INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'09
24 - 27 AUGUST 2009, STANFORD UNIVERSITY, STANFORD, CA, USA

AUTOMATED ANALYSIS OF PRODUCT DISASSEMBLY TO DETERMINE ENVIRONMENTAL IMPACT

David Ikechukwu Agu[1] and Matthew I. Campbell[2]
(1) The University of Texas at Austin, USA (2) The University of Texas at Austin, USA

ABSTRACT
Manufacturers are increasingly being held responsible for the fate of their products during the end-of-life phase. In this research, a product's end-of-life environmental impact is calculated as a function of cost and recyclability. Cost is associated with the time and effort of removing specific components. Recyclability is governed both by the material used and how components of different materials are connected. This paper presents a graph grammar based algorithm which will analyze any product given information concerning individual components and how they are connected within the overall assembly. The result of this analysis is a set of Pareto optimal candidates representing various stages in the disassembly process and evaluated using associated costs and recyclability. This Pareto set can be used to judge a product's end-of-life suitability against the manufacturer's or industry standards, or against the suitability of a similar product.

Keywords: Automated disassembly, environmental impact, Pareto

1 INTRODUCTION
The consumer product market is continually evolving. Traditionally, we have used and discarded products with little regard for the environmental impact of our actions. Recently however, there have been signs that these practices are changing. On one hand, closer attention has been paid to the consumption of raw materials. Many resources are in increasingly scant supply and close attention must be paid to ensure that the limited reserves which are still available are not used to the point of exhaustion. On the other hand, disposal practices are also being investigated. Disposal falls into what is called the "end-of-life" phase of a product. In recent years producers have begun to pay more attention to the fate of their products once they reach their end-of-life. In fact, recent legislation, called Extended Producer Responsibility (EPR) initiatives, has begun to ensure this change.
The United States currently lags behind other economically developed countries in formally regulating producer responsibility in the end-of-life phase. The European Union and Japan have already enacted government legislation which forces producers to meet certain environmentally-conscious standards with their products. EPR mandates that certain requirements are met before a specific product is allowed to reach the market. The United States has begun to look in this same direction however EPR legislation currently exists only at the state level [1]. As of May 2009, thirty states have passed varying levels of EPR initiatives into law but there are no signs that federal legislation will be enacted in the near future [2]. This becomes particularly relevant after realizing how much waste the United States produces and the degree to which waste processing is becoming a sizeable industry. For example, in 2006 exports of waste from the United States to China were valued at $6.7 billion [3]. In addition to the legislative aspects, producers also cater to increasingly selective customers. Many consumers in the United States now make selections depending on the "environmental friendliness" of one product over another.
For these reasons, environmental considerations are becoming increasingly important for producers. This research proposes a tool which can be used to evaluate the environmental impact of any given product. Such a tool would be useful for both producers and consumers and also those responsible for enforcing and verifying any environmental standards.

2 DEFINING ENVIRONMENTAL IMPACT

Before evaluating a product's performance, environmental impact must be defined. The environmental impact would be of interest at the end-of-life phase of a product when disassembly is to take place. The goal of the disassembly process may be to separate components from each other for repair and resale, or to separate down to the raw materials so that they can be recycled. This process occurs in a series of steps which begins with an entire product and progressively separates individual components away from the main assembly. Each step in the process can be evaluated based on cost and recyclability. Cost is considered to be a function of the time taken to disassemble a product and also the value of the materials contained within the assembly. Recyclability is linked with a parameter called wasted weight which is the amount of material still contained within the assembly which cannot be recycled. A component within the assembly cannot be recycled if it is connected to another component which is of a different material type and so both of their weights are included in the total wasted weight. It is these parameters which are used to define the environmental impact of a product, not only at its completely disassembled state but at various stages in the disassembly process.

For any given product there may be many different orders in which it may be taken apart. Depending on the order, the disassembly time may vary. It would be advantageous to know which order minimizes the disassembly time and simultaneously minimizes the wasted weight prior to committing to a particular disassembly sequence. This research culminates in a computer-based tool which can both automate and optimize the disassembly process, and simultaneously present results on the environmental impact of the product.

3 AUTOMATED DISASSEMBLY

The first step in evaluating a product is to run a simulation of the disassembly process based on a suitable graphical representation of the assembly. This project was initiated by the Automated Design Lab at the University of Texas at Austin which had previously developed a software tool called GraphSynth [4] for use in automated design projects. This tool provides a convenient means for both representing product assemblies and analyzing their disassembly.

