Proceedings of the 2000 IEEE/RSJ
International Conference on
Intelligent Robots and Systems

Reducing the Complexity of Geometric Selective Disassembly

Miguel Angel García[†] Albert Larré[†] Beatriz López[†] Albert Oller[‡]

Intelligent Robotics and Computer Vision Group
[†]Dept. of Computer Science and Mathematics [‡]Dept. of Electrical, Electronic and Automation Engineering
Rovira i Virgili University
Ctra. Salou s/n. 43006 Tarragona, Spain
[magarcia, alarre, blopez, aoller]@etse.urv.es

Abstract

This paper presents an efficient technique for determining a low-cost disassembly sequence suitable to extract a subset of s components from an assembly containing n components, e of which are exterior ($e \ll n$). The most efficient solution to this so-called geometric selective disassembly problem is the wave propagation algorithm, which is reported to have a computational complexity of $O(sn^2)$. Instead, the complexity of the proposed algorithm is $O(en\log n)$ when $s \ll n$, and $O(sn)$ when $s \cong n$. Experimental results with synthetic 3D assemblies are presented.

1 Introduction

Disassembly is becoming an important issue in industrial and mechanical engineering due to the increasing interest of manufacturers in being environmentally conscious [1]. The need for disassembly is appearing in more and more industrial activities, including maintenance, repairing, remanufacturing, recycling and disposal. The general problem of disassembly can be tackled from different perspectives, including the definition of mechanical design strategies that simplify the disassembly of complex products, or the automatic determination of disassembly sequences. This paper focuses on the latter subproblem, which is known in the literature as *disassembly planning*.

At first glance, disassembly planning appears to be highly related to assembly planning, the latter having been extensively studied over the last decades. However, both problems have operational and physical differences that make it convenient to study them separately. Hence, it is not surprising the amount of work that has already been done in the area of disassembly planning [1][2][3][4].

Within disassembly planning, it is possible to distinguish between *complete disassembly* and *selective disassembly* [5]. Complete disassembly involves disassembling all the components of a complex object, and it has been mainly studied as a solution to assembly planning, since the reverse of a disassembly sequence is, in fact, an assembly sequence. Alternatively, selective disassembly is usually more appropriate for de-manufacturing applications, such as maintenance, repairing or recycling. For example, in order to upgrade the graphics card of a personal computer, it is not necessary to dismantle the whole equipment up to the last screw, but only to remove those parts that hinder the access to the card (luckily some screws and the computer cover).

This paper focuses on the problem of *geometric selective disassembly planning*, which consists of determining a minimum sequence of components that must be extracted in order to disassemble *s* selected components from an assembly that contains *n* components [5]. We assume that *e* of the *n* components are exterior and can be directly extracted. In complex assemblies, the number of exterior components is typically a small fraction of the total number of components ($e \ll n$).

The problem of finding optimal or even near-optimal disassembly sequences is known to have an exponential computational complexity [1][2][7]. This complexity can be reduced to a more practical polynomial time through heuristic algorithms that do not guarantee optimal solutions, but feasible, low-cost solutions that, in practice, can be useful enough and which, in many cases, may approximate or even coincide with the optimal solutions. To our knowledge, the best polynomial solution for the selective disassembly problem is the *wave propagation algorithm* [5], which generates low-cost solutions (referred to in [5] as *best known optimal solutions*) in $O(sn^2)$ time. The cost considered in [5] is the number of components removed from the assembly.

This paper presents an algorithm that allows the determination of low-cost selective disassembly sequences in $O(en\log n)$ time when $s \ll n$, and $O(sn)$ time when $s \cong n$. The cost of the sequences is the number of removed components. The algorithm is described in section 2. Section 3 shows experimental results with simple 3D assemblies. Conclusions are finally given in section 4.

