Evolutionary Decomposition for 3D Printing

Eric A. Yu, Jin Yeom, Cem C. Tutum, Etienne Vouga, Risto Miikkulainen
The University of Texas at Austin, Department of Computer Science
Austin, Texas 78712
[yu.eric,jinyeom]@utexas.edu,[tutum,evouga,risto]@cs.utexas.edu

ABSTRACT

Capabilities of extrusion-based 3D-printers have progressed significantly, but complex forms are still challenging to print. One major problem is overhanging surfaces. These surfaces require extra support structure to be printed, wasting material and time. Furthermore, delicate parts of the object can be damaged when these structures are removed. One potential solution is to print the object in parts, but decomposition is difficult. This paper proposes an evolutionary approach for determining optimal object decompositions for 3D printing. Two alternative methods, with different complementary strengths, are tested: Multi-objective Genetic Algorithm (MOGA) and Covariance Matrix Adaptation Evolution Strategy (CMA-ES). MOGA is able to evolve a set of decompositions at variable complexity, i.e. number of pieces, whereas CMA-ES is able to find a limited number of comparable decompositions with significantly less computational time.

CCS CONCEPTS

•Computing methodologies → Continuous space search; Genetic algorithms; Shape analysis;

KEYWORDS

Real parameter Genetic Algorithm, Multiobjective Optimization, CMA-ES, 3D Printing, Decomposition, Support Structure

ACM Reference format:
Eric A. Yu, Jin Yeom, Cem C. Tutum, Etienne Vouga, Risto Miikkulainen. 2017. Evolutionary Decomposition for 3D Printing. In *Proceedings of GECCO '17, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3071178.3071310

1 INTRODUCTION

Additive Manufacturing (AM), commonly known as 3D Printing or Rapid Prototyping, is a flexible fabrication method in which a 3D Computer Aided Design model of a product is digitally sliced into thin layers that are fused on top of each other to form the final design. Fused Deposition Modeling (FDM) is an extrusion-based 3D printing process that is widely adopted because it is simple and inexpensive. However, FDM technology also has significant limitations. Most importantly, the heated and liquefied material has to be laid down in layers with each layer supported by the layers beneath it. Therefore, the maximum slope of overhanging geometry is limited. Often, extra *support structures* composed of the same printing material but used in lower density will be printed alongside the original object to support the overhanging regions with shallower slope (Fig. 1, Left). This results in a waste of material since these support structures are removed and discarded after the printing process. Most significantly, removal of these supports can be time-consuming and frustrating (Fig. 1, Right), especially when the support structures are printed in difficult-to-access, but visible regions of the part. Moreover, some extra material can remain attached to the printed object after the supports are removed. In many cases, additional post-processing surface-treatment operations such as sanding or acetone vapor smoothing need to be applied to remove blemishes caused by this extra material. There is also the risk that delicate parts of the object may break while removing extra support material. Therefore, the amount of support structures used in 3D printing plays an important role in post-processing efforts and the quality of the final print.

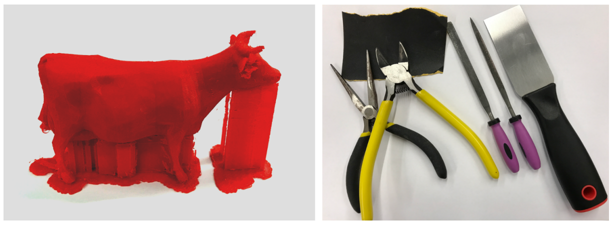
Figure 1: Left: 3D printed cow with support structure. Right: Some of the basic tools used for removing support material including pliers, a spatula, and sandpaper.

There are several approaches to avoid problems associated with the use of support structures in FDM: **(1)** Water soluble filament can be used as the support structures and washed away after the printing process. However, this approach is not widely available as it requires a printer with two extruders that can handle multiple types of filament. **(2)** The printing orientation of an object can be optimized, but this approach can only reduce the use of support material up to a certain level [12]. **(3)** More efficient design of the support structures can reduce the printing time by using less support material [4, 10]. **4)** The topology of the original design can be optimized subject to extra overhang constraints [5, 9]. **5)** The object can be decomposed into smaller printable parts. This paper will focus on the 5th approach, because the other methods are less practical; they either aim to slightly reduce the use of supports or to redesign the 3D object to be printed. Fig. 9 also shows that the proposed methodology can completely remove the need for support structures for most objects.

Several related studies have attempted to decompose an object into printable parts. Luo et al. [8] used object decomposition to cut a large object into parts that fit within a printing volume. Similarly, Chen et al.[1] decomposed an object and packed it to fit into a smaller print volume. However, these studies do not consider the reduction of support structures for 3D printing. Vanek et al. [11] suggested a methodology, that converts 3D solid mesh into a shell by hollowing its inner parts and then dividing the shell into smaller segments, however this method is unsuited for FDM since the resulting arbitrary shell pieces would need support. Hu et al.[7] proposed a bottom-up clustering algorithm for decomposing an object into approximate pyramidal pieces. The idea is that pyramidal pieces can be printed without supports as any point within the object will have material beneath it. However, in practice, 3D printers can print non-pyramidal objects as long as the orientation of the faces fall within the printer's tolerances. In contrast to the pyramidal decomposition work, the proposed methods employ a top-down evolutionary approach to the decomposition problem and does not limit decompositions with pyramidality constraints.

