Optimally orient and position multiple solid objects for batch production in 3D printing

Jun Zeng, Ana Patricia Del Angel and Gary Dispoto; Hewlett-Packard Laboratories; Palo Alto, CA

Abstract

3D printing or layered manufacturing is a computer-aided manufacturing process that fabricates parts through layer-wise deposition of material(s). Even though its material deposition techniques exploit wide spectrum of process physics including fused model deposition, stereolithography, selective laser sintering, laminated object manufacturing and inkjet print, its process planning are largely universal: similar workflow steps (e.g., build tray stacking, slicing, tool-path/dot-map generation) and similar objectives (e.g., fast build time, high accuracy, reduced material consumption). The configurations of the print step encapsulate the heterogeneity of the process physics and the associative assistive procedures (e.g., compensation, support).

This paper describes our ongoing research into developing an optimization solution to assist process planning to meet the multiple, oftentimes competing objectives. In particular, this paper focuses on batching multiple objects for simultaneous production within the same build tray. Even though throughout this paper we use Fused Deposition Modeling (FDM, [1]) as example to illustrate 3D printing production, our solution is printing technology agnostic and generally applicable to other printing technologies.

Keywords

Layered manufacturing, batch production, genetic algorithm, parallel programming

Problem statement

Since the development of stereolithography in 1986 [2], the layer-by-layer build technique used in rapid prototyping has also proven to be useful for manufacturing. As a computer-controlled manufacturing process, layered manufacturing enables efficient and effective production of complex parts – complex geometry that may also be coupled with complex heterogeneous material distribution. Additionally, the setup cost of layered manufacturing is low; this makes fabricating short run production even parts of single copy to be economically viable.

What prevents layered manufacturing from even broader adoptions includes production speed, addressable material, final part quality (mechanical, geometrical, etc.), ease of use and production cost. This is a systems engineering effort requiring multi-disciplinary approach across process physics, material, compute, and more. This paper describes one aspect of this systems engineering undertaking: batch fabrication. It principally addresses the reduction of the production time and potential reduction of support material.

We describe a runtime software solution that provides dynamic batching recommendation accounting for the uniqueness of individual printing processing technology and different business objective. This solution optimally selects, places and orients parts to be produced simultaneously through which one or more user-specified objectives can be optimized, for instance, raise throughput, raise quality of service, and/or raise diversity of product offering. This solution can be applied to manage the trade-off between the desire for high level of the granularity of the produced objects (driven by the packing efficiency) and the cost of assembly. Optionally, our method can integrate this optimization into the business-level optimization.

Even though throughout this paper we use Fused Deposition Modeling (FDM, [1]) as example to illustrate 3D printing production, our solution is printing technology agnostic and generally applicable to other printing technologies.

Our solution

Figure 1 shows the system architecture of our solution. It includes three foundational blocks: 1) a genetic-algorithm based Optimizer (in Java) that systematically sweeps through the design space and finds a (pareto-)optimal process plan based on performance scores; 2) an Evaluator (in C++) that generates machine instruction (e.g., g-codes) based on a given process plan and analyzes it to produce performance metrics (e.g., build time, cost); and a browser-based Visual Debugger (in Javascript) that allows both quantitative and visual inspection of the performance. These blocks are described below.

Optimizer. Written in Java, this code uses the genetic algorithm [3] to automate the search for a near-optimal solution over a very large design space. A *codec* module encodes a complete design solution (e.g., valid placement of each object for all objects to be batch-processed together) into a chromosome, a list of floating points.

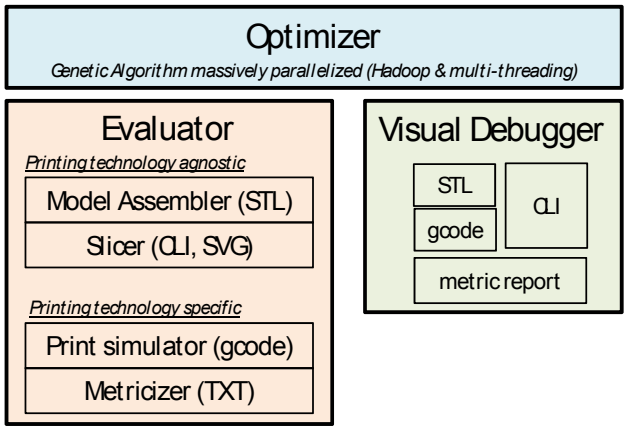

Figure 1. Solution architecture. It includes three foundational blocks: an Optimizer in Java, an Evaluator in C++, and a browser-based Visual Debugger in Javascript.