2 Selective Disassembly Planning

This section presents an algorithm for determining a low-cost sequence of components suitable for extracting *s* selected components from an assembly of *n* components, from which *e* components are directly accessible from the

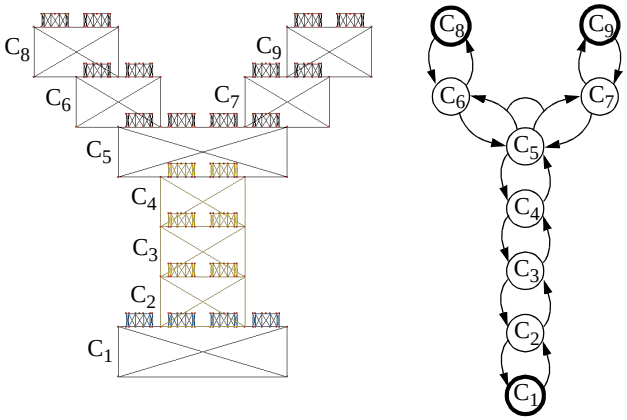

Figure 1. (*left*) Simple assembly with 9 components. (*right*) Corresponding precedence graph.

exterior. The cost of a sequence is considered to be the number of components contained in it.

Before being able to apply any disassembly planner, it is necessary to model the structure of the given assembly. A *precedence graph* [6] is a suitable representation of the relationships among the components of an assembly. Such a graph can be obtained through a procedure similar to the one utilized for constructing *Non-Directional Blocking Graphs* [8], which runs in $O(c^2)$ time, with c being the number of contacts between adjacent assembly components. This graph is created once for every new assembly.

Each node in a precedence graph represents a single component of the assembly. A directed edge from a certain node C_1 to another node C_2 indicates that if C_2 has already been extracted, C_1 can also be removed from the assembly. Several directed edges leaving from the same node express an OR relationship, indicating alternative components that must be extracted before being able to disassemble the component associated with that node. For example, if C_1 has directed edges with C_2 and C_3, C_1 can be removed after either C_2 or C_3 have been extracted.

Alternatively, several directed edges linked by an arc express an AND relationship, indicating that all the destination components must be extracted prior to being able to disassemble the source component. For example, if C_1 has linked directed edges to C_2 and C_3, C_1 can be removed after both C_2 and C_3 have been extracted. Outgoing arcs from exterior nodes would not be necessary from the semantic point of view, but they are required by the first step of the proposed algorithm, as described in section 2.1.

Fig. 1 shows a simple assembly and its precedence graph. This assembly contains 9 components, 3 of which are exterior (C_1, C_8 and C_9). As an example, C_5 has an OR relationship with C_4 and an AND relationship with both C_6 and C_7. This means that C_5 can be disassembled right after extracting either C_4 or both C_6 and C_7.

The precedence graph in Fig. 1 would be simplified if, for example, component C_1 was not exterior—that would occur, for instance, if C_1 was the only contact point between the assembly and its supporting workbench. In that case, all the edges that point downwards in Fig. 1 would disappear.

The proposed disassembly algorithm consists of four stages. The first stage computes the shortest path between each exterior node of the precedence graph and the rest of nodes of the graph. After this stage, each edge in the graph is labeled with the distance to its closest exterior node. The computational complexity of the first stage is $O(en\log n)$.

The second stage determines, for each selected component, shortest paths to its closest exterior components, taking into account the edge labels computed in the first stage. Each shortest path induces a partial disassembly sequence suitable for disassembling that selected component. For each selected component, as many partial disassembly sequences are generated as edges depart from the node associated with that component. As an exception, every group of linked edges (AND relationship) generates a single partial sequence. Assuming that any node of the graph is linked to a small bounded number of other nodes, the computational complexity of this stage is $O(sn)$, and it generates $O(s)$ partial disassembly sequences.

The third stage tries to merge the partial disassembly sequences obtained above in $O(s^2)$ time. Finally, the fourth stage sorts the merged sequences in descending order of selected components. Sequences with the same number of selected components are sorted in ascending order of components. Then, the algorithm extracts successive sequences from the head of the sorted list until all the selected components have appeared in the extracted sequences. This stage can be run in $O(s\log s)$ time.

In the end, the computational complexity of the whole algorithm is $max(O(en\log n), O(sn), O(s^2))$. By assuming that $s \ll n$ and $e \ll n$, the complexity of the algorithm becomes $O(en\log n)$, as determined by the first stage. Taking s to the extreme, $s \cong n$, the complexity would be $O(sn)$, which is also lower than the complexity of the wave propagation algorithm, $O(sn^2)$. The four stages of the proposed algorithm are described below.

