3D Printable Vascular Networks Generated by Accelerated Constrained Constructive Optimization for Tissue Engineering

Andrew A. Guy, Alexander W. Justin, Dulce M. Aguilar-Garza and Athina E. Markaki

Abstract—One of the greatest challenges in fabricating artificial tissues and organs is the incorporation of vascular networks to support the biological requirements of the embedded cells, encouraging tissue formation and maturation. With the advent of 3D printing technology, significant progress has been made with respect to generating vascularized artificial tissues. Current algorithms to generate arterial/venous trees are computationally expensive and offer limited freedom to optimize the resulting structures. Furthermore, there is no method for algorithmic generation of vascular networks that can recapitulate the complexity of the native vasculature for more than two trees, and export directly to a 3D printing format. Here, we report such a method, using an accelerated constructive constrained optimization approach, by decomposing the process into construction, optimization and collision resolution stages. The new approach reduces computation time to minutes at problem sizes where previous implementations have reported days. With the optimality criterion of maximizing the volume of useful tissue which could be grown around such a network, an approach of alternating stages of construction and batch optimization of all node positions is introduced and shown to yield consistently more optimal networks. The approach does not place a limit on the number of interpenetrating networks that can be constructed in a given space; indeed we demonstrate a biomimetic, liver-like tissue model. Methods to account for the limitations of 3D printing are provided, notably the minimum feature size and infill at sharp angles, through padding and angle reduction, respectively.

Index Terms—Constrained Constructive Optimization, vascular networks, tissue engineering, 3D printing

I. INTRODUCTION

VASCULARIZATION is one of the key challenges for engineering new tissues and organs in the lab [1]. Tissue constructs without a perfusable vascular network lack a mechanism for supplying cells with nutrients and removing CO2 and cellular waste; for this supply mechanism to be efficient, it is important that the network displays hierarchy [2], [3]. By incorporating a vascular network into a cell-laden scaffold, tissues with clinically relevant dimensions can be developed and maintained, which could then be used as a replacement for a diseased tissue. Furthermore, tissues and organs consist of multiple interpenetrating tubular systems, including those specific to the tissue function, such as the biliary system in the liver. Until such complex systems are constructed in the lab, fabrication of engineered tissue equivalents for complex organs (e.g. liver, kidney) will remain a formidable challenge.

By algorithmically generating biomimetic and architecturally complex vascular network models, one can rapidly produce vascular networks capable of uniformly supporting high densities of metabolically active cells in a 3D environment. Such models can be used to simulate fluid flow (e.g. wall shear stresses, velocity field), metabolite diffusion, and the cellular environment. Furthermore, template structures from these models can be fabricated via 3D printing techniques (e.g. inkjet, extrusion-based, stereolithography) and incorporated into tissue engineered scaffolds to yield perfusable channels [4]. Such approaches involve a solid sacrificial template around which biomaterials (frequently loaded with cells) are cast. These sacrificial structures are removed (e.g. chemically, thermally, mechanically) to produce a vascular network in the biomaterial with channels of the same diameter as the 3D printed feature.

Since manually constructing vascular structures using computer aided design (CAD) packages is unfeasible for the level of complexity required, procedural generation of vascular networks is greatly preferable. Other applications for such models include comparisons of diseased and physiological tissues [5], surgical planning [6], [7], and the validation of vessel segmentation algorithms [8]–[10].

A number of algorithms have been developed over the past 25 years for arterial tree representation, considering the physiological constraints associated with blood flow and the hierarchical vasculature in the human body. The most common
The required input parameters (those which do not have default values) are listed in Table II. We follow the established terminology of Schreiner [13] and refer to nodes in a tree with no children as terminals, and the node which starts the root vessel as the source. We used a constant-viscosity model, as the vessel sizes we are currently able to manufacture (r ≥ 125 µm) are above the radius at which viscosity correction has previously been applied [14]. Considering steady laminar flow, the pressure difference over a branch, \( \Delta P \), is given as

\[
\Delta P = \frac{8\mu L}{\pi r^4} Q = RQ,
\]  

(1)

