Generating Optimal Disassembly Process Plans from AND/OR Relationships using a Hierarchical Genetic Algorithm

R. Edmunds, M. Kobayashi, M. Higashi
Engineering Design Laboratory, Research Center for Sustainable Mechanical Systems,
Toyota Technological Institute, 2-12-1 Hisakata, Tempaku-ku, Nagoya 468-8511, Japan
Email: edmunds@toyota-ti.ac.jp

Abstract
A Hierarchical Genetic Algorithm (HGA) is presented which, not only successfully finds the optimal disassembly sequence – under a set of given criteria – using information derived from AND/OR relationships, but also reduces the problem size. Whilst many authors agree that AND/OR graphs are the most complete representation of Disassembly Process Plans (DPPs), few have generated optimal sequences from AND/OR information due to the rapid increase of solution paths. For complex systems having a natural hierarchical structure often the optimal solution can be missed, HGA overcomes this and has been proven to be faster and more accurate than traditional Genetic Algorithms.

Keywords:
Disassembly, AND/OR Graph, Numerical Optimization, Hierarchical Structure

1 INTRODUCTION

The biggest damage to the environment is done when a product completes its useful life. However, many products are already in existence for which their End-Of-Life (EOL) use was never considered. Disassembly of a system into its component parts or subassemblies is vital to most EOL strategies. This paper uses a Hierarchical Genetic Algorithm (HGA) to find optimal disassembly paths from the space of all feasible Disassembly Process Plans (DDPs) described using AND/OR information.

Development of a disassembly theory is fundamental for practically all EOL policies devised for the disposal of products – in particular, the careful selection and removal of components for recycling and reuse.

With environmental ideologies ingrained in the public consciousness, and with large new deposits of natural resources becoming increasingly rare, there is extra pressure on industry to take back possession of products at the end of their useful life for reclamation of parts and materials, plus safe disposal of waste products. For the companies, this involves extra time, effort and money, and so the process must be done in the most economically and environmentally viable way [1].

Therefore, there are three elements that will be evaluated when assessing an EOL decision [2]: the costs and benefits of each recovery option; the present disposal cost and the possibility that disassembly is required to recover a valuable part.

If disassembly is needed, the arrangement of the components will, most likely, constrain the sequences such that removal of parts in perfect order (e.g. that the most expensive are removed first and the heaviest last) is unlikely [2-3].

As a consequence, it is necessary to represent the relationships between the components. The most common methods are via AND/OR graphs, which show the possible operations applicable to the current assembly, and precedence graphs, which show the geometrical relationships between the components. Both lead to a hierarchical structure.

For problems with an inherent hierarchical arrangement, traditional Genetic Algorithms (GAs) can miss the true optimal solution [4]. Even if the correct optimum is found, this can take increased computational time when compared to systems with vector arrangements. The Hierarchical Genetic Algorithm [4-5] was developed to overcome these issues and has been proven to work better than standard GAs for hierarchical systems [4–8].

Using the permissible operations upon the assembly to find the antecedents for each of its subassemblies, the configuration required for optimization with HGA is constructed. In doing this, for complete disassembly, the problem size is appreciably reduced. Extra constraints upon the removal order are easily added under this formulation.

The possibly of partial disassembly can also be introduced. This increases the dimensions of the hierarchy, but as it is mainly the width that is augmented, HGA still allows rapid and efficient resolution to obtain the optimal disassembly path.

The paper continues by giving a brief overview of disassembly and explains the difference between the two graphical representations. This is followed by an explanation of the HGA routine, which is subsequently used to model two examples from the literature and optimal disassembly paths are generated. Thus it is shown how disassembly problems represented via AND/OR graphs can be solved using HGA and highlights the benefits of doing this. Finally conclusions are drawn.

2 DISASSEMBLY

The addition of disassembly into the recovery process incurs extra costs. Specifically, disassembly contributes to the investment and labour overheads [9].

The cost involved in these operations are then constraints upon the methods employed and minimizing the cost of some or all of them can lead to finding the optimal choice, a preferred sequence that maximizes the reclaimed value.

Optimal disassembly sequence generation – in terms of minimal cost, maximum benefit and the degree of disassembly – is a non-trivial task. It is often done by analysing design characteristics of the assembly [2]:

- Geometrical relationships;
- Characteristics of operations such as tooling or accessibility and how much they overlap;
- Clustering of materials;
- Concurrent operations and the amount of material recovered.

To find the optimal sequence it is usually considered that all of the feasible paths – called Disassembly Process Plans (DPPs) (or Disassembly Sequence Plans (DSPs)) – must be generated and assessed, leading to a two-stage process [2,10]. Generating the total sequences often involves some sort of complex searching methodology and a significant amount of research has been put into extracting the DPPs from assembly diagrams and representing these plans in graphical form [11].

Most disassembly problems are variations of the same basic model structure, which is described as a list of possible disassembly operations or subassemblies [11]. From these feasible subassemblies and feasible actions, hierarchical disassembly graphs can be built by, either representing the geometric/relational properties (AND/OR graphs), or via the precedence knowledge alone (precedence graphs) [12]. In both cases this information can then be used to find the optimal sequence based on one or more criteria.

Although there is no general procedure for converting one type into the other [13], often assemblies can be described using both precedence and AND/OR information. The latter is the more flexible of the representations and there are a number of products that can be formulated using AND/OR graphs alone.

