Bibliographic Information


Copyright Information

Copyright 2010, Surendra M. Gupta.

Contact Information

Dr. Surendra M. Gupta, P.E.
Professor of Mechanical and Industrial Engineering and
Director of Laboratory for Responsible Manufacturing
334 SN, Department of MIE
Northeastern University
360 Huntington Avenue
Boston, MA 02115, U.S.A.

(617)-373-4846 Phone
(617)-373-2921 Fax
gupta@neu.edu e-mail address

http://www.coe.neu.edu/~smgupta/ Home Page
A GENETIC ALGORITHM APPROACH TO END-OF-LIFE DISASSEMBLY SEQUENCING FOR ROBOTIC DISASSEMBLY

Ahmed ElSayed*, Elif Kongar† and Surendra M. Gupta‡

ABSTRACT

End-of-life (EOL) processing options include reuse, remanufacturing, recycling and proper disposal. In almost all cases, a certain level of disassembly may be required due to possible changes in the original product structure. Thus, finding an optimal or near optimal disassembly sequence is crucial to increasing the efficiency of the process. Disassembly operations are labor intensive, can be costly, have unique characteristics and cannot be considered as reverse of assembly operations. Since the complexity of determining the best disassembly sequence increases as the number of parts of the product grow, an efficient methodology is required for disassembly sequencing. In this paper, we present a Genetic Algorithm for disassembly sequencing of EOL products. A numerical example is provided to demonstrate the functionality of the algorithm.

Keywords: Disassembly sequencing, genetic algorithm, product recovery, robotics.

1. INTRODUCTION

The growing amount of waste generated by the end-of-life (EOL) of products has become a severe problem in many countries. This fact, couple with the decreasing number of landfills and virgin resources has led to extended product and/or producer responsibility policies that mandate manufacturers to take-back their products at the end of their lives. In response, the manufacturers are now seeking solutions to address the potential accumulation of large inventories consisting of technologically invalid and/or non-functioning products. The challenge is to process these products in an environmentally benign and cost effective manner. There are various ways to accomplish this task under the general umbrella of end-of-life (EOL) processing, including reuse, recycle, and storage for future use or proper disposal.

In majority of EOL processing, a certain level of disassembly may be required. Disassembly can be partial or complete and may use a methodology that is destructive or non-destructive. In this paper, the case of complete disassembly is considered where the components are retrieved by either destructive or non-destructive methodology.

* Ahmed ElSayed, Department of Computer Science and Engineering, University of Bridgeport, 221 University Avenue, School of Engineering, 141 Technology Building, Bridgeport, CT 06604, E-mail : aelsayed@bridgeport.edu.
† Elif Kongar, Ph.D., Departments of Mechanical Engineering and Technology Management, University of Bridgeport, 221 University Avenue, School of Engineering, 141 Technology Building, Bridgeport, CT 06604, USA, Phone: (203) 576-4379, Fax: (203) 576-4750, E-mail : kongar@bridgeport.edu.
‡ Surendra M. Gupta, Ph.D., P.E., Mechanical and Industrial Engineering and Director of Laboratory for Responsible Manufacturing, 334 SN, Department of MIE, Northeastern University, 360 Huntington Avenue, Boston, MA 02115, U.S.A., Phone: (617) 373-4846, Fax: (617) 373-2921, E-mail : gupta@neu.edu.
When the electronic products are concerned, which is the focus of this paper, modeling EOL processing problems can be complex due to the number of components in the product structure. In these cases, exhaustive search algorithms may mathematically be inefficient making heuristic methods more appealing for obtaining near-optimal or optimal solutions. With this motivation, this paper presents a genetic algorithm for optimal or near optimal disassembly sequencing in the presence of constraints and precedence relationships.

2. LITERATURE REVIEW

In recent years, genetic algorithm (GA) has been gaining popularity for solving combinatorial and NP-complete problems. One of the multi-objective optimization applications was proposed by Valenzuela-Rendón and Uresti-Charre [1]. Keung et al. [2] also applied a multi-objective GA approach to a tool selection model. Lazzerini and Marcelloni [3] used GA in scheduling assembly processes. In the area of disassembly, Kongar and Gupta [4] proposed a GA for disassembly sequencing problem, while McGovern and Gupta [5] applied genetic algorithm to disassembly line balancing. For further literature on disassembly scheduling and processing, see Ilgin and Gupta [6], Gungor and Gupta [7], Lee et al. [8], and Lambert and Gupta [9].

Precedence relationships are one of the factors that add to the complications in sequencing problems. In this regard, Sanderson et al. [10] considered precedence relationships in assembly sequence planning in such a manner. Seo et al. [11] proposed a genetic algorithm for generating optimal disassembly sequences considering both economical and environmental factors. Bierwirth et al. [12] and Bierwirth and Mattfeld [13] proposed a methodology to overcome this problem by introducing the precedence preservative crossover (PPX) technique for scheduling problems which preserves the precedence relationships during the crossover function of GA. This approach is also employed in this paper.