Automated disassembly is not a new field of mechanical design. Many different methods have been created to both represent product assemblies and evaluate their disassembly. In GraphSynth, the user is allowed to build a graph-based representation of any product. In the approach used here, nodes represent product components while arcs, which connect nodes to each other, represent connections between two components (e.g. welds, screws, etc.).

Complete disassembly is often not the goal of the disassembly process. For example, if repairs are being made to a product, only one component needs to be removed and it would be beneficial to find the easiest method of reaching a specific, faulty component. However, when looking at the recycling and reuse of components, the goal would likely be to find a more complete disassembly sequence. Therefore, when looking at the disassembly process it is very important to differentiate component connection types from each other.

3.1 Component Connection Types

Before beginning a discussion of the different connection types that have been used in this research it is important to point out the coordinate system that is used in defining certain connection types. An inertial, Cartesian coordinate system with dual polarity (positive and negative) in each of three directions (X, Y and Z) is used. The coordinate axes are to be defined by the user in the most logical and/or convenient orientation with respect to the fully assembled product.

The approach used to build a list of component connection types was to take existing products and systematically remove components from them while taking note of the various connections being used. This approach is an ongoing process since the investigation of more diverse products will likely lead to the discovery of different connection types. At present, three products have been investigated: a Dell Latitude laptop, a Toro Model #51586 leaf blower and a Troy-Bilt TB190BV leaf blower. The connections which have been found so far can be divided into the following categories: rectangular constraints, virtual rectangular constraints, radial constraints, threaded fasteners, press fits, adhesives and windings. Each arc in the graphical representation must have at least one connection type and this information is stored as a label within the arc.

Two additional arc types which often do not represent actual, physical connections are the rectangular and virtual rectangular constraints. A rectangular constraint exists between two objects when, due to

the positioning of one component with respect to another, one of the components cannot be removed from the entire assembly if translated in a specific direction. The virtual rectangular constraint is very similar but the two connected or related objects would only come into contact if another is removed first. This concept is explained further in section 3.2 with the cascading rule. The graphical representation of a rectangular constraint between two components in GraphSynth is shown in Figure 1.

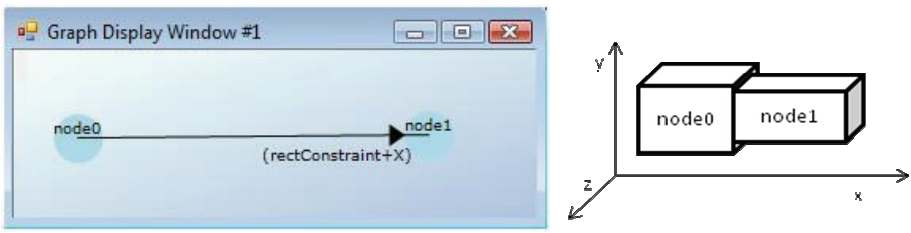


Figure 1. Rectangular constraint in GraphySynth (left) and a three-dimensional representation of the situation (right).

The graphical representation can easily be understood by forming a statement from what is depicted. It can be said that node0 is constrained by node1 in the positive X direction by a rectangular constraint. Therefore (and this is true for all rectangular constraints) there would also be a rectangular constraint imposed on node1 on by node0 along the same Cartesian axis but with opposite polarity. When building the entire assembly it is not necessary for the user to draw both arcs in each pair of rectangular constraints. The user may simply draw one labeled arc and a series of "fixer rules" – which will be explained in section 3.2 – will automatically draw the second arc with an appropriate label. Since there is one type of rectangular constraint for each polar direction along each Cartesian axis, there are a total of six rectangular constraints that may be used to describe a real system. The labels corresponding to each of these constraints is written as rectConstraint followed by a directional suffix, e.g. +X or –Z.

The second group of connections, radial constraints, works much in the same way as rectangular constraints visually however their meaning is slightly different. Radial constraints are used most often for components positioned around a shaft. In this case the directional subscript after the constraint name (radConstraint) does not have polarity attached to it. The direction indicates the Cartesian axis along which the axis of the shaft is directed. Therefore, there are only three types of radial constraints. Like rectangular constraints, radial constraints also exist in pairs between components however – unlike rectangular constraints – they are identical since polarity does not exist. Another interesting relationship between radial and rectangular constraints is that radConstraintX, for example, carries nearly identical meaning to the following set of rectangular constraints: rectConstraint+Y, rectConstraint-Y, rectConstraint+Z and rectConstraint-Z.