Precedence graphs are constructed by determining the parts directly obstructing the removal for each component and hence which other components must be extracted prior to their removal. By adhering to the structure – the existence of precedence relations stops every combinatorially possible permutation of parts resulting in a feasible subassembly [14] – the graphs hold all of the information on the possible solution paths.

Thus precedence graphs are conceptually simple and, as they consist of conditions that must be satisfied by the sequences, they are compact and have few nodes.

A major disadvantage with using precedence graphs to represent an assembly is that, although it is possible to concatenate ``OR'' sequences using dummy operations, for most products, no single graph can encompass every sequence [15].

Precedence relationships themselves, which represent individual operations or logical combinations of operations, **can** encompass all plans; however, this set of operations must be fixed – only the order can change – and can only be executed serially (parallel operations cannot be represented).

In fact, six types of disassembly relationship exist [16]:

- No precedents;
- No antecedents;
- AND relationships;
- OR relationships;
- AND relationships within an OR;
- OR relationships within an AND.

A representation methodology should be able to cope with these types, as well as have the ability to handle parallel disassembly operations.

AND/OR graphs map the tasks that can be performed on the assembly and thus are capable of depicting all technically feasible sequences. They are therefore explicit and it can be seen when operations can be executed in parallel [13,17].

The drawback of this completeness is that the growth of AND/OR graphs is exponential and representing all DPPs using AND/OR graphs is computationally expensive [18]. As a result it has, in general, proved impractical to use AND/OR data in order to find optimal disassembly paths – the search for all solutions leading to combinatory explosion.

These problems are reduced by organizing AND/OR information expressing the DDPs into the format utilized by HGA and then optimizing. This is especially true in the case of complete disassembly.

3 HIERARCHICAL GENETIC ALGORITHM

Many (complex) systems have a hierarchical structure. In conventional GAs and other optimization methods such hierarchical systems are transformed into a one-dimensional data array. However, these arrays are not suitable for expressing problems having hierarchical structures as lower level genotype variables depend upon upper level genotype variables.

Although there are several methods for dealing with hierarchical structures [5], for concurrent optimization of variables in different levels of the structural hierarchy, other techniques must carry out separate optimizations and the structure is not accurately represented. This is especially true as problems grow in size and finding the optimal solution becomes more complicated.

Such issues are avoided by using a genetic algorithm in which the hierarchical genotype coding exactly expresses the structure and detail of the hierarchical system. This uses the idea that when the upper level genotype variables change, the lower level genotype variables must also change. As the length of the genes may additionally vary, new crossover and mutation operators for treating the hierarchical genotype representations have also been required to be defined [5].

Crossover operations between individuals are conducted by: selecting another individual as the crossover partner and then exchanging the corresponding genes of the individuals, where to preserve consistency, all corresponding lower substructures are also swapped. Mutation operators are applied to the set of genes at the highest level of the hierarchical structural system, and then recursively applied to their child genes in the same manner as the crossover operator.

To represent a hierarchical structural system:

When a substructure has n lower substructures, each of which has a_i alternatives (i = 1, 2, …, n), this is denoted (a_1, a_2, …, a_n). When the substructure corresponds to the tth alternative for the sth substructure at the upper substructures level, the prefix symbols s, t- are added to the original notation (a_1, a_2, …, a_n). If there are upper level substructures, these procedures are repeated, and further prefix symbols are added to the notation.

Hence, using this method, each of the substructures is independently described. The positions of all substructures in the hierarchical structure are denoted by nodes, where a node located higher than the node being considered is called a "parent" node and one located in a lower position a "child" node.

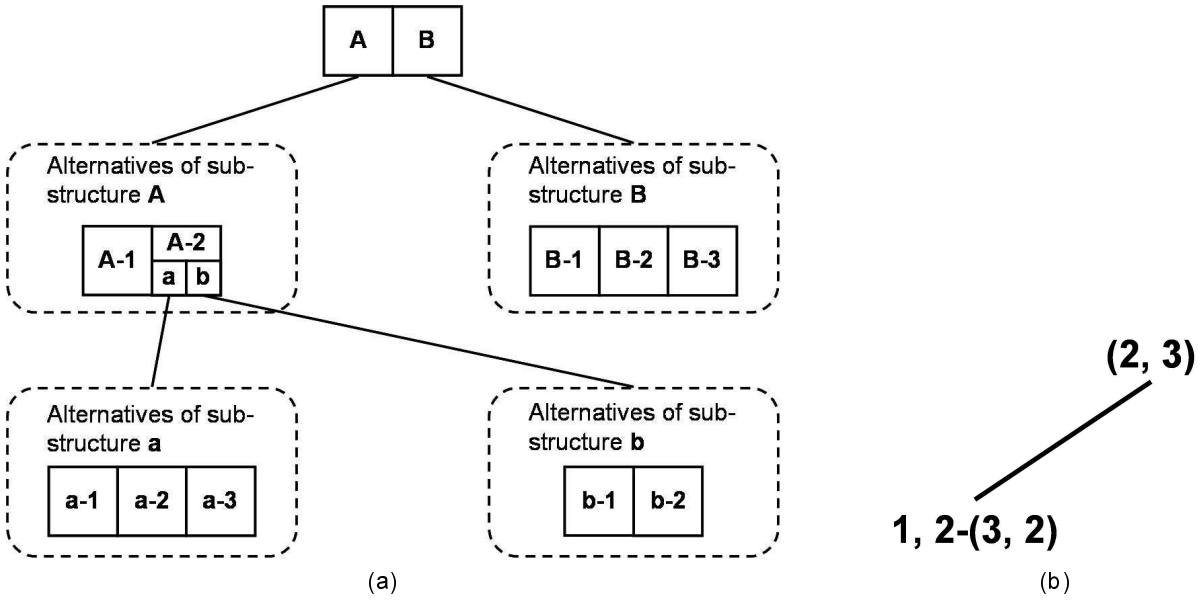

Figure 1: (a) Example of a hierarchical structure for a machine, (b) Representation of the structure of the machine as required by HGA. (Following Yoshimura and Izui [5])