Furthermore, Shimizu et al. [14] developed a prototype system for strategic decision-making on disassembly for recycling at the design stage of the product lifecycle. Hui et al. [15] utilized a genetic algorithm to determine feasible and optimal disassembly solutions.

In the area of automated disassembly, Torres et al. [16, 17] presented a personal computer disassembly cell that is able to handle a certain degree of automatism for the non-destructive disassembly process. This work is then followed by Pomares et al. [18] who generated an object-oriented model required for developing a disassembly process. Gil et al. [19] proposed a flexible multi-sensorial system for automatic disassembly using cooperative robots. As a follow-up work, Torres et al. [20] presented a task planner for disassembly process based on decision trees.

3. GENETIC ALGORITHM FOR DISASSEMBLY SEQUENCING

Genetic algorithm starts with a set of randomly selected potential solutions called the population. Each member of the population is encoded as an artificial chromosome which contains information about the solution mapping. Each chromosome is assigned a score
based on a predefined fitness function. A new population of chromosomes is iteratively created in the hope of finding a chromosome with a better score. At each step of creation, a mutation may occur in a chromosome, and/or two chromosomes may mate to produce a child; this process is known as crossover. The selection of parent chromosomes is biased towards fitter members of the population. The process is iterated until some predetermined termination conditions are satisfied.

![Figure 1. Original product structure of the EOL product](image)

<table>
<thead>
<tr>
<th>Item#</th>
<th>Description</th>
<th>Material</th>
<th>Time (sec)</th>
<th>Method</th>
<th>Coordinates (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Back/Front Cover</td>
<td>Aluminium (A)</td>
<td>2</td>
<td>Destructive (D)</td>
<td>(30, 0, 10)</td>
</tr>
<tr>
<td>1</td>
<td>Side Cover</td>
<td>Aluminium (A)</td>
<td>3</td>
<td>Destructive (D)</td>
<td>(30, 40, 10)</td>
</tr>
<tr>
<td>2</td>
<td>Power Supply</td>
<td>Copper (C)</td>
<td>3</td>
<td>Destructive (D)</td>
<td>(50, 35, 5)</td>
</tr>
<tr>
<td>3</td>
<td>System Fan</td>
<td>Plastic (P)</td>
<td>2</td>
<td>Destructive (D)</td>
<td>(50, 40, 5)</td>
</tr>
<tr>
<td>4</td>
<td>RAM</td>
<td>Plastic (P)</td>
<td>3</td>
<td>Non-Destructive (N)</td>
<td>(40, 10, 0)</td>
</tr>
<tr>
<td>5</td>
<td>DVD Drive</td>
<td>Aluminium (A)</td>
<td>4</td>
<td>Non-Destructive (N)</td>
<td>(40, 7, 7)</td>
</tr>
<tr>
<td>6</td>
<td>Hard Drive Slot</td>
<td>Plastic (P)</td>
<td>2</td>
<td>Destructive (D)</td>
<td>(35, 5, 7)</td>
</tr>
<tr>
<td>7</td>
<td>CPU</td>
<td>Plastic (P)</td>
<td>1</td>
<td>Non-Destructive (N)</td>
<td>(40, 20, 0)</td>
</tr>
<tr>
<td>8</td>
<td>Heat Sink Fan</td>
<td>Aluminium (A)</td>
<td>3</td>
<td>Destructive (D)</td>
<td>(35, 20, 0)</td>
</tr>
<tr>
<td>9</td>
<td>Hard Drive</td>
<td>Aluminium (A)</td>
<td>2</td>
<td>Non-Destructive (N)</td>
<td>(40, 5, 7)</td>
</tr>
</tbody>
</table>

### 3.2 Elements of Genetic Algorithm for Disassembly Sequencing

The structure of the example EOL product and related data are provided in Figure-1. The given product consists of ten components indexed by integers from 0 to 9. Therefore, \( j \in \{0, 1, \ldots, n-1\} \). Locations of each component is defined by their 3-D coordinates. The methodology used for the disassembly of each component may be destructive (D) or non-destructive (N). A component may or may not be demanded. If it is not demanded, it is represented by \( s \). If it is demanded, it may be demanded for reuse (u) or recycling (r). A component has one of three types of joints, viz., latch (L), screw (S) or slot entry (E). In
addition, the precedence relationships are given as follows: component 1 or 2 must be
disassembled prior to any other component; component 3 must be disassembled prior to
components 7 and 8; and component 6 must be disassembled prior to component 9.