To solve this problem, two alternative evolutionary methods with complementary strengths are proposed: Multi-objective Genetic Algorithm (MOGA) and Covariance Matrix Adaptation Evolution Strategy (CMA-ES)[6]. MOGA is developed to simultaneously evolve solutions with different topologies and is able to find a final set of decompositions at variable complexity, i.e. number of pieces. On the other hand, CMA-ES evolves a population of solutions with identical topologies and is able to find a limited number of comparable decompositions with significantly less computational time.

The paper is organized as follows: First, the decomposition problem is formulated, and the representation of the candidate object decompositions is introduced. Next, the working principles of MOGA and CMA-ES are given. Then, six 3D objects that are frequently used in the computer graphics community for test purposes are decomposed using the both evolutionary methods and the results are compared. Quantitative results are accompanied by the prints of successful decompositions, many of which do not require any support structure. Finally, outcomes of both methodologies are briefly discussed and some ideas for the future work are elaborated.

2 METHODOLOGY

Two separate evolutionary approaches searching for object decompositions that result in the greatest reduction of overhanging area are presented and compared. Because of the correlation between overhanging area and the need for support structures during printing, solutions resulting in large reductions of total overhanging area will also be the solutions that reduce the amount of support structures required to print the object. The first approach uses MOGA to search for an optimal decomposition while our second approach uses CMA-ES. Each of these methods has advantages over the other that can be beneficial in the object decomposition domain. In contrast to CMA-ES, MOGA allows for encodings of candidate solutions to be of variable length. As a result, object decompositions with varying number of cuts can be evolved simultaneously with MOGA. However, CMA-ES is able to converge on optimal solutions with a much smaller population size, greatly decreasing the number of fitness evaluations performed compared to MOGA.

Both approaches are implemented using Distributed Evolutionary Algorithms in Python (DEAP) [2], an evolutionary computation framework for Python which provides functionality for rapid development of Evolutionary Algorithms. Another advantage of DEAP is that its integration with the Scalable COncurrent Operations in Python (SCOOP) module allows for objective evaluations of candidate solutions to be run in parallel.

2.1 Representation of Object Decompositions

Object decompositions for both approaches are represented as a Binary Space Partitioning (BSP) Tree. The BSP tree is a flexible and recursive formulation for the separation of pieces through a series of planar cuts [8]. Each node of the BSP tree contains the information necessary to represent a single cutting plane as a point and a normal vector. The root node represents an initial cutting plane that divides the object into two pieces. Each left child then represents a further division of the resulting piece that is opposite the normal direction of its parent's cutting plane. Each right child represents a further division of the resulting piece that is in the direction of its parent's cutting plane. A diagram depicting one such tree and its corresponding object decomposition is shown in Figure 2.

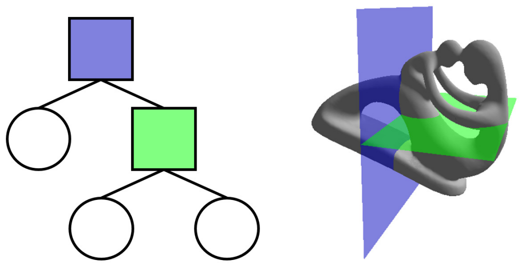
Figure 2: A diagram illustrating object decomposition using a Binary Space Partitioning (BSP) tree representation. Left: Each square node represents a cutting plane through the object while each circle represents a resulting piece. Right: A visualization of the decomposition process is given using the BSP tree on the left with plane colors corresponding to the square nodes.

Each BSP tree is encoded as an array of continuous values between 0 and 1. Each node of the BSP tree is represented by a group of five contiguous values within this array. The first three values represent the location of a point within the bounding box around the object as determined by the minimum and maximum x, y, and z values of the points composing the object. The final two values represent the direction of the cutting plane's normal vector. The first of these values represents a rotation of the normal vector around the x-axis while the second value represents a rotation of the normal vector around the y-axis. Together, these two angles can represent any orientation of the normal vector. To construct the BSP tree from the array representation, the first set of five values within the array represent the root cutting node. Then for each following cut, the point value of the cutting plane is checked to determine which piece it lies within. This comparison determines the location within the tree where the new node should be added. The process repeats until every cutting plane is added to the tree.

2.2 Objective Function

When applied to the original object, the BSP tree divides the object into several smaller parts that make up the original object when pieced together. The total overhanging area of these parts is calculated by choosing the optimum printing orientation for each piece and summing the area of faces with orientation less than 45 degrees with respect to the printing plane. Our optimization algorithms try to find the object decomposition that produces parts that have the smallest total overhanging area.