Figure 1(a) shows a simple hierarchical design example, composed of substructures *A* and *B*. *A* has two alternatives and *B* has three alternatives. Alternative *A-2* has two lower substructures *a* and *b*, where *a* has three alternatives and *b* has two alternatives. Using this example, as substructure *a* has alternatives: *a-1*, *a-2*, *a-3*; and *b* has alternatives: *b-1*, *b-2*; the description is 1, 2-(3, 2). The full system is represented in Figure 1(b).

It is seen from Figure 1(a) that, by putting a system into the form required for HGA, AND operations are combined and, with HGA working best with short, wide hierarchies, these are then naturally admitted from disassembly AND/OR data. If further constraints are added to the problem, these are easily included by eliminating the appropriate paths from the problem structure. The possibility of incomplete disassembly can also be incorporated by putting extra OR options into the hierarchy allowing the solution path to stop.

Next it is shown how disassembly problems described using AND/OR information can be reformulated using the antecedents of the subassemblies such that the configuration required for HGA can be achieved. In doing this the size of the output hierarchical graphs becomes more compact – significantly, if all components are to be removed – and the benefits of generating the optimal disassembly path using HGA are gained.

For example, intricate structures can appear in even straightforward systems; in particular, complex AND/OR relationships can exist where, if C_1 – C_4 are components in an assembly, C_1 along with either C_2 or C_3 must be removed prior to C_4 [15]. Most research does not deal with these; however these relationships can be included using HGA.

4 SOLUTION METHOD AND EXAMPLES

In order that the optimal disassembly path can be found using HGA, the data used to construct the AND/OR graph must first be slightly modified. This is done by finding the antecedents of the feasible subassemblies.

AND/OR graphs are usually described via the set of all feasible subassemblies that become available during the extraction process and the operations that are allowed on each of these parent subassemblies such that two child assemblies are created. Thus, from the operations, the antecedents to each subassembly can be generated.

Antecedents of the subassemblies are used as they remove ambiguity from the possible solution paths. Precedents can cause situations where: operation *x* is preceded by operation *y*, only if *y* is not proceeded by operation *z*; this further complication is eliminated using antecedents.

The methodology is outlined as follows (this procedure has been automated.):

- First enumerating the subassemblies and operations, using these labels, the antecedents for each subassembly are found. These are written in terms "&" and "/", which are used to express AND and OR operators respectively. The operations associated with the antecedents are also listed, as well as any other optimization criteria, e.g. the profit gained in releasing the subassemblies.
- If full disassembly is required, now begins a process of amalgamating the subassemblies, during which the number of alternative paths is diminished (for incomplete disassembly only stage 1 is performed):
  1. Single components are initially removed from the list of subassemblies, as they have no antecedents, operations or profit associated with them.
  2. Subassemblies with a single AND antecedent consisting of purely of components are identified. Removing these, the antecedents of the remaining subassemblies are checked to see if they contain only the removed subassemblies. If the answer is "yes", the label of the removed subassembly is replaced with its antecedent. For example, if the only antecedent of subassembly 18 is 20&21, where 20 and 21 are components, then if subassembly 12 has antecedents 15&18 / 13&16, this is replaced by: 15&20&21 / 13&16.
  3. In a similar manner, all subassemblies with a single AND antecedent are found and removed. Again, if the antecedents of the remaining assemblies contain any of the removed subassemblies, these are replaced with the associated antecedent.

     (Remark: 3. may need to be done iteratively as the removed subassemblies may contain other removed subassemblies.)

4. In doing 2. and 3., when the antecedents are replaced, the operations and other values linked to the subassemblies are added to those already present, leading to a partial ordering of the operations and the costs/benefits carrying these out.

- With only OR operations remaining, the data is in a form that can then be optimized using HGA, such that the path that minimizes/maximizes the optimization criteria can be located.
- The flexibility of the AND/OR representation in showing parallel operations can be maintained whilst doing procedures 1–4. However, it is more simple to do a post-analysis of the output sequence to highlight whether any concurrency to the operations exists and a routine is added to achieve this.

The approach is now illustrated by way of two examples. The first of these is the famous "Bourjault's Pen" [19], a simple example to highlight the details of the solution method.

4.1 Example 1: Bourjault's Pen

The benefit of using HGA is that, not only is the structure of the assembly maintained, but it is also used at the core of the optimization process. In the remainder of this section it is shown how, by using the subassembly antecedents generated from the AND/OR information, HGA can be implemented such that optimized solutions are found.