Threading constraints exist between two components which have adjacent threaded surfaces. There are six possibilities within the thread group (thread+X, etc.). The directional suffix indicates the direction in which one of the two components is removed. Unlike the two previous groups, thread constraints do not exist in pairs. There is only one arc between the two components in question and the arc is directed (depicted graphically with an arrow) towards the object which is removed. This object is typically a screw, a bolt or a nut.

The press fit group is self-explanatory and exists between objects which are press-fitted together. Like rectangular constraints there are six variations. The directional suffix indicates the direction in which a specific component is press-fitted onto/into another. For example if an arc originates from node0, terminates at node1 and carries the label pressFit+X then component 0 is press-fitted onto component 1 in the positive X direction.

The adhesive group is unique in that there are no directional suffixes and the graphical representation features arcs which have arrowheads on both ends (as shown in Figure 2). This "group" therefore only contains one connection possibility. The constraint exists when two components are joined using any type of adhesive.

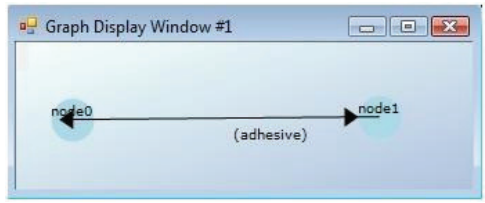


Figure 2. Adhesive constraint.

The winding rules represent situations in which one component is wound around another. This may be used in instances such as a length of tape wrapped around parts or electrical wire wrapped in a coil. The three variations correspond to the three Cartesian axes. The axial direction of the "shaft" which the coil is wound around is the direction assigned to the winding arc. The arc originates at the node representing the shaft and terminates at the node representing the wound material.

With this completed list of component connection types it is possible to create a graphical representation of a completed product. Graphical representations of the leaf-blowers from Toro and Troy-Bilt can be seen in Appendix A and Appendix B respectively. Also take note that in each of these diagrams a directional suffix can be seen in the top left-hand corner of the window. This direction gives the disassembly algorithm a starting orientation for the product and is intended to make the computer simulation more realistic. Many disassembly operations, such as removing a bolt, are performed with the product resting on a table. As was stated previously, the user chooses the orientation of the coordinate system with respect to the product. Using this pre-defined orientation, the user would also choose one of the six surfaces (virtually estimating the assembly to be contained within a rectangular prism) to start off resting on a table. This surface is what is indicated in the top left-hand corner and within the program is termed a global variable. Now that the connection types are known it is necessary to define how these connections can be removed if a simulation representing a real-life disassembly process is to be run.

3.2 Disassembly Grammar Rules

After defining types of component connectivity, it is necessary to define the methods which are used to disassemble a product. A series of "grammar rules" have been created along with the various conditions necessary for their recognition and application. Each one of these grammar rules represents an actual method of separating two components, such as the adhesive between two components. There are four groups of grammar rules which are termed rule sets and can be described as follows:

- Rule set #0: Fixer rules,
- Rule set #1: Disassembly rules,
- Rule set #2: Flip rules,
- Rule set #3: Module rule.

The easiest of these four groups to understand in the context of the disassembly process and also the most important in terms of analyzing this process is rule set #1. The rules in this set estimate the separation of components from each other as a product is being disassembled. In GraphSynth grammar rules are displayed visually as having a left and right hand side. The grammar rule that breaks an adhesive bond between two components is shown in Figure 3.

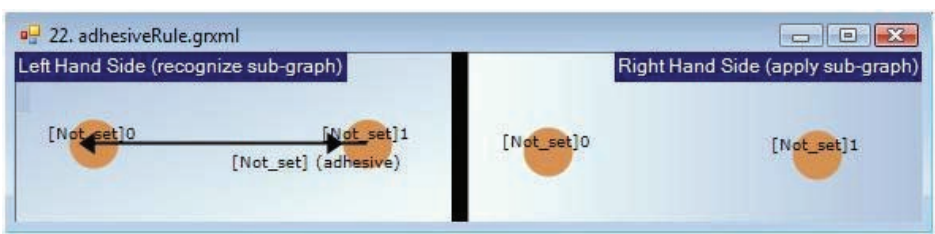


Figure 3. Adhesive grammar rule.