2.1 Shortest Paths to Exterior Components

Given a precedence graph with n nodes, e of which are exterior, this stage computes the shortest paths between each exterior node and the other nodes of the graph in $O(n\log n)$ time, by applying *Dijkstra's algorithm* implemented with heap-based priority queues.

The particularity of the application of Dijkstra's algorithm to this problem is that a node C_1 is considered to be connected to a node C_2 when there is a directed edge between C_2 and C_1. The distance between any pair of connected nodes is considered to be unitary.

As an outcome of applying Dijkstra's algorithm, each edge of the precedence graph is labeled with its distance to the starting node, taking the aforementioned particularity into account. For example, Fig. 2 shows the application of Dijkstra's algorithm to the precedence graph shown in Fig. 1, considering exterior nodes C_1, C_8 and C_9 respectively.

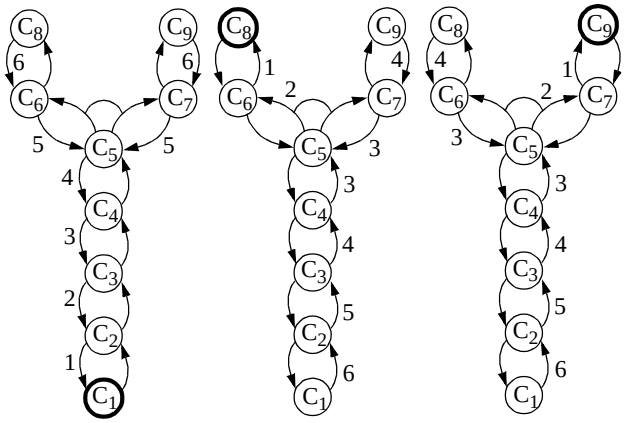

Figure 2. Precedence graph with its edges labeled according to the shortest distance to each exterior node (C_1, C_8 and C_9).

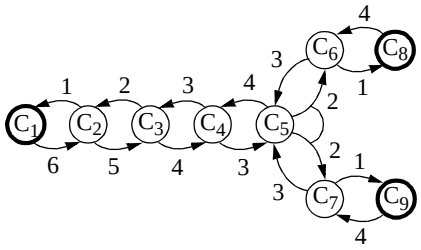

Figure 3. Precedence graph after computing the shortest paths to all the exterior nodes.

Since Dijkstra's algorithm is applied *e* times, any edge may be labeled with *e* different distances. Instead of storing all those values though, each edge is labeled with the smallest distance to its closest exterior node, as it is shown in Fig. 3. Therefore, the total computational complexity of this first stage is $O(en\log n)$.

2.2 Partial Sequence Extraction

Given a precedence graph with each of its edges labeled with the shortest distance to its closest exterior node, the goal of this stage is the generation of partial disassembly sequences for each of the *s* selected components to be disassembled. A *partial disassembly sequence* is a sequence of components whose removal allows the extraction of, at least, one selected component from the assembly. Each selected component induces as many partial sequences as edges depart from its corresponding node in the graph. As an exception, if several outgoing edges belong to the same AND relationship (they are linked), they all contribute to a single partial sequence.

A vector of *n* bits is maintained in order to keep track of the nodes that have already been visited during the process. The partial sequence extraction process starts with the node corresponding to the selected component, and finds out the shortest path to its closest exterior node, using the distances computed after the *e* successive applications of Dijkstra's algorithm.

In order to find out this shortest path, the graph is traversed starting with the selected node. For every node visited during this traversal, its outgoing edge with the smallest label (distance) and whose destination node has not yet been visited is chosen as the next edge to be followed. When the chosen edge belongs to an AND relationship, all the other edges from that relationship are explored in a similar way.

The exception to the previous traversal rule is that an edge is ignored when it belongs to an AND relationship and any of the other edges in that relationship is linked to a node already visited during the current traversal. Without this exception, loops in the partial sequence would be generated. For example, after visiting nodes C_6 and C_5 in Fig. 3, node C_7 cannot be visited, since the edge between C_5 and C_7 belongs to the same AND relationship as the edge between C_5 and C_6, with C_6 having already been visited.

Notice that the aforementioned process can be considered to generate a *minimum spanning tree* rooted at the selected node. The traversal process ends up and, hence, the partial sequence completes when all the branches in that spanning tree contain either exterior nodes or already visited nodes. When the partial sequence is completed, a new sequence will be generated if the selected node still has unexplored outgoing edges.