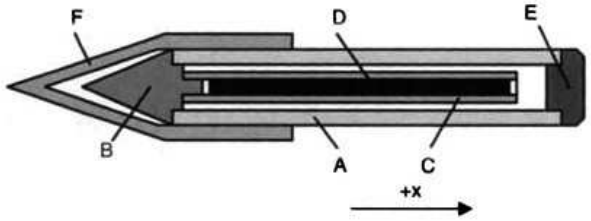

Figure 2: Pen studied by Bourjault [19]. (After Lambert and Gupta [11])

To outline the technicalities in the application of HGA in practice a deliberately simple case study is first considered. This is the well-known disassembly of a ballpoint pen, as introduced and studied extensively by Bourjault [19]. The assembly for this pen is shown in Figure 2.

Label	Subassembly
1	ABCDEF
2	ABCDE
3	ABCDF
4	ABCD
5	ABF
6	BCD
7	AB
8	AE

Label	Subassembly
9	CD
10	A
11	B
12	C
13	D
14	E
15	F

Table 1: Subassemblies for Bourjault's pen.

Here the full AND/OR problem is optimized, which includes parallel operations and the obtainable subassemblies are given in Table 1.

Following Lambert & Gupta [11], a set of operations to reach the subassemblies given in Table 2, the profit gained in doing each of these operations, is also described. The optimal disassembly path is therefore found by maximizing the total profit recovered by completely removing all of the components.

Operation	Parent	Children	Profit
(1)	1	2, 15	4
(2)	1	3, 14	5
(3)	2	8, 6	4
(4)	2	4, 14	8
(5)	3	4, 15	9
(6)	3	5, 9	1
(7)	4	10, 6	3
(8)	4	7, 9	3
(9)	6	11, 9	5
(10)	5	7, 15	9
(11)	9	12, 13	5
(12)	8	10, 14	6
(13)	7	10, 11	5

Table 2: Permissible operations upon the subassemblies and the profit recovered in performing them.

Once the operations have been defined it becomes a fairly trivial task to identify the antecedents for each subassembly. As a result, using Table 2, the information in Table 3 is generated, where "&" is used to specify that the two antecedent assemblies are released in parallel (AND operator) and "/" represents a choice between antecedents (OR operator). A graph showing the relationships between the subassemblies is given in Figure 3.

Subassembly	Antecedents	Profits	Operations
1	2&15 / 3&14	4 / 5	(1) / (2)
2	4&14 / 6&8	8 / 4	(4) / (3)
3	4&15 / 5&9	9 / 1	(5) / (6)
4	7&9 / 6&10	3 / 3	(8) / (7)
5	7&15	9	(10)
6	9&11	5	(9)
7	10&11	5	(13)
8	10&14	6	(12)
9	12&13	5	(11)
10	–	–	–
11	–	–	–
12	–	–	–
13	–	–	–
14	–	–	–
15	–	–	–

Table 3: Antecedence information for the subassemblies.

If a solution path consists of only AND operators, i.e. there are no alternative paths and only leaves of the hierarchy are involved, when optimizing, HGA merely combines any associated values. This evaluation gives the same result at all stages of resolving the system and must be done for every iteration. This redundancy is removed by combining AND operators and hence the amalgamation procedure described in 1–4 is done.

The first stage involves removing the components from Table 3. Subassemblies 10–15 are single components and these are therefore eliminated. It can also be seen that subassemblies 7–9 have solitary antecedents which consist of only components and these are also subtracted from Table. Studying the remaining subassemblies, it is ascertained that 2–6 all have antecedents containing the removed subassemblies and hence for these antecedents, "7" is replaced with "10&11", "8" with "10&14" and "9" with "12&13". Additionally the profits and operations

associated with 7–9 are added to those of the antecedents altered, leading to Table 4.

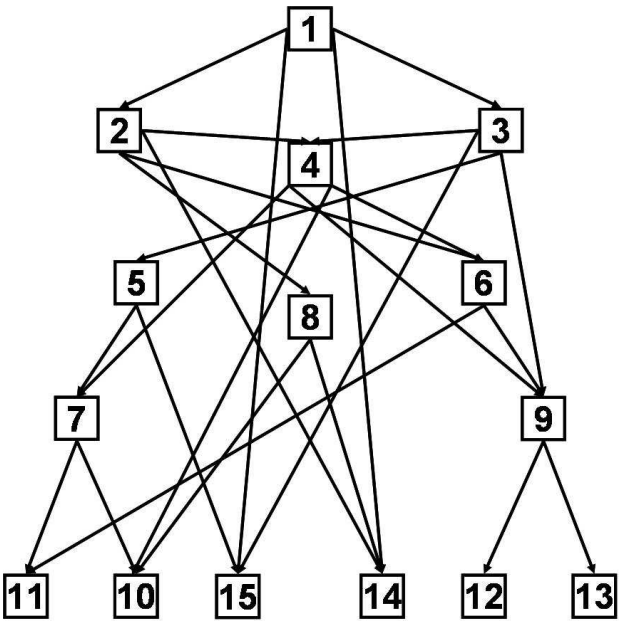

Figure 3: Antecedence graph of the subassemblies.

From Table 4, it is subsequently observed that subassemblies 5 and 6 have lone AND antecedents and so these are removed. As before, it is recognized that subassemblies 2–4 have antecedents which include those deleted and, as a result, in these antecedents "5" and "6" are replaced with "10&11&15" and "12&13&11" respectively and the related profits and operations are again incorporated into those associated with the antecedent. Finally Table 5 is obtained.