The left hand side of the grammar rule is used for rule recognition. The image is visually matched with an identical set of nodes and arcs in the graph of the overall assembly. The right hand side is used for rule application and shows the result of the application of the rule. Therefore this rule is recognized whenever two components are bonded together with an adhesive and the application of the rule results in the removal of the adhesive connection, whether it is by hand or with some type of solvent (these can be specified by providing extra information which will be explained in section 4.2). Each of the connections specified in section 3.1 has a corresponding grammar rule which describes how a connection between components is separated. There there are six grammar rules for rectangular constraints, three grammar rules for radial constraints and so on; this leads to a total of 26 disassembly rules.

The fixer rule set, rule set #0, does not include any disassembly operation information. Rather, this rule is included to save time while creating the graph of a product assembly and to fix some of the errors that a user may easily make. The rectangular "single" fixer will add the second arc – with corresponding labels – which exists between nodes if the user only draws one of the two rectangular constraints. The rectangular "double" fixer checks the pair of arcs and makes sure that the labels are correct. For example, if the user chooses to draw both arcs but only includes labels with directions of +X and –Z on one arc, the fixer rule will automatically add labels with directions of –X and +Z on the second arc. Similar fixer rules exist for radial and press-fitted connections. The final rule in this rule set is called the "cascade rule" and provides a way around a common problem which would otherwise underconstrain some components. The cascading rule establishes relationships between pairs of components which are not in contact by looking at their respective relationships with components which they are both in contact with. This is done because in certain situations the removal of a component which is between two other non-touching components can leave the assembly underconstrained. The cascading rule searches for locations in the assembly which have the potential to develop this problem and adds virtual rectangular constraints which will prevent it. The fixer rule set contains a total of five rules.

Rule set #2, the flip rule set, returns to the important fact that the process of recognizing and applying grammar rules is meant to estimate real-life disassembly sequences. The importance of the global variable was touched on earlier and it is a key factor in determining whether or not the flip rule set is used. Depending on the orientation of the assembly (specified by the global variable) certain grammar rules will not be recognized. For example if the +Z surface of the product is defined to be resting on a table then a rule which removes an arc carrying the label thread+Z (which would entail the removal of a screw in the +Z direction) could not be applied. If the simulation recognizes that there are no possible rules left to apply but further rule application would be possible if the assembly were in a different orientation, then the flip rule set will automatically flip the assembly into the desired position. Rather than changing the global variable, the flip grammar rules would change the polarity of all arcs within the graph along the Cartesian axis containing potentially applicable rules. So for this example of a threaded connection, the global variable would remain +Z but the thread+Z label would be changed to thread-Z and would therefore be eligible to be removed. All other labels with a +Z directional suffix would be changed to -Z and likewise -Z would be switched to +Z. There are six rules within the flip rule set, one for each of the possible rotations which could potentially flip the product into a more desirable orientation.

Before explaining the changes made by applying the single rule contained within the module rule set (rule set #3), the concept of a module must first be explained. A module is any component within an assembly which may be modeled as either a single component or as a collection of components. For example, in the leaf blower assemblies depicted in the appendices, the motor may be considered to be either a single component or a collection of all of the components which make up the motor. The reason why this is important is that it may sometimes be desirable to leave the module as one component and not to disassemble it any further. The module rule set gives the user this option. If the user wishes to disassemble the motor then the module rule will be applied otherwise it will not and the disassembly process will be stopped. If a module does exist in an assembly, the nodes representing each of the module components are drawn as well as a node representing the module as a whole. In the appendices, arcs carrying the label of "module" can be seen. These are the only arcs which do not convey information regarding physical connections between components. Instead these arcs connect the single module node with each of its components. The application of the module rule will delete each of these module arcs as well as the module node, indicating that the process of removing the

individual components will begin. Figure 4 below shows a hypothetical motor which consists of four components before (left) and after (right) application of the module rule. Note that the module node and the arcs approximate a hyperarc that spans all the motor component nodes.

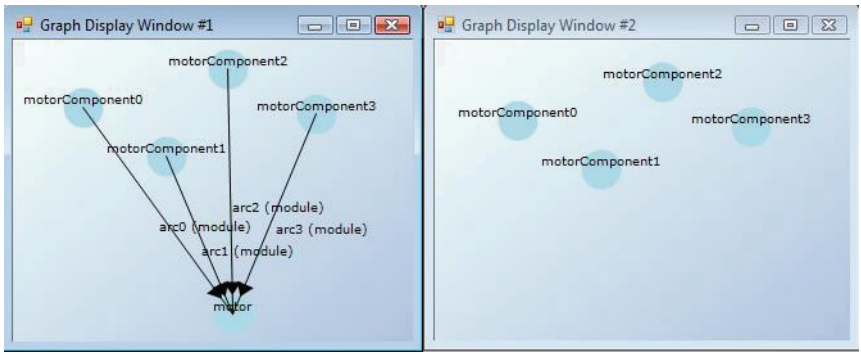


Figure 4. A module before (left) and after (right) module rule application.