In order to illustrate the partial sequence extraction process, let us consider that components C_3 and C_6 from the assembly shown in Fig. 1 have been selected for disassembly. The corresponding labeled graph is shown in Fig. 3.

The first partial sequence generated for C_3 would be $\{C_3, C_2, C_1\}$ by following the edges with weights 2 and 1 respectively up to the closest exterior node C_1. A second partial sequence can also be generated from C_3, as it has a second outgoing edge (C_3C_4) and since C_4 has not been visited yet. The new sequence starts with $\{C_3, C_4, C_5, ...\}$. At C_5 two linked edges are found with the same label. Since they belong to an AND relationship, both are expanded, giving rise to the final partial sequence $\{C_3, C_4, C_5, C_6, C_8, C_7, C_9\}$. This sequence is equivalent to $\{C_3, C_4, C_5, C_7, C_9, C_6, C_8\}$.

In order to delimit the various sequential branches contained in a partial sequence and the presence of AND relationships—this information is useful for the final disassembly scheduler, the previous partial disassembly sequences computed for component C_3 are represented as follows: $\{C_3, C_2, C_1, 0\}$ and $\{C_3, C_4, C_5, -C_7, C_6, C_8, 0, C_7, C_9, 0\}$. The negative identifier indicates a branch point in the sequence. This means that the main sequence $\{C_3, C_4, C_5, ...\}$ has two sequential branches whose disassembly can be scheduled in parallel: $\{..., C_6, C_8, 0\}$ and $\{..., C_7, C_9, 0\}$. In general, every time an AND relationship is found during the graph traversal process, before continuing with one of the sequential branches, the negative identifiers corresponding to the initial nodes of the other branches are inserted in the sequence representation.

With respect to the second selected component in the current example, C_6, two more partial sequences would be generated according to the previous rules: $\{C_6, C_8, 0\}$ and

{C_6, C_5, C_4, C_3, C_2, C_1, 0}. Sequence {C_6, C_5, -C_6, C_7, C_9, 0} is discarded as it violates the exception to the traversal rule, leading to a loop.

Assuming that in real assemblies any component is connected to a small bounded number of other components, and considering that, in the worst case, a partial sequence would contain all the components of the assembly, the computational complexity of the partial sequence extraction stage is $O(sn)$ and it generates $O(s)$ sequences.

2.3 Partial Sequence Merging

Given a set of $O(s)$ partial disassembly sequences obtained above, the objective of this stage is to merge smaller partial sequences into bigger ones as much as possible. Two partial sequences will not merge if any of the selected components that appear in one of them also appear in the other sequence. When two sequences merge, the identifiers of the components of the second sequence are appended to those of the first sequence. If the identifier of a non-selected component of the second sequence also appears in the first sequence, this identifier is negated in the merged sequence, indicating a branch point.

The first step of the merging process consists of sorting all the sequences in ascending order of components. Then, the first sequence in the list is tested for merging with the other sequences. In case of merging, that first sequence is removed from the sorted list and the list is resorted. The process continues by choosing the first sequence from the sorted list that has not yet been considered. If all the sequences have already been considered, the merging process concludes. The computational complexity of the merging process is, therefore, $O(s^2)$.

In the example utilized so far, where components C_3 and C_6 were selected for disassembly, the four generated disassembly sequences are sorted by the number of components: {C_6, C_8, 0}, {C_3, C_2, C_1, 0}, {C_6, C_5, C_4, C_3, C_2, C_1, 0}, {C_3, C_4, C_5, -C_7, C_6, C_8, 0, C_7, C_9, 0}.

At the first iteration of the merging process, sequence {C_6, C_8, 0} can only merge with {C_3, C_2, C_1, 0}. This leads to a new sorted list: {C_3, C_2, C_1, 0, C_6, C_8, 0}, {C_6, C_5, C_4, C_3, C_2, C_1, 0}, {C_3, C_4, C_5, -C_7, C_6, C_8, 0, C_7, C_9, 0}. The first new sequence cannot merge with neither the second or third sequences, since the two selected components appear in all three sequences. The second sequence neither can merge with the last one for the same reason. Therefore, the merging process concludes.