This data is in the correct form to be solved by HGA as it can be represented by the hierarchical structure given in Figure 4. In this diagram, the labels within the solid boxes are the number of choices available at each level of the hierarchy.

Sub	Antecedents	Profits	Operations
1	2&15 / 3&14	4 / 5	(1) / (2)
2	4&14 / 6&10&14	8 / 10	(4) / (3)-(12)
3	4&15 / 5&12&13	9 / 6	(5) / (6)-(11)
4	10&11&12&13 / 6&10	13 / 3	(8)-(13)-(11) / (7)
5	10&11&15	14	(10)-(13)
6	12&13&11	10	(9)-(11)

Table 4: Antecedence information after steps 1 and 2 have been performed.

When this label is "0", it represents a leaf in the tree and attached to each leaf are the operations that lead to parts of the assembly being released. The respective profit obtained by performing the operations is then shown by the numbers on the hyperarcs.

By examining Figure 4, for total disassembly of the pen, it is obvious that the maximum profit is found when the sequence of operations is either:

(2) – (5) – (7) – (9) – (11) or (2) – (5) – (8) – (13)&(11).

Optimizing the problem using HGA confirms this.

4.2 Example 2: Kang et al. Photocopier

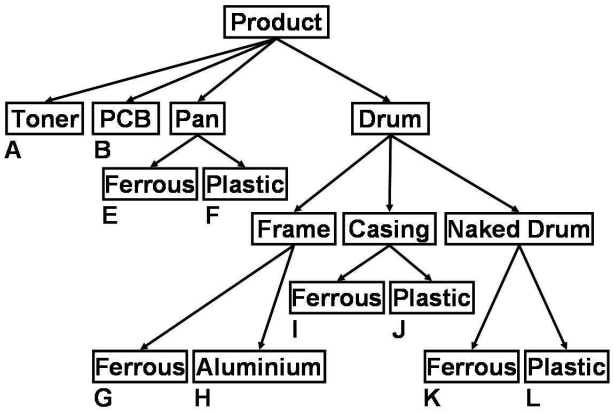

Figure 5: BOM for a photocopier (After Kang et al. [20]).

Subassemblies	Antecedents	Profits	Operations
1	2&15 / 3&14	4 / 5	(1) / (2)
2	4&14 / 12&13&11&10&14	8 / 20	(4) / (3)-(12)-(9)-(11)
3	4&15 / 10&11&15&12&13	9/ 20	(5) / (6)-(11)-(10)-(13)
4	10&11&12&13 / 12&13&11&10	13 / 13	(8)-(13)-(11) / (7)-(9)-(11)

Table 5: Antecedence information after steps 3 and 4 have been performed.

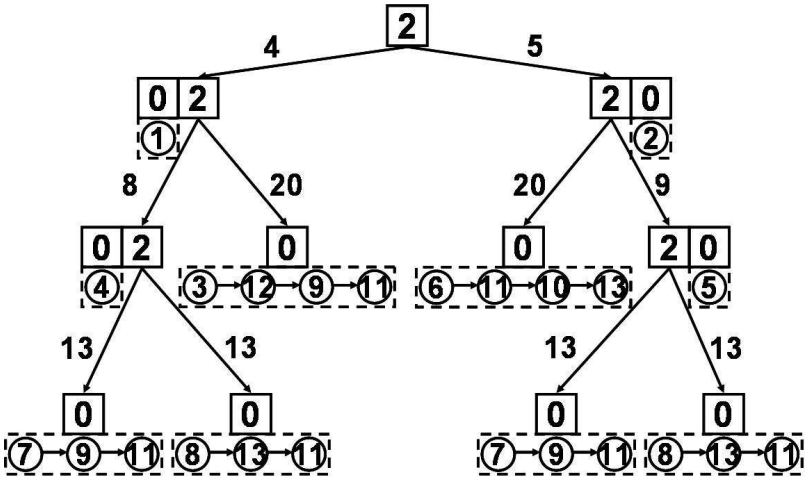

Figure 4: Resulting hierarchical structure required for HGA.

Attention is turned to the more complex and realistic industrial example of dismantling a photocopier in order to obtain the materials for recycling. This was first examined by Kang et al. [20] and, following that article, the Bill of Materials (BOM) is shown in Figure 5.

This example is used to illustrate how the possibility of incomplete disassembly is included as a solution and also how extra constraints are added to the problem structure. Whilst the former can increase the number of levels in the hierarchy, when compared to complete disassembly, the benefits of HGA to deal with both wide structures and sub-hierarchies means that the optimal path can still be obtained efficiently.

Label	Sub	Cost
1	ABE–L	965
2	AE–L	949
3	BE–L	915
4	ABG–L	880
5	E–L	829
6	AG–L	817
7	BG–L	806
8	G–L	739
9	ABEF	721
10	GHIJ	653
11	GHKL	507
12	IJKL	482
13	AEF	374
14	BEF	362

Label	Sub	Cost
15	AB	355
16	EF	321
17	GH	267
18	IJ	142
19	KL	107
20	A	43
21	B	-156
22	E	-216
23	F	-258
24	G	-370
25	H	-384
26	I	-410
27	J	-481
28	K	-528
29	L	-575

Table 6: Photocopier subassemblies and respective disposal/recycling costs.