This *codec* module is also responsible to translate a chromosome into a design solution. It updates Evaluator's configuration file (in XML) with this design solution. The Optimizer uses this XML file to instantiate an Evaluator instance and drives this instance to evaluate the quality of this design solution using a set of performance metrics.

A fitness score is computed based on these performance metrics to quantify the quality of this chromosome based on the design objective. This code also includes a set of evolutionary operators, e.g., cross-over, elite selection, mutation and random reproduction. The Optimizer uses these evolutionary operators to generate population of next generation guided by the rule of "survival the fittest".

When the best of current generation meets the design objective criteria, the evolution of the population terminates. Additionally, if the compute time reaches the allowance, the evolution terminates and the code outputs current best solutions. This is particularly useful for on-line deployments because it guarantees always having a valid solution for the printer to act on.

Evaluator. Written in C++, an Evaluator is instantiated based on a configuration XML file created by the Optimizer. As illustrated in Figure 1, the first half of the data pipeline is printing technology agnostic largely dealing with geometrical operations. The second half of the data pipeline depends on the choice of the printing technology, equipment and the application. Its major components are explained below.

Model Assembler. Upon receiving list of objects and placements (including orientation) for each object, this module assembles all the objects into a single STL file following the placement instruction.

Slicer. This module receives the aforementioned STL, generates a set of two dimensional layers based on the layer thickness prescribed in the configuration XML. It outputs the layers in the form of Common Layer Interface (CLI, [4]) and Scalable Vector Graphics (SVG, [5]). Optimization is applied to rid the node redundancy in STL; topological information is cached for fast compute.

Print Simulator. This module simulates the material deposition process.

Almost all the printing technology requires additional assistive efforts to facilitate the production and/or manage the production quality. These assistive efforts highly depend on the