The relationships between rule sets are also important since they must be used in a certain order if they are to function correctly. The fixer rule set is used first and will automatically apply rules until none are recognized. At this point the disassembly rule set is used and rules will be applied until no more are recognized. Then the flip rule set is used. If one of the flip grammar rules is recognized and applied, then the process returns to the disassembly rule set, otherwise the process proceeds to the module rule set. If no grammar rules are applied here, the disassembly process is terminated but if the module rule is applied then the process again returns to the disassembly rule set. This sequence of switching from one rule set to another estimates the real-life decisions which are required to either fully or partially disassembly any product.

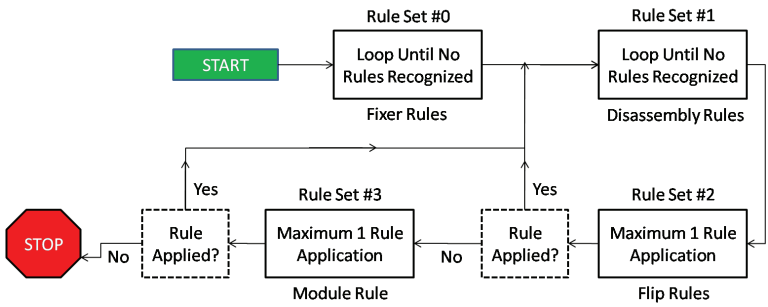


Figure 5. Rule set flowchart.

Now that the process for simulating product disassembly is formulated we would like to use it to attain real information regarding our two important parameters: disassembly time and wasted weight. In order to do this we still need to supply the program with information regarding the components themselves.

4 CALCULATIONS

4.1 Resulting Parameters

The ultimate goal of the computer simulation is to evaluate the process of disassembling a product and therefore the product itself. To reiterate, the two most important parameters which are used in evaluating this process are the disassembly time and the wasted weight. These two parameters are themselves functions of properties of the components within the product (nodes within the graph) and their connections to each other (arcs within the graph).

In a physical disassembly sequence each individual step has a time associated with it depending on a variety of variables. The time required to remove a screw for example is a function of the head type, the type of tool used and the length of threads being disengaged. All of the information required for time calculations is stored within the arcs and nodes in GraphSynth. These variables are stored in a list and each position in the list holds a specific piece on information.

4.2 Required User Inputs

All nodes are capable of holding the same information however not all of it is required for each type of disassembly operation. Some grammar rules, such as those for windings, press-fitted and adhesive operations, require information to be stored in arcs rather than nodes. It may be intuitive to think of these operations as the breaking of a physical bond between two components rather than effortlessly changing the positions of two components with respect to each other. This is why the arcs rather than the nodes hold the most relevant information in these cases.

For both nodes and arcs the lists which hold relevant variables are in a specific order. Within nodes there are ten variable positions. The information in the first two positions should be thought of as mandatory regardless of the disassembly process which will be performed. Position 0 contains the material type. Position 1 is the weight of the component. The material type is required to accurately tell the simulation when to stop while the weight is required to accurately calculate the wasted weight. Positions 2 and 3 are geometric parameters: size and thickness. The definitions of these are specific as defined in "Product Design for Assembly" by Boothroyd and Dewhurst [5] for reasons which will be explained shortly. Positions 4, 5, 6 and 7 are also values defined by Boothroyd and Dewhurst's work and contain information about physically handling the component in question. It is not a strict requirement that all of the positions contain information since default values will be assumed when it is time to perform the disassembly time and wasted weight calculations. However the values for these two parameters become more accurate as the user supplies more information. Positions 8 and 9 contain information that is only relevant to specific threaded components. Position 8 contains a combination of screw head type and hand-driven versus power-tool removal information and position 9 supplies the thread length. Positions 8 and 9 hold all of the information necessary to calculate the removal time of a threaded component [6].

"Product Design for Assembly" provides calculations which are necessary when calculating the time associated with rectangular and radial grammar rules. All of the actions performed in these rule groups involve gripping and moving components from their current position to some place outside of the confines of the assembly. The times associated with these actions are all functions of the weight and geometric dimensions of the components in question.