Evolutionary approaches lend themselves well to multiobjective optimization. While the goal is to find the object decomposition that minimizes overhanging area, breaking the object into hundreds of minuscule pieces is also unacceptable as having many pieces could result in extra work required to put the object together, defeating the goal of reducing the post-processing work. Therefore, the minimization of the number of cuts is considered as a secondary objective. Fig. 3 shows the solutions in the objective space schematically. The Pareto front is drawn in a continuous manner even though the discrete nature of the number of cuts results in columnar distribution.

When working with multiobjective optimization, two important concepts are domination and the diversity preservation. A solution x_1 is said to dominate solution x_2 if for all objective values $f_i(x_1)$ and $f_i(x_2)$ for $i = 0, 1, \dots, m$, $f_i(x_1) \leq f_i(x_2)$, and for at least one i, $f_i(x_1) < f_i(x_2)$. The Pareto Optimal Set is the set of solutions that are not dominated by any other solutions.

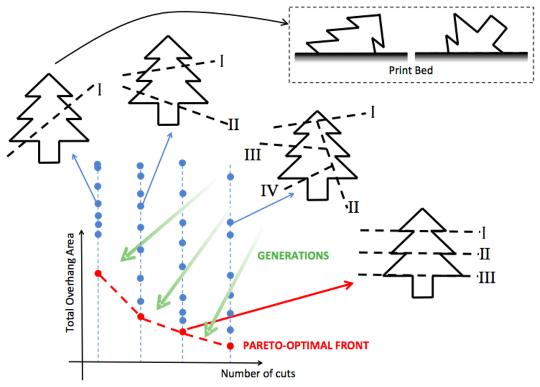

Figure 3: Schematic view of the distribution of different slicing configurations together with the Pareto solutions for the tree object in the objective space.

2.3 Mesh Reduction

Evolutionary Algorithms iterate over multiple generations and the fitness evaluation must be performed on every candidate solution in each generation. This iteration results in a large number of fitness evaluations. The objective function evaluation can be fairly expensive as the execution times of cutting the object, finding the optimal printing plane, and summing the area of overhanging faces all increase as a function of the number of vertices and faces of the object. There are a wide range of algorithms available for reducing the number of vertices and faces in a mesh while maintaining the major structural features of the object. By first simplifying the mesh using a fast mesh simplification program and then performing optimization on the simplified mesh, the time taken for objective evaluation can be greatly reduced. Since the simplified object maintains the basic shape of the original object, the resulting BSP tree solutions found for the simplified mesh will also result in good decompositions of the original object. The process of simplifying the mesh can be done efficiently, taking an insignificant amount of time compared to time taken for fitness evaluation. The reduction of the number of vertices and faces greatly improves the execution time of the objective evaluations in our algorithms. In the experiments, a simplification of the meshes keeping 10% of the verticies of the original mesh is performed using a fast quadratic mesh decimation algorithm. Future work can be done to automatically find the optimal amount of reduction to be used, and feature preserving mesh reduction algorithms can be used to further reduce the resolution of the mesh while preserving important topological features.

2.4 Method I: MOGA

The first method for finding optimal object decompositions relies on MOGA to find a set of solutions minimizing both total overhanging area and number of cuts. During initialization, a population of candidate solutions is created. Each candidate solution is referred to as an individual. Each individual, which is encoded as an array of continuous values between 0 and 1, represents a BSP tree that can be applied to the object to decompose it into smaller pieces. In each iteration of MOGA, there is a selection phase in which the candidate solutions are sorted using that solution's fitness value. The genetic operators of crossover and mutation are then applied to the best candidate solutions in order to form the next generation of candidate solutions. The algorithm is presented in detail below.

Algorithm 1 MOGA Object Decomposition

```
1:  procedure DECOMPOSE($obj$, $pop\_size = 50$)
2:      $pop$ := INITIALIZE_POP($pop\_size$)
3:      $fitnesses$ := EVALUATE_FITNESSES($pop$, $obj$)
4:      for $i$ from 0 to $N$ do
5:          $pop[i].fitness := fitnesses[i]$
6:      $children := \emptyset$
7:      $generation := 0$
8:      while $generation < 50$ do
9:          $pop := pop \cup children$
10:         FAST_NON-DOMINATED_SORT($pop$)
11:         $pop := pop[: pop\_size]$
12:         $children$ := MUTATION_AND_CROSSOVER($pop$)
13:         $fitnesses$ := EVALUATE_FITNESSES($children$)
14:         for $i$ from 0 to LEN($children$) do
15:             $children[i].fitness := fitnesses[i]$
16:         $generation := generation + 1$
```

2.4.1 Selection. Object decompositions are optimized with respect to the objectives of minimizing overhanging area and minimizing the number of cuts used to decompose the object. During the selection phase, the new candidate solutions are first evaluated in parallel to determine their fitness values. Two values are calculated for each object, one for total overhanging area and one for the number of cuts. An elitist approach is chosen, so the newly evaluated candidate solutions (offspring) are added to the already evaluated parent solutions from the previous generation prior to selection. The combined set of candidate solutions is then sorted using the non-dominated sorting based selection algorithm in NSGA-II [3]. In NSGA-II, a crowding distance value is assigned to each solution to determine that solution's distance from its neighboring solutions. This value is used to maintain diversity during selection by preferring solutions that are farther away from their neighbors. However, since the number of cuts is a discrete value, crowding distance does not have much of an effect and can be ignored. The best solutions are chosen to be the parents of the next generation. Crossover and mutation operators are applied to the parents to form new candidate solutions that make up the offspring population.

