of Production Research*, **35**, 921–942.

SANDERSON, A. C., HOMEM DE MELLO, L. S., and ZHANG, H., 1990, Assembly sequence planning. *AI Magazine*, **11**, 62–82.

SUBRAMANI, A. K., and DEWHURST, P., 1991, Automatic generation of disassembly sequence. *Annals of CIRP*, **40**, 115–118.

WOLFRAM, S., 1996, *The Mathematica Book* (Wolfram Media).

ZHANG, H. C., KUO, T. C., LU, H. T., and HUANG, S. H., 1997, Environmentally conscious design and manufacturing: a state-of-the-art survey. *Journal of Manufacturing Systems*, **16**, 352–371.

ZUSSMAN, E., KRIWET, A., and SELIGER, G., 1994, Disassembly-oriented assessment methodology to support design for recycling. *Annals of CIRP*, **43**, 9–14.

ZUSSMAN, E., and ZHOU, M. C., 1999, A methodology for modelling and adaptive planning of disassembly processes. *IEEE Transactions on Robotics and Automation*, **15**, 190–194.

ZUST, R., and WAGNER, R., 1992, Approach to the identification and quantification of environmental effects during product life. *Annals of CIRP*, **41**, 473–477.