**TABLE I**

**Summary of notation**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mu )</td>
<td>Dynamic viscosity</td>
<td>[kPa·s⁻¹]</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>Murray’s law exponent</td>
<td>[-]</td>
</tr>
<tr>
<td>( x )</td>
<td>Position vector in ( \mathbb{R}^3 )</td>
<td>[mm]</td>
</tr>
<tr>
<td>( Q )</td>
<td>Volumetric flow rate</td>
<td>[mm³·s⁻¹]</td>
</tr>
<tr>
<td>( P )</td>
<td>Pressure</td>
<td>[kPa]</td>
</tr>
<tr>
<td>( L )</td>
<td>Branch length</td>
<td>[mm]</td>
</tr>
<tr>
<td>( r )</td>
<td>Branch radius</td>
<td>[mm]</td>
</tr>
<tr>
<td>( R )</td>
<td>Laminar flow resistance of a branch</td>
<td>[kPa·s·mm⁻²]</td>
</tr>
<tr>
<td>( R^* )</td>
<td>Reduced resistance</td>
<td>[kPa·s·mm]</td>
</tr>
<tr>
<td>( f )</td>
<td>Child branch radius fraction of parent</td>
<td>[-]</td>
</tr>
</tbody>
</table>

**Optimization**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( g(x) )</td>
<td>Approximation to the direction of the gradient</td>
<td>[mm³·mm⁻¹]</td>
</tr>
<tr>
<td>( \nabla_s V(x) )</td>
<td>Finite-difference estimation of the volume gradient with respect to the position, ( x ), of a bifurcation</td>
<td>[mm³·mm⁻¹]</td>
</tr>
<tr>
<td>( \Delta )</td>
<td>The stride length to take, having determined the direction: ( x \leftarrow x - \Delta \cdot g(x) / |g(x)| )</td>
<td>[mm]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \eta )</td>
<td>Step length fraction</td>
<td>[%]</td>
</tr>
<tr>
<td>( \tau )</td>
<td>Termination length fraction</td>
<td>[%]</td>
</tr>
<tr>
<td>( \beta )</td>
<td>Step fraction reduction ratio</td>
<td>[%]</td>
</tr>
<tr>
<td>( \zeta )</td>
<td>Iterations between step fraction reduction</td>
<td>[%]</td>
</tr>
</tbody>
</table>

**Construction**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S )</td>
<td>Skip limit: the number of attempts allowed where no progress is made. The most ‘lazy’ selection is achieved with ( S = 0 ), and traditional CCO selection is with ( S = \infty ).</td>
<td>[-]</td>
</tr>
</tbody>
</table>

**Sets and indexing**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( B )</td>
<td>All branches in the tree.</td>
<td>[-]</td>
</tr>
<tr>
<td>( C )</td>
<td>Children, the immediate downstream elements.</td>
<td>[-]</td>
</tr>
<tr>
<td>( U )</td>
<td>The chain of upstream elements to the root. Indexed as ( {1\ldots M} ), with ( 1 ) being the first child of the root and ( M ) being the immediate upstream element.</td>
<td>[-]</td>
</tr>
<tr>
<td>( T )</td>
<td>Bifurcation triad, the elements surrounding a bifurcation:</td>
<td>[-]</td>
</tr>
<tr>
<td>( 0 )</td>
<td>Subscript for the root branch/node.</td>
<td>[-]</td>
</tr>
<tr>
<td>( p )</td>
<td>Subscript for the parent branch/node in a given context.</td>
<td>[-]</td>
</tr>
</tbody>
</table>
where \( L \) and \( r \) are the branch length and radius respectively, \( \mu \) is the dynamic viscosity (taken to be \( 4 \times 10^{-6} \) kPa\( \text{s}^{-1} \) for blood), \( Q \) is the volumetric flow rate and \( R \) the laminar resistance. Physical constraints are provided by flow and pressure consistency between children \( (C) \) and their parent \( (p) \):

\[
Q_p = \sum_{i \in C} Q_i, \quad (2)
\]

\[
P_p = P_i + Q_i R_i \quad \forall i \in C. \quad (3)
\]

In calculations, the reduced resistance, \( R^* \):

\[
R^* = Rr^4, \quad (4)
\]

is used [12], as radii are only determined when needed.

The relationship between parent and child radii is fixed by Murray’s law [2] to minimise power dissipation over the network:

\[
r_p^\gamma = \sum_{i \in C} r_i^\gamma, \quad (5)
\]

where \( \gamma = 3 \), the standard result for minimum work, is the default value. Values in the range \( 2 \leq \gamma \leq 3 \) are physiologically relevant, with lower exponents being observed in high-flow vessels. [3].