2.4 Merged Sequence Sorting and Extraction

The merged sequences obtained above are sorted in descending order of selected components in $O(s\log s)$ time. The sequences with the same number of selected components are sorted in ascending order of components.

The first sequence in the list is then extracted. If this sequence already contains all the selected components, that sequence is the final solution. If the sequence does not contain all the selected components, the remaining sequences in the list are resorted in descending order of pending selected components, and the process is repeated. Each new sequence extracted from the head of the sorted list is merged with the previous solution as indicated in Section 2.3. This iterative process concludes when all the selected components have been found in the extracted sequences. This process can also run in $O(s\log s)$ time.

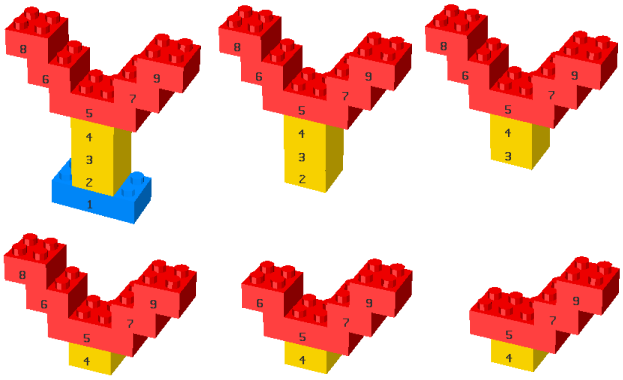

Figure 4. Final disassembly sequence for selected components C_3 and C_6 from the assembly of Fig. 1.

In the current example, the three merged sequences contain the two selected components. Thus, they are sorted in ascending order of components: {C_3, C_2, C_1, 0, C_6, C_8, 0}, {C_6, C_5, C_4, C_3, C_2, C_1, 0}, {C_3, C_4, C_5, -C_7, C_6, C_8, 0, C_7, C_9, 0}. The first merged sequence, {C_3, C_2, C_1, 0, C_6, C_8, 0}, already contains the two selected components C_3 and C_6. Hence, it is the solution to the problem.

3 Experimental Results

The proposed selective disassembly planning algorithm has been implemented in C on Linux and tested upon synthetic assemblies made up of Lego pieces. Legos allow the simple reproduction of many complex assembly topologies corresponding to more realistic industrial assemblies. This section presents some basic examples with the purpose of illustrating the behavior of the algorithm. All the examples were run in a fraction of a second on a Pentium.

A simple scheduler has also been implemented to convert the obtained disassembly sequence into an actual list of components to be sequentially extracted. Firstly, the scheduler reverts the order of the components in the given disassembly sequence, in order to reflect the real disassembly order. For instance, in the example utilized so far, sequence {C_3, C_2, C_1, 0, C_6, C_8, 0} is converted to {C_1, C_2, C_3, 0, C_8, C_6, 0}. Then, the components are listed sequentially: {C_8, C_6, C_1, C_2, C_3}. This is the final disassembly plan, which is shown in Fig. 4. The solution is, in this case, the optimal solution.

Now let us suppose that the selected component is C_5. Two partial sequences are obtained {C_5, -C_7, C_6, C_8, 0, C_7, C_9, 0} and {C_5, C_4, C_3, C_2, C_1, 0} from the precedence graph shown in Fig. 3. These sequences cannot merge and have the same number of components. Therefore, one of them is arbitrarily chosen. In that case, the solution chosen by the program was {C_5, C_4, C_3, C_2, C_1,

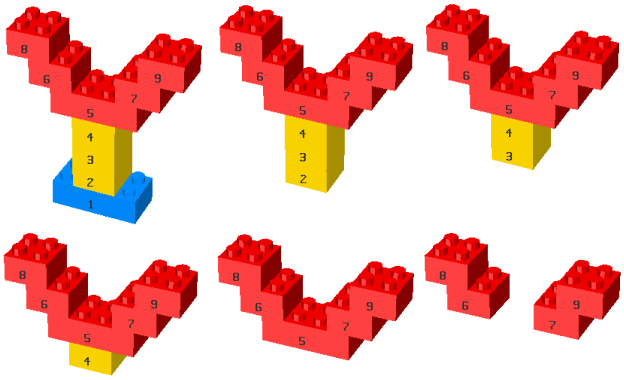

Figure 5. Final disassembly sequence for selected component C_5 from the assembly of Fig. 1.