The available combinations of materials A – L are given in Table 6. To the set up the optimization problem, a cost is associated with each combination and singular material and these are also given in Table 6.

Often it possible to recycle an assembly consisting of more than one material, if the material types are compatible. However, none of the groupings in Table 6 involve only metals or plastics and, as a result, a disposal cost is assigned to all assemblies of more than one substance.

For the individual materials, value is gained by recycling and this is reflected in Table 6, by a negative cost. The exception is the toner (material A), which is considered a hazardous substance having a harmful environmental impact, thus it must be disposed of safely.

From Kang et al. [20], the set of allowable operations to release the materials for recycling, and the cost of performing these operations, are given in Table 7. From this table the relationship between the subassemblies is depicted in Figure 5.

As in Section 4.1, the optimal disassembly path is found by maximizing the total profit recovered, where the profit is found using the formula:

$$\textit{Recovered Profit} = \textit{Cost(Parent)} - \textit{Cost(Child 1)} - \textit{Cost(Child 2)} + \text{Cost(Operation)}. \quad (1)$$

Figure 6 shows that, similarly to Bourjault's pen, if complete disassembly is required, converting the AND/OR information to a format solvable using HGA notably reduces the problem size.

Operation	Parent	Children	Cost
(1)	1	2, 21	73.50
(2)	1	3, 20	70.00
(3)	2	5, 20	61.10
(4)	3	5, 21	61.00
(5)	6	8, 20	49.80
(6)	7	8, 21	48.60
(7)	9	13, 21	30.60
(8)	9	14, 20	25.20
(9)	13	16, 20	20.30
(10)	14	16, 21	18.30
(11)	15	20, 21	12.60
(12)	16	22, 23	7.10
(13)	17	24, 25	3.60
(14)	18	26, 27	3.20
(15)	19	28, 29	3.10
(16)	1	4, 16	85.50
(17)	2	6, 16	80.80
(18)	3	7, 16	77.20
(19)	4	8, 15	74.60
(20)	5	8, 16	57.40
(21)	8	10, 19	47.00
(22)	8	11, 18	46.50
(23)	8	12, 17	39.40
(24)	9	15, 16	37.60
(25)	10	17, 18	36.20
(26)	11	17, 19	23.70
(27)	12	18, 19	21.10
(28)	1	8, 9	98.40
(29)	2	8, 13	89.50
(30)	3	8, 14	85.80
(31)	4	6, 21	53.70
(32)	4	7, 20	51.80

Table 7: Permissible operations with execution costs.

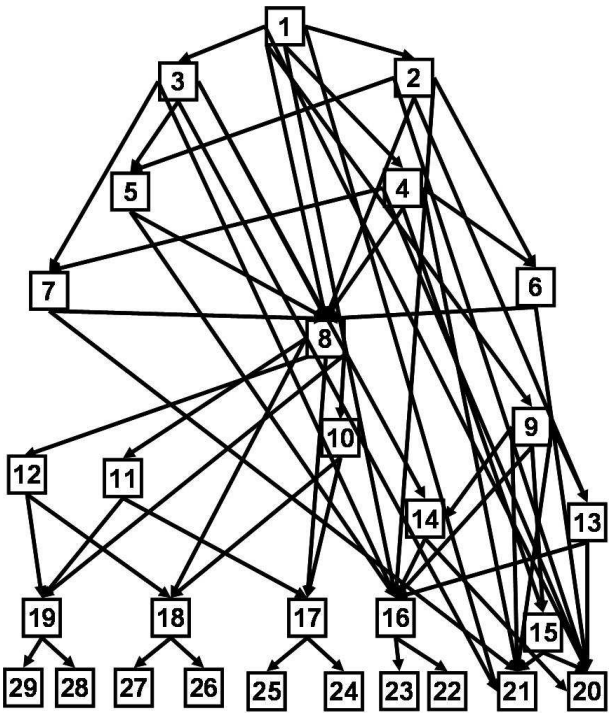

Figure 5: Antecedence graph of the subassemblies.