2.4.2 Crossover. The crossover operation takes two parent solutions as input to create offspring solutions. The intuition is that taking features from two strong parent solutions may produce similarly good offspring solutions. Two crossover operations are employed in this approach. The first type, macro crossover, takes a random set of cuts from both parent solutions and concatenates these cuts to form a new solution. The remaining cuts from both parents are then combined in the same manner to create a second new solution. A diagram illustrating this type of crossover is shown in Fig. 4. The second type, micro crossover, involves selecting a random cut from each parent solution and either swapping the points of the two cutting planes or swapping the angles of the planes. The motivation behind this type of crossover is that combining features of a cutting plane that has a good orientation, but bad position with a cutting plane that has a good position, but bad orientation can result in a better overall cutting plane. A diagram illustrating this type of crossover is shown in Fig. 5.

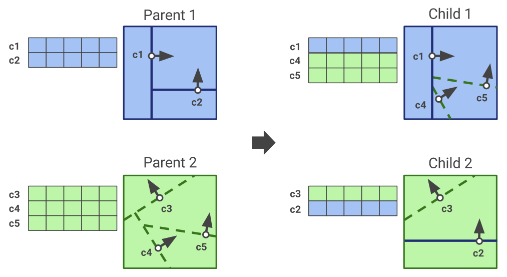

Figure 4: Diagram illustrating macro crossover. This crossover swaps entire cutting planes between two candidate solutions.

2.4.3 Mutation. The mutation operation takes one candidate solution and alters it to create a new candidate solution with similar features. Two different types of mutation are employed. The first type takes a candidate solution and randomly permutes the order that cuts appear in the solution's array representation. Because of the way the solution array is translated to the BSP tree, ordering of the cuts can have a significant influence on the object decomposition produced. The first cut is always considered as the root of the BSP tree and cuts falling earlier in the array tend to lie closer to the root. The second type of mutation adds a small amount of 0-mean Gaussian noise to either the coordinate point of a cutting plane or to the angles that determine the normal vector of the plane. A diagram illustrating the two different mutation operations is shown in Fig. 6.

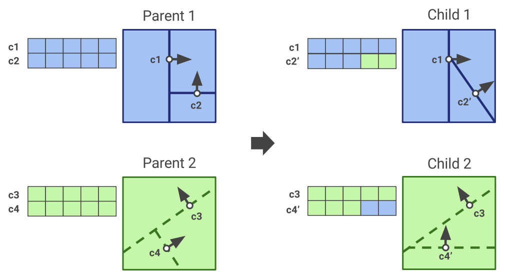

Figure 5: Diagram illustrating micro crossover. This crossover swaps either a cutting plane's location or its orientation with that of another cutting plane

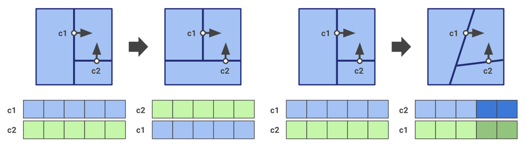

Figure 6: Diagram illustrating mutation. Left: An example of a mutation switching the order of cuts. Right: An example of mutation with a small perturbation of the cutting planes.

2.5 Method II: CMA-ES

The second method uses the CMA-ES algorithm. As opposed to MOGA, which requires a large population of candidate solutions, CMA-ES is able to converge onto a good solution with a much smaller population size. Also, while there are many parameters to tune for MOGA (e.g. crossover rate, mutation rate, ect.) with no single combination of parameters working best across all objects, CMA-ES requires minimal parameter tuning. In CMA-ES, a new population of candidate solutions is created in each iteration by drawing samples from a multivariate Gaussian distribution as determined by the algorithm's internal parameters. Then, the fitness of each candidate solution is evaluated in parallel by applying the objective function. Finally, the calculated fitness values of the candidate solutions are used to update the internal parameters of the algorithm including the covariance matrix and mean which results in a new distribution from which new candidate solutions can be drawn. Iterations continue until convergence or until a maximum number of iterations is reached. Convergence occurs when all solutions have fitness values within a small range or when the fitness of the median solution and fitness of the best solution show no improvement over 20 iterations. The algorithm is shown in detail below.