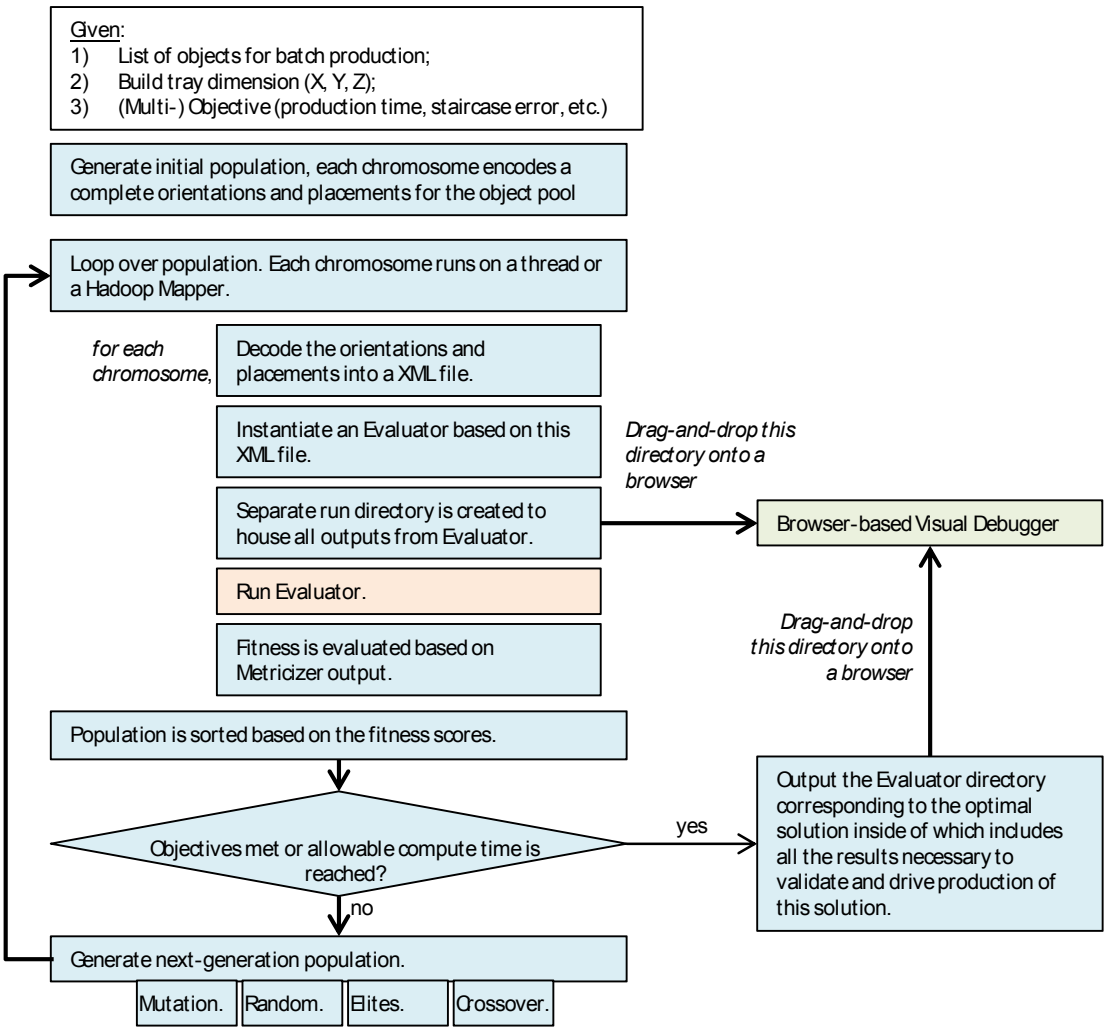

Figure 2. The compute workflow.

printing technology. For instance, in FDM, these assistive efforts include creating structures to support overhanging features against gravity. Other assistive efforts may aim to enhance higher geometrical fidelity, appearances, structural integrity, and more. Both production procedures (build) and these assistive procedures (support) need to be accounted for when simulating the printing process.

The simulator in its current form generates the necessary support information, and then simulates the production based on modeling the kinetics of the servo motor controls – the movement of the build platform, the movement of extrusion heads, and so forth. It outputs the estimates of the production time and material cost.

We do not yet simulate realistic shape, for instance, with a path profile. We assume each layer is replicated by the printing technology ideally with straight wall. Based on this we estimate the volume error and staircase effect. One of the future works can be integrating realistic profiles of the manufacturing process. Another future extension is to simulate the structural performance, for instance, integrating finite element analysis (FEA), so that we can provide predictions on part's structural performance.

Metricizer. A set of predefined quantities are used to measure different aspects of the production performance. Currently we have implemented production time, material consumption (both build material and possible support material), foot-print, build height, build volume, geometrical error, and staircase error. These quantities are evaluated based on integrating the simulation results. A fitness score of a particular design is quantified based on these metrics.

Visual Debugger. Written in JavaScript leveraging libraries such as three.js and two.js and managed by WampServer, this browser-based debugger visualizes entire set of simulation results (Figure 1). The visualization is triggered by drag-and-drop the entire simulation result folder onto the browser canvas. Different pieces of the canvas will update itself accordingly, including:

- STL Viewer allowing interactive visualization of multiple input objects;
- Gcode Viewer allowing interactive, close examination of both tool paths of the extrusion heads (both build and support);
- CLI viewer allowing visualization of layers; and
- Metric Report displaying key performance metrics in addition to production time and material consumption on per layer basis.

Figure 2 describes the compute workflow. The inputs include the build tray dimension, and list of objectives we want to achieve, for instance, minimizing production time, maximizing production quality. These are encoded in a XML document to be read by the Optimizer. Additional input is list of objects to be batch-fabricated together in the form of STL.

Based on the inputs an initial population is generated. Each element of the population, a chromosome, represents one valid design solution. Chromosomes are evaluated in parallel; each chromosome evaluation runs on a single thread.

On each thread the Optimizer first decodes the chromosome into placement instructions for all objects and then writes it into a XML file with which it then instantiates an Evaluator object and starts simulating the print production. Upon completion of the simulation, a fitness score is generated for this chromosome based on the metric produced by the Evaluator.

The Optimizer then sorts the entire population based on the fitness. If the termination condition is met, it outputs the fittest chromosome as the design recommendation. The termination condition may be the objective threshold (e.g., compared to the fitness score), or may be the maximum allowable compute time (e.g., for online applications).