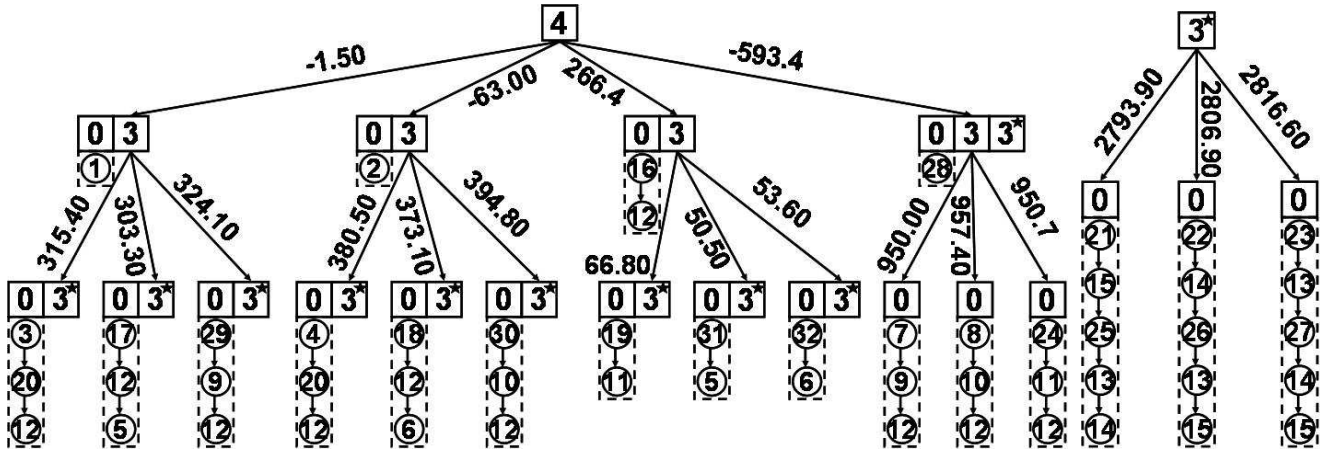

Figure 6: HGA hierarchical structure. (Note the repetition of the sub-hierarchy marked with a star.)

It can also be seen from Figure 6 that a regular occurrence in DDPs is duplication of at least one part of the hierarchy. It is particularly advantageous for HGA to recognize these recurring sub-hierarchies as they can be optimized as a pre-process and therefore do not have to repeatedly resolved, thus increasing the efficiency.

In fact, only the data pertaining to subassemblies 1, 2, 3, 4, 8 and 9 remain and, optimizing with the HGA routine, the series of operations leading to the maximum recovered profit is quickly obtained:

(28) – [(8) – (10) – (12)] & [(23) – (13)&(27) – (14)&(15)].

Adding further constraints

As HGA uses the problem structure directly, extra constraints upon the ordering of the operations are added without difficulty. This is done by simply eradicating the paths that become infeasible through the introduction of the constraints.

By way of example, suppose that it is necessary to remove the toner (subassembly 20) as early as possible within the disassembly sequence. Looking at Table 7, there are four initial operations: (1), (2), (16) and (28); of these only operation (2) involves the toner. Hence all paths that do not follow directly from operation (2) can be ignored.

As a result, the new optimal path is:

(2) – (30) – [(10) – (12)] & [(23) – (13)&(27) – (14)&(15)].

If the toner must be extracted within the first two operations then, as before, all paths emanating from operation (2) are feasible. Moreover, the disassembly paths starting with sub-sequences: (1) – (3), (16) – (12) and (28) – (8) are also viable. In this case the optimal solution reverts to that for the full problem.

Incomplete Disassembly

Determining the depth of disassembly, the extent to which components should be subtracted, is also a vital part of creating an optimal process plan.

Depending on the motives for disassembly, the “best” cause of action maybe to halt the plan partway through its execution and leave the rest of the assembly as it is. This is particularly true when balancing the value of the recovered elements with the cost of obtaining them and when performing maintenance.

To this end, a robust solution method for discovering the optimal ordering of removal of parts must incorporate the option of incomplete disassembly. Formulating the problems such that they are optimized using HGA admits this possibility.

This is done by inserting an extra choice when each “OR” function is encountered. This choice uses a dummy operation – operation (0) – and is a potential dead-end, allowing the resolution routine to stop at this point if it is deemed to be the optimal result.

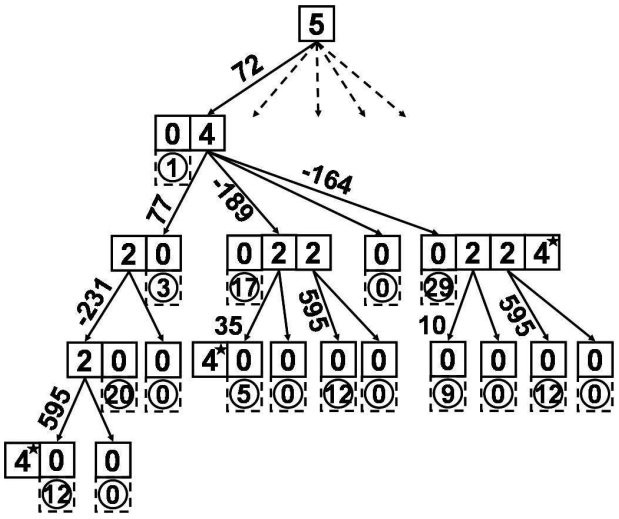

Figure 7: Including partial disassembly into the hierarchical structure. Note that the repeated sub-hierarchy denoted by the star is omitted.

Figure 7 illustrates the new solution hierarchy structure for the leftmost branch of the graph associated with the photocopier when applying this methodology. As previously, an algorithm has been constructed to automate the generation of this arrangement of the operation data.

The value associated with selecting operation (0) depends on the stage of disassembly and often corresponds to cost of disposing of what remains of the assemblage.

From Figure 7 it can be seen that the full structure of the problem must be used, it is no longer practical to condense the representation. However, although supplementary levels can be added to the hierarchy in doing this; including partial disassembly predominantly increases the width of the graph. HGA is specifically designed to cope with wide hierarchies, hence the speed and efficiency of optimization is not impaired.

Furthermore, in this situation, the benefit of finding the optimal paths for repeated/shared sub-hierarchies as a precursor to solving the main body of the problem becomes more pertinent.

By way of an example, regard the cost of the operations as being zero in all cases. The recovered profit is then a comparison between the cost of disposing of the remaining subassembly and of disposing/recycling its potential children.