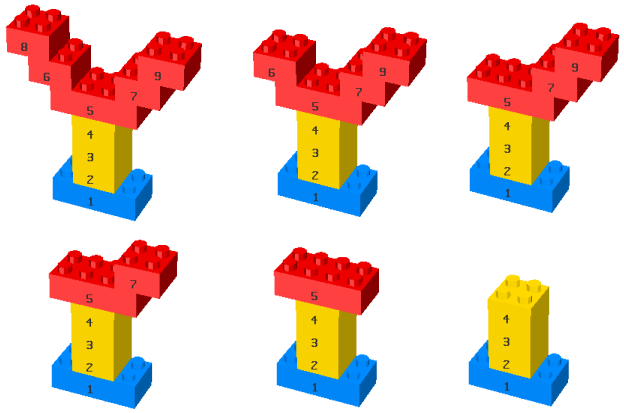

Figure 6. Final disassembly sequence for selected components C_5 and C_6 from the assembly of Fig. 1.

0}, which is transformed by the scheduler into {C_1, C_2, C_3, C_4, C_5}, see Fig. 5.

A new selected component C_6 was then added. In that case 4 partial sequences are generated: {C_5, -C_7, C_6, C_8, 0, C_7, C_9, 0}, {C_5, C_4, C_3, C_2, C_1, 0}, {C_6, C_8, 0} and {C_6, C_5, C_4, C_3, C_2, C_1, 0}. After the merging stage, three sequences come out: {C_5, -C_7, C_6, C_8, 0, C_7, C_9, 0}, {C_5, C_4, C_3, C_2, C_1, 0, C_6, C_8, 0} and {C_6, C_5, C_4, C_3, C_2, C_1, 0}. These sequences contain the two selected components. Therefore, they are only sorted by number of components. The shortest sequence is {C_5, -C_7, C_6, C_8, 0, C_7, C_9, 0} with 5 components. The other two sequences have 6 and 7 components respectively. The scheduler reverts the disassembly sequence into {C_8, C_6, -C_7, C_5, 0, C_9, C_7, 0} and, by taking into account the branch point (negative index), generates the final plan {C_8, C_6, C_9, C_7, C_5}, see Fig. 6, which is an optimal solution to the problem.

In summary, the arbitrary solution found with a single constraint, C_5, is modified when a second constraint, C_6, is added, since, in the latter case, one of the solutions becomes more advantageous than the others, in being able to disassemble both selected components with a minimum number of extractions. This behavior of the proposed algorithm is equivalent to the behavior of the wave propagation algorithm [5].

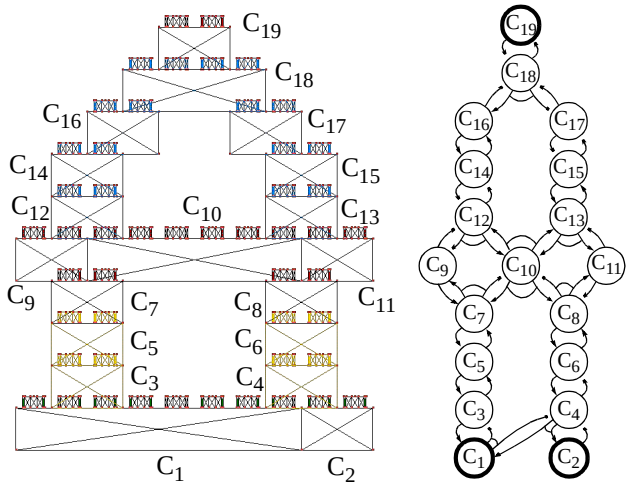

Figure 7. (*left*) Simple assembly with 19 components. (*right*) Corresponding precedence graph.