Algorithm 2 CMA-ES Object Decomposition

```
procedure DECOMPOSE($obj$)
    $num\_cuts := 1$
    $solutions := \emptyset$
    $best\_score := infinity$
    repeat
        $result :=$ CMA-ES_DECOMPOSE($obj, num\_cuts$)
        $solutions$.APPEND($result.fitness$)
        $score := result.fitness$
        if $score < best\_score$ then
            $best\_score := score$
        $num\_cuts := num\_cuts + 1$
    until $score > best\_score$
    return $solutions$
procedure CMA-ES_DECOMPOSE($obj, num\_cuts$)
    $best\_solution := null$
    $best\_fitness := infinity$
    repeat
        $pop :=$ GENERATE()
        $fitnesses :=$ EVALUATE_FITNESSES($pop, obj$)
        sort ZIP($pop, fitnesses$) by $fitnesses$
        if $fitnesses[0] < best\_fitness$ then
            $best\_solution := pop[0]$
            $best\_fitness := fitnesses[0]$
        UPDATE_PARAMETERS()
    until all solutions have similar fitness or
          no improvement in fitness in 20 generations
    return $solution, fitness$
```

Because candidate solutions must have the same size for CMA-ES, the algorithm is applied iteratively, increasing the number of cutting planes after each iteration. The process of increasing the number of cuts and running CMA-ES ends when the best solution found at the end of the current iteration results in pieces with greater total overhanging area than the best solution found in the previous iteration where solutions used one less cut.

3 RESULTS

In this section, results for both MOGA and CMA-ES are presented. The two algorithms are compared using six 3D objects that are frequently used in the computer graphics community. First, both algorithms are executed to find a Pareto set of object decompositions for each object. The overhanging area of the solution pieces is compared to the overhanging area of the original object. Then, to determine the real reduction in support structures used, the original objects are printed alongside select solutions and the amount of support structures is compared. The chosen models exhibit a range of sizes and differ widely in the amount of support structures required during printing.

3.1 Reduction in Overhanging Area

First presented are results relating to the reduction of total overhang area obtained through object decomposition. The total overhanging area of each object prior to decomposition is shown in Table 1. Note that different objects may differ amounts of overhanging area depending on the scale of the object as well as topological differences. Both MOGA and CMA-ES produced results that reduced overhanging area compared to that of the original object.

Object	Total Overhanging Area
Bunny	59.5603
Dolphin	0.4607
Elephant	0.6048
Fertility	0.1542
Homer	0.3138
Horse	0.3151

Table 1: Total area of overhanging faces prior to object decomposition. The overhanging area varies greatly depending on the topology and size of the object.

3.1.1 MOGA Results. MOGA was tested for object decomposition over a variety of crossover and mutation rate parameter combinations. A 4-by-4 grid search was performed to find the combination of crossover and mutation rate which works best for object decomposition. Each initial population consisted of 50 randomly generated candidate solutions each consisting of a BSP tree encoding up to six cutting planes. In each experiment, the Genetic Algorithm was allowed to run for 50 generations. No particular pair of crossover and mutation rates had a clear advantage across all objects. The overhanging areas of the best set of Pareto solutions for each object are shown in Fig. 7. The solutions are very good with even the single cut solutions resulting in over 90% reductions in total overhanging area compared to the original object.

3.1.2 CMA-ES Results. CMA-ES was tested using the same set of objects. Unlike MOGA, which was run for a total of 50 generations, the CMA-ES algorithm was allowed to run until convergence. The overhanging area of the Pareto solutions found by the CMA-ES approach is shown in Fig. 7. Though this approach used a much smaller population size, the quality of solutions obtained using CMA-ES is very similar to the quality of those obtained through the MOGA approach.

3.1.3 Timing Statistics. Execution of both approaches are timed on a machine with an Intel(R) Core(TM) i7-3770 3.40GHz CPU with 12GB of RAM. The timing statistics are shown in Table 2. While it looks like execution time is rather high, the time it takes to decompose an object into printable parts is quite small compared to the many hours it takes to print an object. For the majority of objects, CMA-ES took less time to execute compared to MOGA. Because different object decompositions within a population do not depend on each other, fitness calculation can be done in parallel to significantly reduce execution time.