**Chromosomes:** In every GA problem, the solution and parameters must be coded into
chromosomes, represented by a combination of numbers, alphabets and/or other
characters, before they can be processed. In this study, in order to capture the five
variables, chromosomes are codified in the form of a string consisting of five ordered
sections of equal length, representing the disassembly sequence, the disassembly method,
the demand type of each component, and the material type of each component
respectively. Coordinate data are kept separate from the chromosome structure to provide
ease in calculations.

**Initial Population:** The initial population consists of \( ncr \) random chromosomes that
satisfy the precedence relationships and any other constraints imposed by the product
structure. For the example provided in Figure-1, hundred chromosomes (feasible
solutions) are randomly created to form the initial population \( (ncr = 100) \).

The chromosomes in the initial random population is provided in Table-1 (10 out
of 100 are shown).

Table 1. Partial initial population for the genetic algorithm example

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Method</th>
<th>Demand</th>
<th>Material</th>
<th>( F(ch,gn) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0623795418</td>
<td>CDDDDNNNDN</td>
<td>rrrruusrr</td>
<td>APCCPPAAPAA</td>
<td>28.978</td>
</tr>
<tr>
<td>0613275948</td>
<td>CDDDDNNNDN</td>
<td>rrrruusrr</td>
<td>APAPCPAAPA</td>
<td>29.881</td>
</tr>
<tr>
<td>1630824957</td>
<td>CDDDDNNNDN</td>
<td>rrrrrsuuu</td>
<td>APPAACPAAP</td>
<td>30.563</td>
</tr>
<tr>
<td>0327468195</td>
<td>CDDNNDNNDN</td>
<td>rrrusrrruu</td>
<td>APCCPAPAAA</td>
<td>30.586</td>
</tr>
<tr>
<td>1328047659</td>
<td>CDDDDNNNDN</td>
<td>rrrrsuruu</td>
<td>APCAAPPAPA</td>
<td>30.813</td>
</tr>
<tr>
<td>1385206479</td>
<td>CDDNNDNNDN</td>
<td>rrrrrsuuu</td>
<td>APCAAPPAPA</td>
<td>35.052</td>
</tr>
<tr>
<td>0463159827</td>
<td>CDDDNDDNNDN</td>
<td>rrrruuuru</td>
<td>APPAAPAAP</td>
<td>35.137</td>
</tr>
<tr>
<td>134276598</td>
<td>CDDNDNDDNND</td>
<td>rrrruuruu</td>
<td>APPAACPAAP</td>
<td>35.142</td>
</tr>
<tr>
<td>1385476209</td>
<td>CDDNNDDNNDN</td>
<td>rrrusrrruu</td>
<td>APAAAPPCCA</td>
<td>35.296</td>
</tr>
<tr>
<td>0213657984</td>
<td>CDDDDNNDNDN</td>
<td>rrrruuuruu</td>
<td>ACAPPAPAAP</td>
<td>35.430</td>
</tr>
</tbody>
</table>

**Crossover:** This study employed the precedence preservative crossover (PPX)
methodology for crossover. In this methodology, in addition to the two strings
representing the chromosomes of the parents (Parent\(_1\) and Parent\(_2\)), two additional strings
pass on the precedence relationship based on the two parental permutations to two new
offspring while making sure that no new precedence relationships are introduced. A
vector, representing the number of operations involved in the problem is randomly filled
with elements of the set. This vector defines the order in which the operations are
successively drawn from Parent\(_1\) and Parent\(_2\).

The algorithm starts by initializing an empty offspring. The leftmost operation in
one of the two parents is selected in accordance with the order of parents given in the
vector. After an operation is selected it is deleted in both parents. Finally, the selected
operation is appended to the offspring. This step is repeated until both parents are empty
and the offspring contains all operations involved.
**Mutation:** The population is subjected to mutation operation with a given probability. If the probability holds, the mutation operator selects a random number of genes \((rnd = 1, \ldots, 9)\), and exchanges them in such a way that the same precedence relationships are preserved. Otherwise the population remains unchanged and is copied to the next generation. The mutation operator proposed in this paper exchanges components 0 and 1. The rest of the strings remain the same.