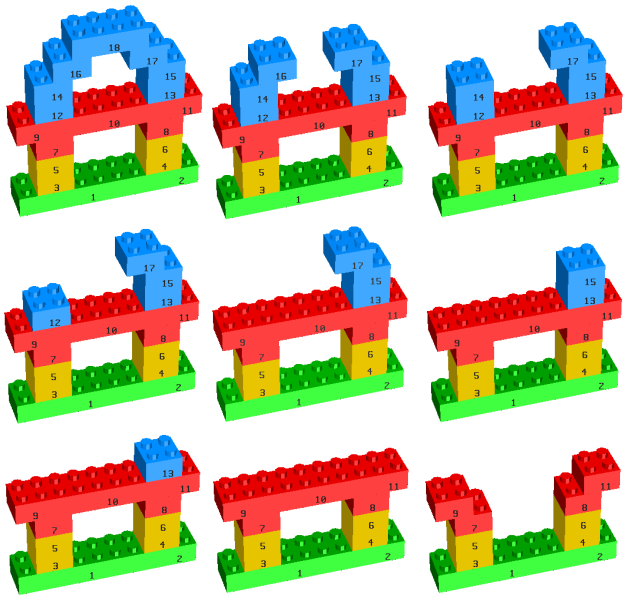

Figure 8. Final disassembly sequence for selected component C_{10} from the assembly of Fig. 7.

Let us now consider another example containing 19 components, 9 AND relationships and 3 exterior components, C_1, C_2 and C_{19} (see Fig. 7). If component C_{10} is only selected for disassembly, two partial disassembly sequences are generated: {C_{10}, -C_8, C_7, C_5, C_3, C_1, 0, C_8, C_6, C_4, -C_1, C_2, 0} and {C_{10}, -C_{13}, C_{12}, C_{14}, C_{16}, C_{18}, C_{19}, 0, C_{13}, C_{15}, C_{17}, 0}. Both sequences are equivalent, since they have 9 components. Thus, the last solution is arbitrarily chosen by the program. With this solution, the scheduler produces the final plan: {C_{19}, C_{18}, C_{16}, C_{14}, C_{12}, C_{17}, C_{15}, C_{13}, C_{10}}, see Fig. 8.

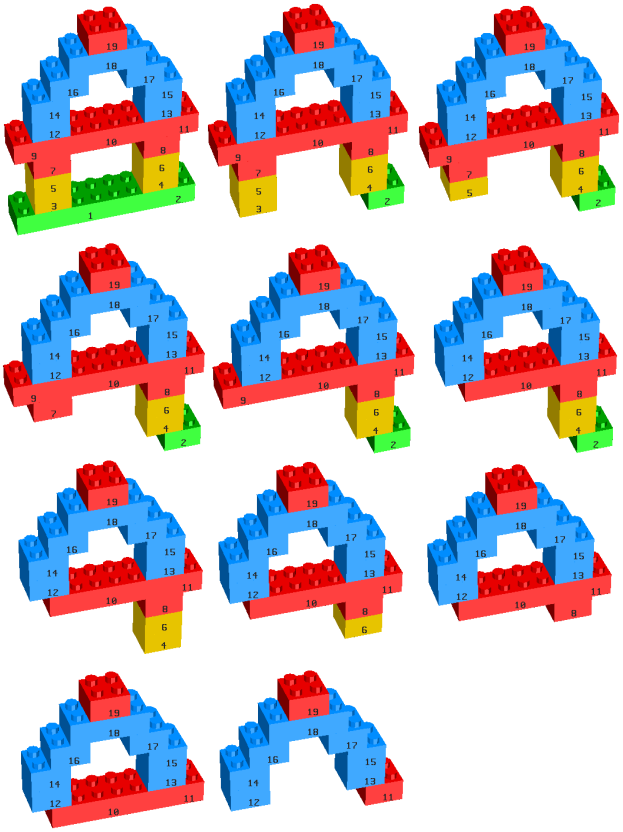

Figure 9. Final disassembly sequence for selected components C_6, C_9 and C_{10} from the assembly of Fig. 7.

Let us now consider that besides C_{10}, components C_6 and C_9 are also selected. In that case, 6 partial sequences are generated: {C_6, C_4, -C_2, C_1, 0, C_2, 0}, {C_6, C_8, -C_{11}, C_{10}, -C_{13}, C_{12}, C_{14}, C_{16}, C_{18}, C_{19}, 0, C_{11}, C_{13}, C_{15}, C_{17}, 0}, {C_9, C_7, C_5, C_3, C_1, 0}, {C_9, C_{12}, C_{14}, C_{16}, C_{18}, C_{19}, 0}, {C_{10}, -C_8, C_7, C_5, C_3, C_1, 0, C_8, C_6, C_4, -C_1, C_2, 0} and {C_{10}, -C_{13}, C_{12}, C_{14}, C_{16}, C_{18}, C_{19}, 0, C_{13}, C_{15}, C_{17}, 0}. The first partial sequence merges with the third, fourth and last ones. The five sequences that remain cannot merge any more and have two selected components. Thus, they are sorted in ascending order of components.