If the termination condition is not met, a new population (next generation) is produced based on this population using a set of

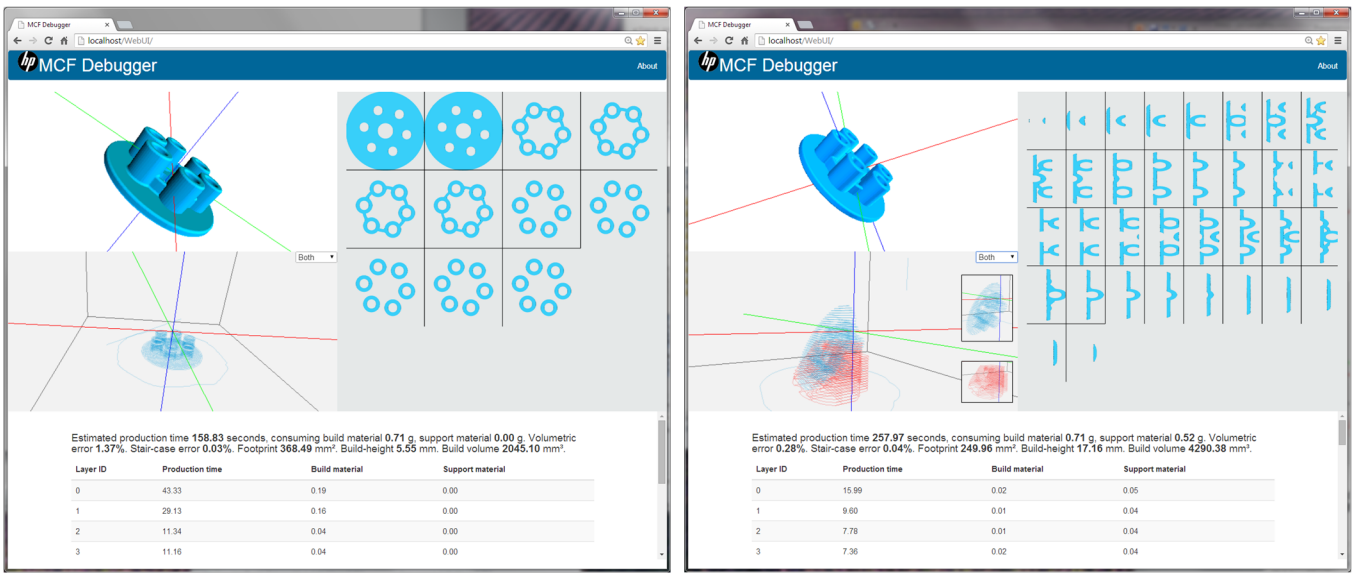

Figure 3. Screenshots of the Visual Debugger. At left shows a part optimally placed for printing. There is no support material required. At right shows this part placed in the build tray with a random orientation. This results not only requirement of support material (layers colored in red) but also longer build time because the number of required layers is increased from *11* to *34*.

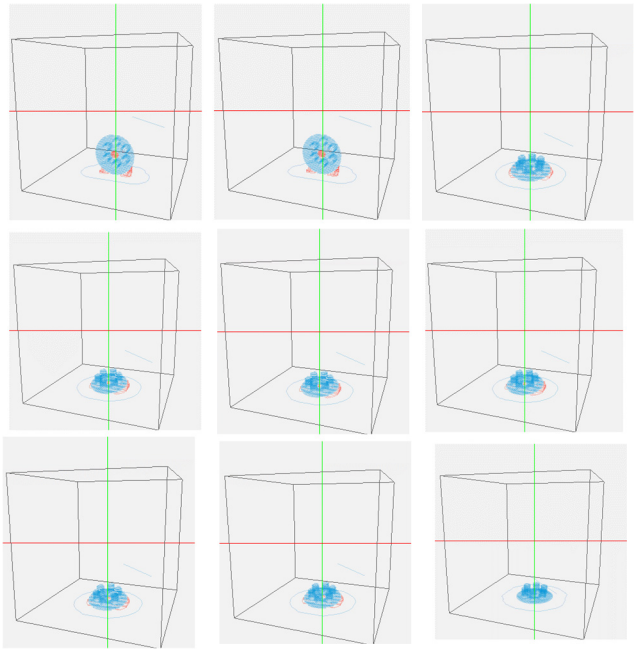

Figure 4. Best orientations of consecutive generations starting with a randomly generated initial result and ending with optimal build orientation.

evolutionary operators (e.g., mutation, elite selection, cross-over, random generation). This new population will be then evaluated for their fitness.

Evidence the solution works

Figure 3 illustrates a problem that requires optimally orient a part. A randomly oriented part (shown at right of Figure 3) not only requires support material for overhanging structure but also requires longer build time due to larger build height (almost *3* times of that of the optimal orientation, shown at left of Figure 3). Using our code, starting from a randomly generated population of size of *8*, it took *8* generations (shown in Figure 4) to converge to an optimal solution costing *35.8* seconds of CPU time (Intel Xeon *3.33GHZ*). As shown in Figure 3, the build time is cut by more than *60%*.