**Fitness Evaluation:** The fitness function is dependent on the increment in disassembly time. There are three factors, which add up to the disassembly time of a component. The first one is basic disassembly time for component \(j\) in sequence \(seq\) \((dt_{j,seq})\). In this paper \(dt_{j,seq}\) values (in seconds) are given as:

<table>
<thead>
<tr>
<th>(j)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>(dt_{j,seq})</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

The second function \((ct_{j,seq})\) is the penalty (in seconds) for each travel time to disassemble component \(j\) in sequence \(seq\), which includes a function of the distance traveled between the \((seq-1)\)\(^{th}\) and \(seq\)\(^{th}\) sequences and the robot arm speed factor \((sf)\):

\[
ct_{j,seq} = \sqrt{(x_{j,(seq-1)}-x_{j,seq})^2 + (y_{j,(seq-1)}-y_{j,seq})^2 + (z_{j,(seq-1)}-z_{j,seq})^2} / sf
\]

The last criterion in fitness function is the penalty for disassembly method change \((mt_{j,seq})\). For each disassembly method change, the sequence is penalized by 1 second:

\[
mt_{j,seq} =
\begin{cases} 
0, & \text{If no method change is required,} \\
1, & \text{If method change is required,}
\end{cases} \quad (e.g. N to N)
\begin{cases} 
1, & \text{If no method change is required,} \\
0, & \text{If method change is required,}
\end{cases} \quad (e.g. N to D)
\]

In addition, the algorithm searches for a “recycling pair” and does not penalize the sequence if the two adjacent components are made of the same material and if they are both demanded for recycling.

Let \(T_{seq}\) denote the cumulative disassembly time after the disassembly operation in sequence \(seq\) is completed for component \(j\):

\[
T_{seq} = T_{seq-1} + dt_{j,seq} + ct_{j,seq} + mt_{j,seq}, \quad \text{for } seq = 0, \ldots, n-2
\]

\[
T_{seq} = T_{seq-1} + dt_{j,seq}, \quad \text{for } seq = n-1
\]

In this proposed GA model, the objective is to minimize the total fitness function \((F)\) by minimizing (i) the traveled distance, (ii) the number of disassembly method changes, and (iii) by combining the identical-material components together, eliminating unnecessary disassembly operations. Let \(F(ch, gn)\) denote the total fitness for chromosome \(ch\) in generation \(gn\). Hence, total time to disassemble all the components can be calculated as follows:

\[
F(ch, gn) = \sum_{seq=0}^{n-1} dt_{j,seq} + \sum_{seq=0}^{n-2} ct_{j,seq} + \sum_{seq=0}^{n-2} mt_{j,seq}, \quad \forall j, j = 0, \ldots, n-1.
\]

**Selection and Regeneration Procedure:** After every generation, the chromosomes obtains a certain expectation depending on their fitness values. A roulette wheel is then implemented to select the sequence of parents that will be included in the next generation.
(the higher the fitness value the higher the chance to be selected). This method aims at allowing the parents in the current generation to be selected for the next generation without getting trapped in the local optima. In addition, a new population is generated eliminating the weak chromosomes.

Termination: The execution of GA terminates if the number of generations reaches up to a maximum value (100 in our example).

3.3 Genetic Algorithm Model Assumptions

Proposed model assumes that the end effector speed for the robot arm is a constant value of 25 cm/sec. In addition, the time spent for robot arm angle change (for all three angles) is assumed to be embedded in the disassembly time for each component. In addition, every component is assumed to have one joint that connects the component with the rest of the product structure.

4. NUMERICAL EXAMPLE

Consider the product in Figure-1. The crossover and mutation probabilities are assumed to be 0.60 and 0.005 respectively. Initial population provided in Table 1 consists of 100 chromosomes \( ncr = 100 \). After the GA is run, only one optimal solution is obtained in the final population with a fitness function value of 26.0248 seconds.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>( F(ch, gn) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 3 2 8 0 6 9 5 4 7 DDDDDDNNNN rrrrrusu APCAAPAAPP</td>
<td>26.0248 secs</td>
</tr>
</tbody>
</table>

The time to calculate the optimal solution (Table-2) took 2.4180 seconds using the exhaustive search algorithm where as this time was increased to 2.5576 seconds for the genetic algorithm. The solution was reached at the 6th generation. The initial solution of 100 chromosomes included the four duplicates, identical chromosomes, where as the rest was unique sequences. It is important to note that the sequence is found in a few iterations even though it did not take place in the initial random population.

As for the average values of 100 runs, the average time to obtain an optimal solution took 2.5576 seconds, and the optimal solution was reached at the 16.56th generation at the average. The code is written in Matlab (version 7.7.0.471(R2008b)) using Genetic Algorithm and Direct Search Toolbox. The code was run on a computer with a Processor Intel Core 2 Duo CPU P8400 2.26 GHz, 4.00 GB RAM (Table-4).

5. CONCLUSIONS

In this paper, a genetic algorithm model was presented in order to determine the optimal disassembly sequence of a given product. The model provides quick and reliable input to the disassembly scheduling environments. As also stated by Keung et al. [2], GAs do not make unrealistic assumptions such as linearity, convexity and/or differentiability. This adds further importance to the proposed model and makes it even more desirable. For the example considered, the algorithm provided optimal disassembly sequence in a short execution time. The algorithm is practical, as it is easy to use, considers the precedence
relationships and additional constraints in the product structure and is easily applicable to problems with multiple objectives.

References