As was stated previously, only grammar rules in the press fit group and the adhesive rule require certain arcs to contain variables. The adhesive rule expects the adhesive arc which it is removing to contain one variable which states what type of adhesive is being used. The disassembly time required to remove the adhesive should only be a function of this adhesive type. For press-fitted connections, the single variable corresponds to a value taken from Boothroyd and Dewhurst. Winding arcs contain two variables: one is the number of turns while the other is the coil diameter. These are used to determine the unwind time.

5 EVALUATION

In existing work within the field of automated design, the tree of disassembly operations is often called a disassembly tree. The disassembly tree most often terminates at states in which the product is completely disassembled. Since the approach taken here allows for the inclusion of component material data, information which often is not considered, our terminal states do not necessarily contain the fully disassembled product. In realistic disassembly processes, if the goal of disassembly is to organize components for recycling, it does not make sense to separate a group of components if they are composed of the same material. This homogeneous group would likely remain assembled and be recycled as a whole. In this research the tree terminates once the assembly is reduced to a number of materially homogeneous groups, once again emphasizing the importance of disassembly order. Each node in the tree is evaluated in terms of the two primary parameters: disassembly time and wasted weight.

The method of evaluation is a significant strength of this method over similar, previous research. Many methods of representing disassembly trees such as Homem de Mello and Sanderson's AND/OR

hypergraph exhibit a common weakness in their evaluation [7]. Since these methods concentrate on general representation of the tree and not on representation of the assembly, their results include computational time rather than the more realistic and more useful property of disassembly time. The assembly representation in this method captures information about how components fit together, rather than simply how they are positioned relative to one another. Other methods concentrate on this latter group of characteristics and therefore their resultant evaluations take neither disassembly time nor disassembly cost into account.
An iterative-deepening A* (IDA*) [8] approach is used to search the disassembly tree. A number of different tree-search algorithms were investigated but IDA* proved to be the fastest technique which also produces a complete set of Pareto candidates for a given product. This Pareto set presents the completed product, the completely disassembled product as well as various points in between.

6 RESULTS

Example simulations have been run using the aforementioned leaf blower models. Instead of using the entire graphical assemblies, two simplified models have been used. This was done to cut down on the computational time of the simulations. Rather than considering the motor modules to be collections of components, each of the leaf blower motors have been assumed to be singular components. A photo of the components of the simplified Toro assembly is shown in Figure 6 below and its GraphSynth assembly representation in Figure 7 (compare to the full assembly in Appendix A).

Figure 6. Components of the Toro Model #51586 leaf blower.

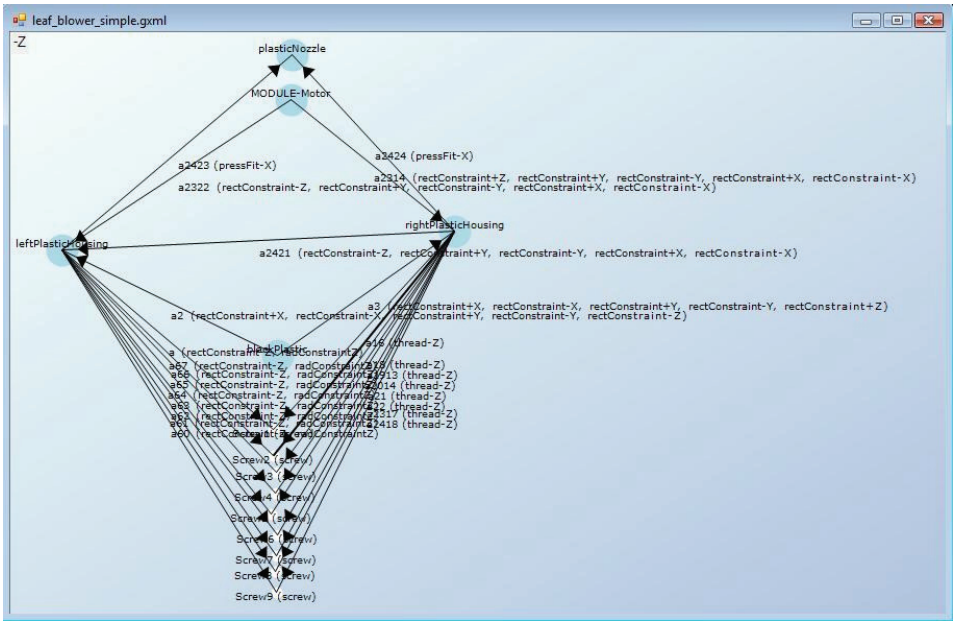


Figure 7. A simplified model of the Toro Model #51586 leaf blower with the motor modeled as a single component.