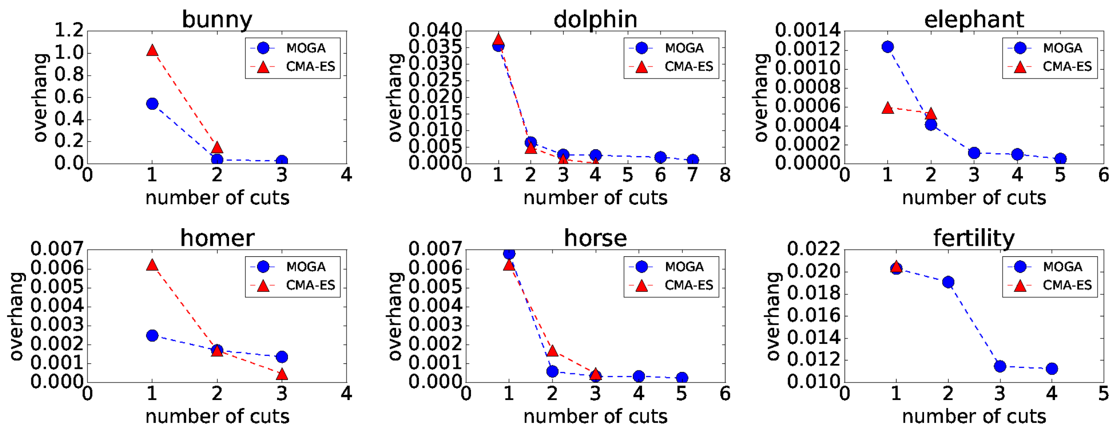

Figure 7: Overhanging area of Pareto solutions found with MOGA and CMA-ES approaches. Both approaches found object decompositions that have significantly less overhanging compared to the original objects. The initial overhanging area for each shape is shown in Table 1.

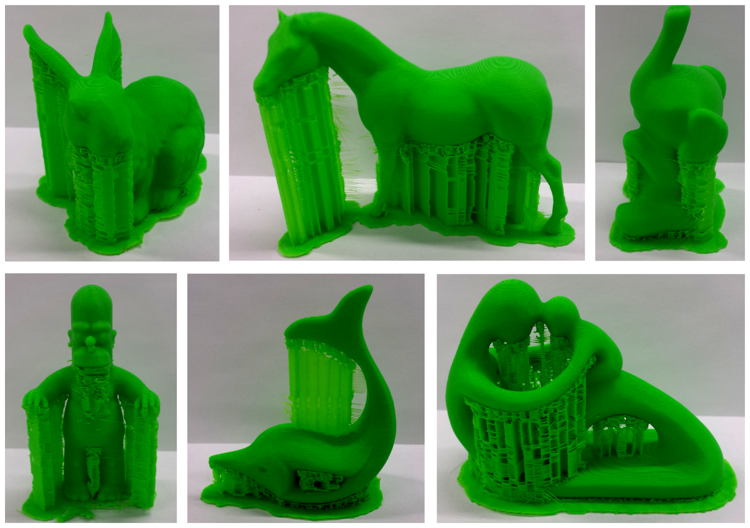

Figure 8: Original objects printed with support structures. Support structures need to be printed under overhanging areas. Supports waste material and are time-consuming to remove.

	Runtime (mins)	
Object	MOGA	CMA-ES
Bunny	7.61	3.90
Dolphin	10.63	11.45
Elephant	17.13	4.30
Fertility	27.30	8.18
Homer	9.68	1.20
Horse	11.22	7.48

Table 2: Timing statistics of object decomposition on an Intel(R) Core(TM) i7-3770 3.40GHz CPU with 12GB of RAM.

3.2 3D Printing Results

To determine the actual reduction of support structures in solutions found with our methods, both the original object as well as select solutions were printed using a Flashforge Creator Pro 3D printer. The original objects printed with support structures is shown in Fig. 8.

3.2.1 Reduction of Support Structures. The solutions produced by the MOGA and CMA-ES approaches greatly reduce the amount of required support material. It is important to note that most printers are able to print very small areas of overhang. This means that sometimes a solution with fewer cuts but with a small amount of overhang can still be printed without requiring support. Having a set of Pareto solutions allows the user to perform qualitative

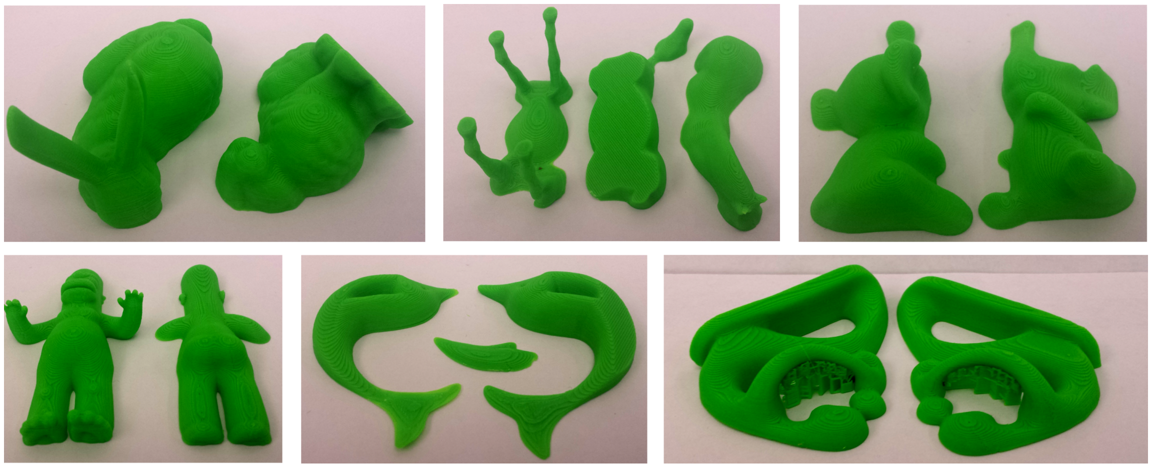

Figure 9: Select printed solution pieces. Homer and Horse solutions were obtained from the MOGA approach while Bunny, Elephant, Fertility, and Dolphin solutions were obtained from the CMA-ES approach. All objects were printed without supports except the fertility figurine, which used much less support than the original.