The optimization criterion is taken to be that disassembling any subassembly must be advantageous. Thus disassembly only continues as long as the disposal/recycling cost of two children is less than the disposal of their parent (i.e. the profit is positive). To do this, the operation (0) is given zero value.

The optimal sequence of operations, as can be seen in Figure 7, is: (1) – (3), i.e. by performing operations (1) and (3) a profit is achieved; after this the cost of disposal of the rest of the subassembly is less than the disposal of its children.

5 SUMMARY AND CONCLUSIONS

This paper introduces a Hierarchical Genetic Algorithm, which has previously been used to optimize a number of engineering problems and successfully utilizes it to find the optimal disassembly path from the group of all feasible process plans derived from AND/OR information.

Products are often considered as modular systems, with the resulting AND/OR data that represents them having a hierarchical configuration. This structure can be exploited by solving using HGA.

The advantages of using HGA to optimize problems best depicted by AND/OR graphs is highlighted through the solution of several assemblies from the literature. In all cases, the optimal disassembly path can be found using the methodology. Additionally, by using the antecedents to convert the problem definition into that required for HGA, the size of the structure is condensed.

However, with the possibility of combinatorial explosion with AND/OR graphs, even with a reduction in the number of solution paths, there is still the difficulty that the problem may get unmanageable. For very large examples, by combining HGA with "lumping" – where the product is first separated into principle modules – and "branch and bound" techniques [18], an efficient strategy may be built to find the optimal disassembly path for even highly complex systems.

6 ACKNOWLEDGMENTS

This study was supported in part by a Grant-in-Aid for Forming Strategic Research Infrastructure from Ministry of Education, Culture, Sport, Science, and Technology, Japan (MEXT), 2008-2012 (S0801058).

7 REFERENCES

[1] Lambert, A.J.D., 1997, Optimal disassembly of complex products, International Journal of Product Research, 35(9): 2509-2523.

[2] Johnson, M.R., Wang, M.H., 1995, Planning product disassembly for material recovery opportunities, International Journal of Production Research, 33(11): 3119-3142.

[3] Wang, M.H., Johnson, M.R., 1995, Design for disassembly and recyclability: a concurrent engineering approach, Concurrent Engineering: Research and Applications, 3(2): 131-134.

[4] Yoshimura, M., Izui, K., 2000, Optimization of machine system designs based on decomposition and hierarchical ordering of criteria and design variables, Proceedings of the ASME 2000 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, 1-15.

[5] Yoshimura, M., Izui, K., 2002, Smart optimization of machine systems using hierarchical genotype representations}, ASME Journal of Mechanical Design, 124: 375-384.

[6] Kumar, R., Izui, K., Yoshimura, M., Nishiwaki, S., 2009, Multilevel redundancy allocation optimization using Hierarchical Genetic Algorithm, Reliability Engineering & System Safety, 94(4): 891-904.

[7] Kumar, R., Izui, K., Yoshimura, M., Nishiwaki, S., 2009, Optimal multilevel redundancy allocation in series and series-parallel systems, Computers & Industrial Engineering, 57(1): 169-180.

[8] Kobayashi, M., Suzuki, Y., Higashi, M., 2009, Integrated optimization for supporting functional and layout designs during conceptual design phase, Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2009-86810:1-9.

[9] Lambert, A.J.D., 1999, Linear programming in disassembly/clustering sequence generation, Computers and Industrial Engineering, 36: 723-738

[10] Delchambre, A., Wafflard, A., 1991, An automatic, systematic and user-friendly computer-aided planner for robotized assembly, Proceedings of the 1991 IEEE International Conference on Robotics and Automation, 592-598.

[11] Lambert, A.J.D., Gupta, S.M., 2005, Disassembly Modeling for Assembly, Maintenance, Reuse, and Recycling, CRC, Boca Raton.

[12] Subramani, A.K., Dewhurst, P., 1991, Automatic generation of product disassembly sequences, Annals of the CIRP, 40(1): 115-118.

[13] Homem de Mello, L.S., Sanderson, A.C, 1989, Representations of Assembly Sequences, International Joint Conferences on Artificial Intelligence, 1035-1042.

[14] Lambert, A.J.D., 2001, Automatic determination of transition matrices in optimal disassembly sequence generation, Proceedings of the 4th IEEE International Symposium on Assembly and Task Planning, 220-225.

[15] Homen De Mello, L.S., Sanderson, A.C., 1990, AND/OR graph representation of assembly plans, IEEE Transactions on Robotics and Automation, 6(2): 188-199.

[16] Moore, K.E., Güngör A., Gupta, S.M., 2001, Petri net approach to disassembly process planning for products with complex AND/OR precedence relationships, European Journal of Operational Research, 135: 428-449.

[17] Zussman, E., Kriwet, A., Seliger, G., 1994, Disassembly-orientated assessment methodology to support design for recycling, Annals of the CIRP, 43(1): 9-14.

[18] Güngör, A., Gupta, S.M., 2001, Disassembly sequence plan generation using a branch-and-bound algorithm, International Journal of Production Research, 39(3): 481-509.

[19] Bourjault, A., 1987, Methodology of assembly automation: a new approach, Abstracts of 2nd International Conference on Robotics and Factories of the Future, 37-45.

[20] Kang, J.-G., Lee, D.-H., Xirouchakis, P., Persson, J.-G., 2001, Parallel disassembly sequencing with sequence-dependent operation times, Annals of the CIRP, 50(1): 343-346.