Figures 5-7 illustrate a problem that requires optimally place *8* parts into the same build tray for batch production. Figure 5 shows the convergence of the fitness of the best candidate solution of each generation. Also plotted are the median and worst fitness within each generation as reference. It shows that the best fitness improves monotonically over generations while the worst fitness remains roughly the same over all generations indicating the role of the exploitation operators. Additional convergence acceleration techniques that we have experimented with other optimization problems can be integrated here for faster compute. Figure 6 shows the best placement solutions for several generations to illustrate the continuous improvements. Visually we observe the reduction of the use of support material and the overall build footprint. Figure 7 confirms this visual observation with quantitative data.

Current status

The code as described above has been completed, tested and fully functional, as evidenced by previous section.

Next steps

Multiple directions will be pursued to expand this research, listed below.

1. Parallelization of the *Evaluator* module with GPU to exploit yet another dimension of massive parallelization.
2. Simulation of the 3D printing. Current *print simulator* module limits to kinetic simulation based on prediction of movements of servo-motors. It can be expanded to simulate physical shapes based on path profile. It can be expanded to simulate the part structural properties integrating FEM tools. It will be expanded to cover additional 3D printing technologies.
3. Additional process optimization applications such as adaptive slicing.
4. The application into heterogeneous objects (multi-material objects, functional-grading material objects and objects made by digitally engineered materials) which is the direction 3D printing inevitably progresses towards. The use-case of this code at this moment assumes single-material parts.

References

[1] http://en.wikipedia.org/wiki/Fused_deposition_modeling

[2] Hull, C. W., Apparatus for Production of Three-Dimensional Objects by Stereolithography, U.S. Patent No. 4575330, 1986.

[3] Srinivas, M. and Patnaik, L. M., 1994, "Genetic algorithms: a survey", *IEEE Comput.* 27, 17-26.

[4] http://www.forwiss.uni-passau.de/~welisch/papers/cli_format.html

[5] http://en.wikipedia.org/wiki/Scalable_Vector_Graphics

Author Biography

Jun Zeng is a principal researcher with Hewlett-Packard Laboratories and leads the Software-Defined Production research within Print Production Automation Lab. His current research interest includes distributed systems, machine learning and 3-D printing. Jun has dual advanced degrees in Computer Science and Mechanical Engineering from Johns Hopkins University. His publication includes 50+ peer-reviewed papers and a co-edited book on computer aided design of integrated systems. He was a guest editor of IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. He also serves as termed faculty and member of PhD dissertation committee with Duke University's Electrical and Computer Engineering Department.

Ana Patricia Del Angel is a core member of the Software-Defined Production research team within Print Production Automation Lab, Hewlett-Packard Laboratories. Her current research focus is optimization of 3D printing data pipeline. Her technical expertise also includes mobile development. Ana received a B.S. degree in Mechatronics Engineering from Anahuac University (Altamira, Mexico). She was selected to participate in an international exchange program with University of Kristianstad (Kristianstad, Sweden) where she completed a year-long Computer Science program.

Gary Dispoto is the director of the Production Automation Team at HP Labs. His team designs and optimizes the short run production processes for consumer, enterprise, industrial and commercial applications. Gary has been involved in various aspects of digital print R&D since joining HP Labs in 1985, including extensive work in the area of digital color reproduction. In the past decade, his research interests have expanded to end-to-end print production and short run manufacturing. Gary received B.S. and M.S. degrees in electrical engineering from Stanford University and an MBA degree from the University of Santa Clara.

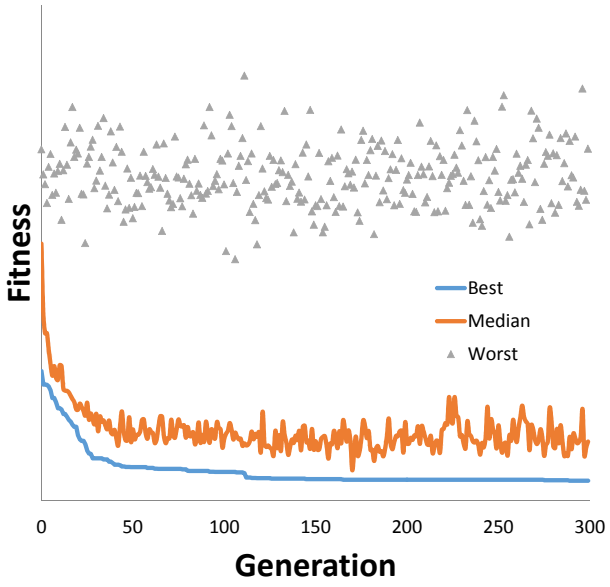

Figure 5. Multi-part placement problem. The fitness of the solution converges monotonically.