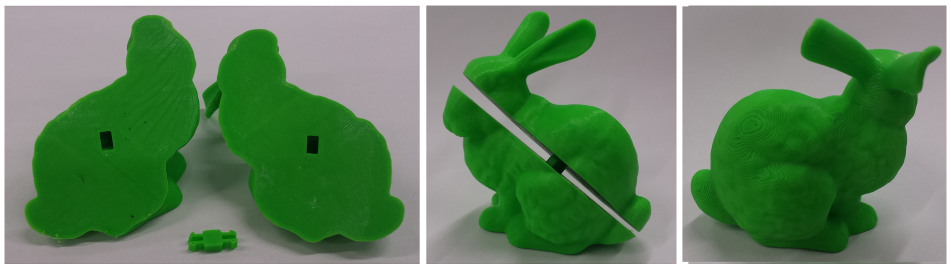

Figure 10: Prints of sliced parts of the bunny model connected with a snapping connector without the need for gluing.

decision-making by choosing the decomposition that they prefer. The results are shown in Fig. 9. Most were printed without any support structure. An estimate of support material reduction is obtained by importing the solution pieces into the Makerware 3D printing software that adds support necessary to print the pieces. The percent reduction of support material used when printing each object is shown in Table 3.

Object	Support Material Reduction(%)
Bunny (CMA-ES, 1 cut)	100
Dolphin (CMA-ES, 1 cut)	100
Elephant (CMA-ES, 1 cut)	100
Fertility (CMA-ES, 1 cut)	93.88
Homer (MOGA, 1 cut)	100
Horse (MOGA, 2 cuts)	100

Table 3: Reduction of support material used for each object when printed on a Flashforge Creator Pro 3D printer.

3.2.2 Assembly of Printed Parts. There are several ways to assemble the printed pieces to recreate the original object. Since the objects are decomposed or sliced along planar surfaces, the original object can be built by putting together the smooth surfaces formed by the cuts. An obvious way of recombining pieces is by gluing them together. Another method is to create snapping type connectors and corresponding grooves in the sliced sub-pieces as shown with the bunny model in Fig. 10.

4 DISCUSSION AND FUTURE WORK

The results show that evolutionary methods can be effective at decomposing objects into printable parts. Both methods produced solutions that could be printed with little if any need for extra support structure. While the MOGA approach is more flexible in its ability to allow candidate solutions with differing numbers of cuts to be evolved at the same time, the CMA-ES approach requires a much smaller population size and thus executes much faster, especially when it finds a good solution involving a small number of cuts. If execution time is of concern, CMA-ES can be used since it uses a smaller population size and performs fewer objective evaluations. Otherwise, if a wide variety of results with varying number of cuts is preferred, MOGA can be used.

Compared to the most similar work [7] in decomposing objects to minimize support structure, which uses a bottom-up clustering algorithm to decompose an object into approximately pyramidal parts, both approaches introduced in this paper can find solutions that are printable without support using fewer pieces, since the proposed methods do not enforce pyramidality assumptions and optimize for reduction of support structures directly.

One advantage of the methods introduced in this paper is that they both provide a collection of solutions that can involve variable number of cuts. This flexibility allows a user to choose the most appropriate solution for a particular purpose. In many cases, there is an inverse relationship between number of cuts and overhanging area of the solutions. However, there are also advantages to using a smaller number of cuts. Solutions with fewer cuts require less effort to piece together as there are fewer resulting pieces. These solutions may also have better structural stability and may be more aesthetically pleasing. The user can opt to print using a solution with fewer cuts at the expense of requiring a small amount of support. Also, as printability can vary from printer to printer, it may be possible that a solution with a smaller number of cuts may be printable without supports even though the amount of calculated overhang is greater than that of a solution involving more cuts.

Because both CMA-ES and MOGA can be easily extended to include more objectives, one area of future work can be to test object decomposition with a variety of other objectives. Some possible additional objectives could be those encouraging decompositions that divide along natural symmetry lines in the object, or objectives discouraging decompositions with cuts through important areas in the object (i.e. context or feature-based slicing). Such objectives could result in more aesthetic decompositions. Other possible objectives could be to maximize the size of the sliced pieces or to maximize the structural integrity of the resulting components.

One major limitation of the proposed approach is that it uses planar cuts that extend completely through the bounding box of the object. In some cases, this limitation can result in multiple pieces being produced with a single cut if the cutting plane passes through a concave section of the object. One area of future work could be to improve the representations of object decompositions to overcome this limitation.

The CMA-ES approach makes an assumption that the process of increasing the number of cuts should stop when the result found by CMA-ES has more overhanging area compared to a solution found involving fewer cuts. This assumption may not be entirely correct as it is possible that allowing more cuts could allow for better solutions to be found. One likely scenario is that a local optimum is found that has much more overhanging area compared to the global optimum. This issue may be addressed in future work by employing random restarts when this situation is encountered.