The first sequence in the sorted list is {C_9, C_7, C_5, C_3, C_1, 0, C_6, C_4, -C_2, -C_1, 0, C_2, 0}. Since it only contains components C_6 and C_9, the next sequence in the sorted list that contains the third selected component, {C_{10}, -C_8, C_7, C_5, C_3, C_1, 0, C_8, C_6, C_4, -C_1, C_2, 0}, is extracted and merged with the previous one (Section 2.4), giving rise to the final disassembly sequence: {C_9, C_7, C_5, C_3, C_1, 0, C_6, C_4, -C_2, -C_1, 0, C_2, 0, C_{10}, -C_8, -C_7, -C_5, -C_3, -C_1, 0, C_8, -C_6, -C_4, -C_1, -C_2, 0}, which is converted by the scheduler to the final plan: {C_1, C_3, C_5, C_7, C_9, C_2, C_4, C_6, C_8, C_{10}}, see Fig. 9. Again, this is an optimal solution.

4 Conclusions

An efficient technique for determining low-cost selective disassembly sequences has been presented, with this cost being the number of removed components. Starting with the precedence graph associated with the given assembly, the algorithm computes the minimum distances from the exterior components of the assembly to the rest of components. Then, for each component selected for disassembly, a set of partial disassembly sequences is obtained by finding minimum spanning trees in the precedence graph. If possible, the partial sequences are merged and finally sorted by number of components and selected components. The solution is found by extracting sequences from the head of this sorted list until all the given selected components have been considered.

The best polynomial solution to the geometric selective disassembly problem is, to our knowledge, the *wave propagation algorithm* [5], which has a complexity of $O(sn^2)$, where n is the total number of components in the assembly and s is the number of selected components. The proposed algorithm computes low-cost solutions similar to the ones obtained with the wave propagation algorithm with a lower complexity, $O(en\log n)$ when $s \ll n$ and $O(sn)$ when $s \cong n$, with e being the number of external components in the assembly, $e \ll n$.

The proposed algorithm is being applied upon larger and more realistic assemblies in order to assess the quality of the obtained disassembly sequences. An algorithm for constructing precedence graphs from complex assemblies described through VRML files is also being developed.

5 References

[1] A. Gungor and S. M. Gupta, ”Issues in Environmentally Conscious Manufacturing and Product Recovery: A Survey”. *Computers and Industrial Engineering*, vol.36, no.4, 1999, 811-853.

[2] B. O’Shea, S. S. Grewal and H. Kaebernick, “State of the Art Literature Survey on Disassembly Planning”, *Journal of Concurrent Engineering: Research and Applications*, vol.6, no.4, Dec. 1998, pp. 345-357.

[3] N. Shyamsundar and R. Gadh, "Geometric abstractions to support disassembly analysis", IIE Transactions, vol.31, no.10, 1999, 935-946.

[4] D. Dutta, “Algorithm for multiple disassembly and parallel assemblies”, Transactions of the ASME, vol.117, 1, Feb 1995, 102-109.

[5] H. Srinivasan and R. Gadh, “Complexity Reduction in Geometric Selective Disassembly Using the Wave Propagation Algorithm”, *IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, 1998, 1478-1483.

[6] K. Yokota and D. R. Brough, “Assembly/ Disassembly Sequence Planning”, *Assembly Automation*, vol.2, no.3, pp. 31-38.

[7] M. H. Goldwasser and R. Motwani, “Complexity Measures for Assembly Sequences”, *Int. Journal of Computational Geometry & Applications*, vol.9, no.4 & 5, 1999, 371-417.

[8] R. H. Wilson and J. C. Latombe, “Geometric Reasoning About Mechanical Assembly”, *Artificial Intelligence* 71(2), December 1994, 371-396.