A comparison of the Pareto sets resulting from the simulations for the two similar products is shown in Figure 8. This graphical depiction allows a direct comparison of the environmental impacts of these similar products. On the plot, the goal is to minimize wasted weight while simultaneously minimizing disassembly time. This means the utopian point is in the lower left corner of the plot.

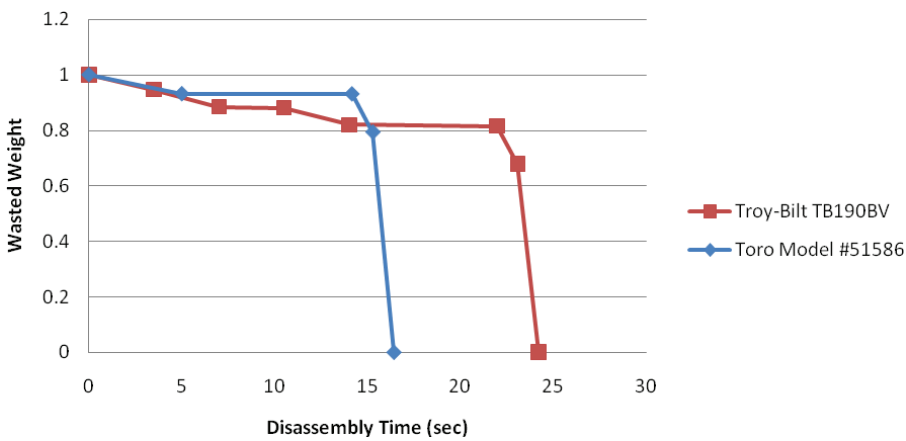


Figure 8. Pareto sets for two leaf blower models.

In addition to the Pareto sets shown in Figure 8 the simulation also outputs the disassembly sequences for each of the points indicated. Direct comparisons can be made between similar products which would suggest that the Toro model is more fit for recycling than the Troy-Bilt model since it takes less time to completely disassemble it.

7 FUTURE WORK & POTENTIAL RECOMMENDATIOS

Future work will involve decreasing the computational time of the tree search and evaluation process. Originally a depth-first search was used, followed by best-first and currently IDA*. The evaluation of nodes in the disassembly tree is a very quick process at the moment therefore work in this area will focus on decreasing the branching factor of the disassembly tree by grouping identical operations – such as removing multiple, identical screws – into a single operation.

More work will also go into analyzing the information conveyed by the Pareto sets. By observing the shapes of the Pareto curves it can be determined which components within a product's assembly have a large impact on the wasted weight to disassembly time relationship. Suggestions such as changing the material type or the method of connection to other components would therefore have an effect on the overall environmental impact of the product. These suggestions could be taken into account in the design phase of product which would increase the usefulness of the evaluation technique presented here.

ACKNOWLEDGMENTS

The authors would like to thank the National Science Foundation for supporting this work under grant award number DMI-044880.

REFERENCES

[1] Gutowski, T., Dahmus, J., Albino, D. and Branham, M. Bayesian, Material Separation Model with Applications to Recycling, in *IEEE International Symposium on Electronics and the Environment,* Orlando, May 2007.

[2] *Extended Producer Responsibility Laws as of May 2009*, http://www.productstewardship.us/displaycommon.cfm?an=1&subarticlenbr=280

[3] Gross, D. and David, E., The Tao of Junk, *Newsweek Web Exclusive*, 5 September 2007, www.newsweek.com/id/39771

[4] *GraphSynth web site,* http://www.me.utexas.edu/~adl/graphsynth/

[5] Boothroyd, G. and Dewhurst, P., *Product Design for Assembly*, 1989 (Wakefield: Boothroyd Dewhurst, Inc.)