Neither MOGA nor CMA-ES were able to find the optimal one-cut solution for the dolphin object. This inability to find the optimal solution is likely due to the non-linearity in the fitness landscape around the optimal cut. Cuts parallel to the tail, but not passing through it, will result in pieces that have a large sum of overhanging area, but a cut passing through the tail will result in pieces that have almost none. Evolutionary algorithms can have difficulty finding the optimum solution when this solution is completely surrounded by suboptimal solutions. Running the algorithms with larger population sizes or employing random restarts may allow these difficult solutions to be found.

5 CONCLUSION

In this paper, two evolutionary approaches are introduced to address the problem of decomposition of 3D objects to minimize the use of support structures for FDM 3D printing. Both MOGA and CMA-ES show great promise in finding object decompositions that reduce the amount of overhanging area in the resulting pieces. While MOGA provides flexibility by allowing solutions of varying length to be evolved simultaneously and results in solutions that have varying number of cuts, CMA-ES allows for solutions to be found with fewer objective function evaluations and is much faster for finding object decompositions that use a small number of cuts. The results of several object decompositions are printed and show significant reductions in the amount of support material used. The methods presented in this paper provide a first step towards using evolutionary methods that decompose objects for 3D printing. This work can be easily extended to incorporate a variety of new objectives such as structural stability, aesthetics, and symmetry preservation.

6 ACKNOWLEDGMENTS

The authors wish to acknowledge the funding and support provided by the Freshman Research Initiative Program at the College of Natural Sciences in The University of Texas at Austin as well as BEACON Research Center at Michigan State University.

REFERENCES

[1] Xuelin Chen, Hao Zhang, Jinjie Lin, Ruizhen Hu, Lin Lu, Qixing Huang, Bedrich Benes, Daniel Cohen-Or, and Baoquan Chen. 2015. Dapper: Decompose-and-pack for 3D Printing. *ACM Trans. Graph.* 34, 6, Article 213 (Oct. 2015), 12 pages. DOI: http://dx.doi.org/10.1145/2816795.2818087

[2] François-Michel De Rainville, Félix-Antoine Fortin, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: A Python Framework for Evolutionary Algorithms. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '12)*. ACM, New York, NY, USA, 85–92. DOI: http://dx.doi.org/10.1145/2330784.2330799

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (Apr 2002), 182–197. DOI: http://dx.doi.org/10.1109/4235.996017

[4] Jérémie Dumas, Jean Hergel, and Sylvain Lefebvre. 2014. Bridging the Gap: Automated Steady Scaffoldings for 3D Printing. *ACM Trans. Graph.* 33, 4, Article 98 (July 2014), 10 pages. DOI: http://dx.doi.org/10.1145/2601097.2601153

[5] Andrew T. Gaynor and James K. Guest. 2016. Topology optimization considering overhang constraints: Eliminating sacrificial support material in additive manufacturing through design. *Structural and Multidisciplinary Optimization* 54, 5 (2016), 1157–1172. DOI: http://dx.doi.org/10.1007/s00158-016-1551-x

[6] N. Hansen and A. Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9, 2 (2001), 159–195.

[7] Ruizhen Hu, Honghua Li, Hao Zhang, and Daniel Cohen-Or. 2014. Approximate Pyramidal Shape Decomposition. *ACM Trans. Graph.* 33, 6, Article 213 (Nov. 2014), 12 pages. DOI: http://dx.doi.org/10.1145/2661229.2661244

[8] Linjie Luo, Ilya Baran, Szymon Rusinkiewicz, and Wojciech Matusik. 2012. Chopper: Partitioning Models into 3D-printable Parts. *ACM Trans. Graph.* 31, 6, Article 129 (Nov. 2012), 9 pages. DOI: http://dx.doi.org/10.1145/2366145.2366148

[9] Amir M. Mirzendehdel and Krishnan Suresh. 2016. Support structure constrained topology optimization for additive manufacturing. *Computer-Aided Design* 81 (2016), 1 – 13. DOI: http://dx.doi.org/10.1016/j.cad.2016.08.006

[10] Juraj Vanek, Jorge A. G. Galicia, and Bedrich Benes. 2014. Clever Support: Efficient Support Structure Generation for Digital Fabrication. *Computer Graphics Forum* (2014). DOI: http://dx.doi.org/10.1111/cgf.12437

[11] J. Vanek, J. A. Garcia Galicia, B. Benes, R. Mech, N. Carr, O. Stava, and G. S. Miller. 2012. PackMerger: A 3D Print Volume Optimizer. *Computer Graphics Forum* (2012). DOI: http://dx.doi.org/10.1111/cgf.12353

[12] Marijn P Zwier and Wessel W Wits. 2016. Design for Additive Manufacturing: Automated Build Orientation Selection and Optimization. *Procedia CIRP* 55 (2016), 128–133.