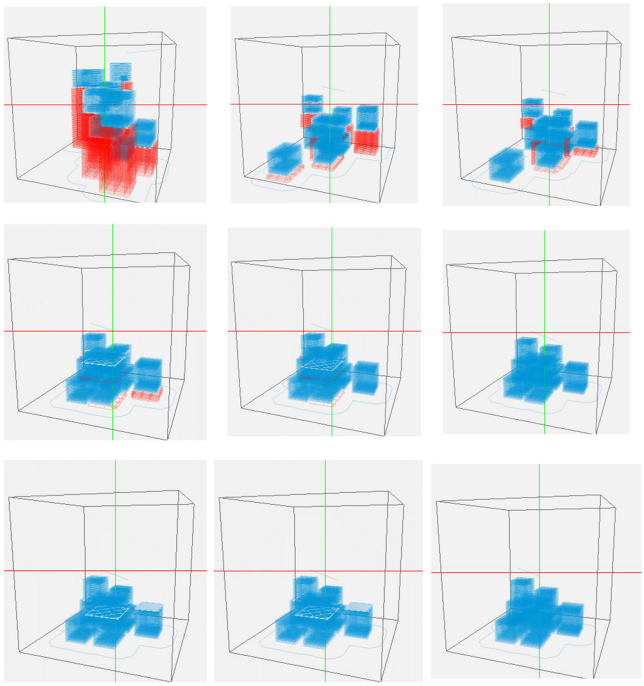

Figure 6. Multi-part placement problem. Best placements of generations of *0, 10, 20, 50, 100, 150, 200, 250* and *299*.

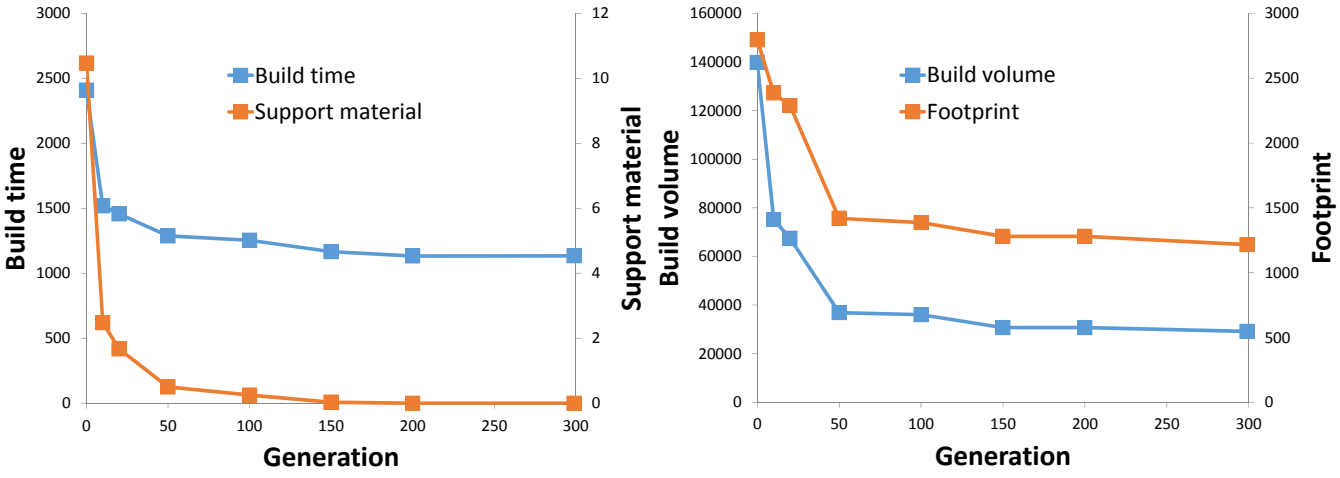

Figure 7. Multi-part placement problem. The reduction of the build time, use of support material, build volume and footprint is observed.