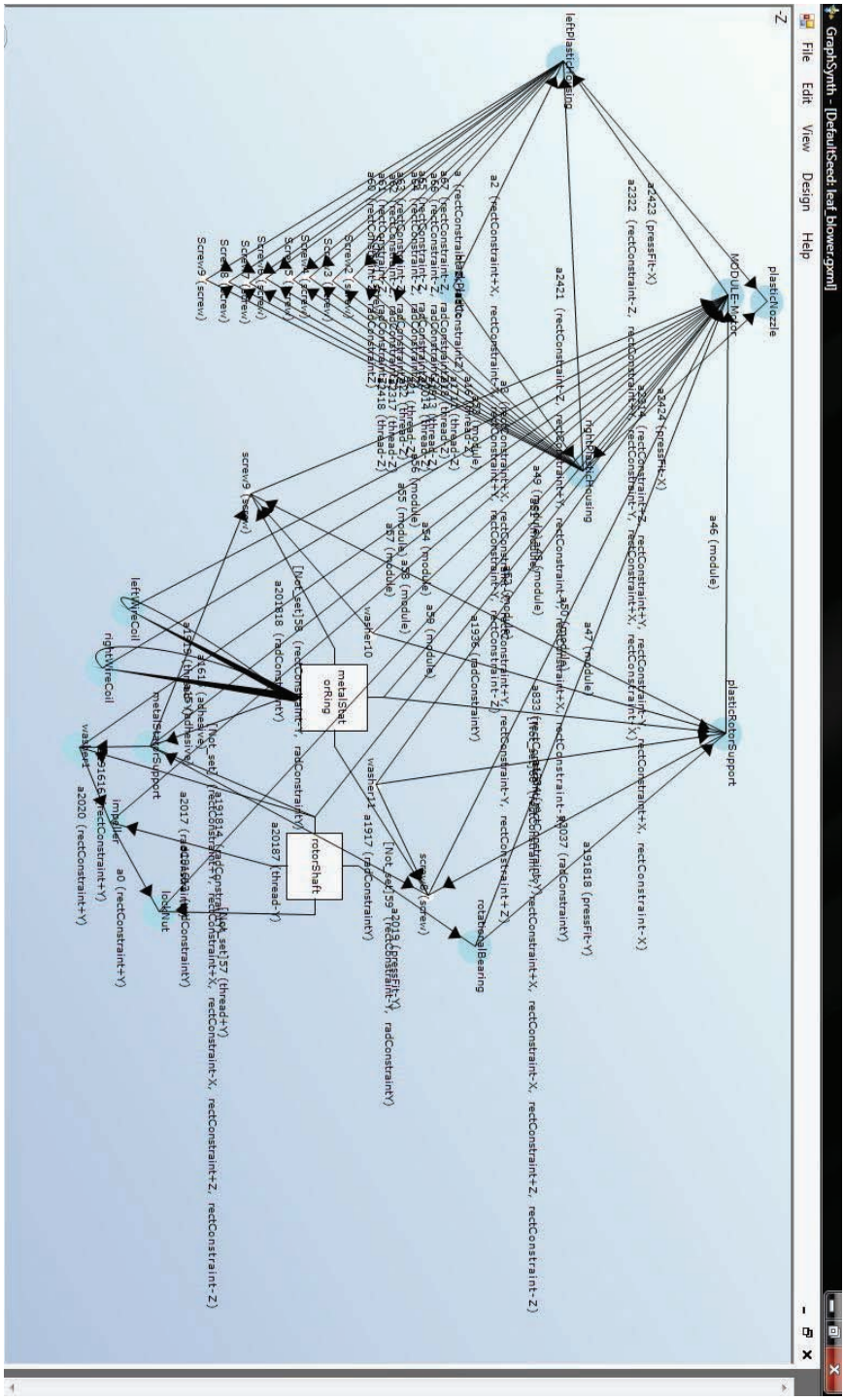
[6] Boothroyd, G., Dewhurst, P. and Knight, W., *Product Design for Manufacture and Assembly*, 2002 (New York: Marcel Dekker)

[7] Homem de Mello, L. S. and Sanderson, A. C., AND/OR Graph Representation of Assembly Plans, in *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 2, April 1990.

[8] Kork, R. E., Depth-First Iterative-Deepening: An Optimal Admissible Tree Search, in *Artificial Intelligence*, No. 27, 1985.

Contact Author: Matthew I. Campbell
The University of Texas at Austin
Department of Mechanical Engineering
1 University Station, C2200
Austin, TX 78712-0292
United States of America
Tel: 512-232-9122
Fax: 512-471-6356
E-mail: mc1@mail.utexas.edu
URL: www.me.utexas.edu/~campbell

APPENDIX A: TORO MODEL #51586



APPENDIX B: TROY-BILT TB190BV