In this implementation, we use only nodes with at most two children (i.e. bifurcations), but nodes with higher splitting may be approximated by multiple close bifurcations with low separation. At each bifurcation, the child radii are set by their fraction, \( f \), of the parent radius:

\[
r_i = f_ir_p, \quad (6)
\]

from which follows:

\[
f_1^\gamma + f_2^\gamma = 1, \quad (7)
\]

and, by dividing (5) by \( r_p^\gamma \):

\[
f_1 = \left[ 1 + \left( \frac{f_1}{f_2} \right)^{-\gamma} \right]^{-\frac{1}{\gamma}}, \quad (8)
\]

with a similar form for \( f_2 \). By defining the terminal pressures to be uniform, we can determine the global resistance from the total flow rate, \( Q_0 \), and pressure drop from source to terminals, \( \Delta P_s \), allowing the root radius, \( r_0 \), to be calculated with (1) and (4):

\[
r_0 = \left( \frac{R_0}{R_0} \right)^{\frac{1}{4}} = \left( \frac{Q_0 R_0^\gamma}{\Delta P_s} \right)^{\frac{1}{4}}, \quad (9)
\]

which is propagated down the tree using (6).

Since terminal pressures are uniform, we can use the consistency constraint in (3) to calculate the child radius ratio at a bifurcation,

\[
\frac{r_1}{r_2} = \left( \frac{R_1^2 Q_1}{R_2^2 Q_2} \right)^{\frac{1}{4}}, \quad (10)
\]

with which the radius fractions may be calculated using (8). The reduced resistance of the parent branch at a bifurcation is then calculated from its downstream components as

\[
R^*_p = \frac{8\mu L_P}{\pi} + \left( \frac{f_1^4}{R_1^4} + \frac{f_2^4}{R_2^4} \right)^{-1}. \quad (11)
\]

III. NEW METHOD: ACCELERATED CCO

The established CCO approach to producing multiple non-intersecting networks adds terminals into the network by considering multiple candidate topologies, optimising each for volume and then selecting the minimum volume network with no intersections [11], [15], [22]. The intersection test at each stage leads to poor performance: the most efficient algorithm reported in the literature [15] scales as \( O(N^2 \log N) \), where \( N \) is the number of terminal nodes, requiring days of computing time for complex cases.

We propose a novel method, breaking the process into 3 separate stages: Construction, Optimization, and Collision Resolution (see Fig. 1). With the reasoning that the diameter of any practical vasculature is small compared to the space it serves, networks are constructed in parallel with no knowledge of each other, and collisions between them are resolved by making small excursions around the contact points. This is achieved by introducing a new type of node, the transient (see Table III), which acts to create piecewise approximations to curved branches. Such nodes have previously been considered in the context of single arterial trees [10] to introduce tortuosity.

We will use the term depth of a node similarly to its standard usage with respect to tree data structures: the number of edges between a node and the root. The difference is that we do not regard the transient as increasing logical depth, as we can package segments connected bytransients into single branches, which we view as the fundamental unit of the tree. The tissue volume around a single terminal will be referred to as a unit cell, and will be considered to be supplied by the network if there is a terminal vessel linking this node to the network (the terminal is constructed), or if a much larger vessel is occupying the unit cell. The volume supplied by the entire network is referred to as the perfusion space. When multiple networks are created to meet at the same terminal points (e.g. arterial/venous pairs), we refer to them as being matched. There is no limit to the multiplicity of matching.

### TABLE II

<table>
<thead>
<tr>
<th>Type</th>
<th>Global</th>
<th>Per Terminal</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Delta P_s )</td>
<td>Pressure drop from source to terminals</td>
<td>kPa</td>
</tr>
<tr>
<td>( x_s )</td>
<td>Source node position</td>
<td>mm</td>
</tr>
<tr>
<td>( x )</td>
<td>Position</td>
<td>mm</td>
</tr>
<tr>
<td>( Q )</td>
<td>Volumetric flow rate</td>
<td>mm(^3)(\text{s}^{-1} )</td>
</tr>
</tbody>
</table>

### TABLE III

<table>
<thead>
<tr>
<th>Node types, their relationships, and depth modifiers.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>Source</td>
</tr>
<tr>
<td>Bifurcation</td>
</tr>
<tr>
<td>Transient</td>
</tr>
<tr>
<td>Terminal</td>
</tr>
</tbody>
</table>
We demonstrate techniques for these stages which scale as $O(N \log N)$ in the best case, no more than $O(N^2)$ at worst, with an expected average best case performance for volumes of $O(N^{4/3})$.

A. Creating a bifurcation

CCO iteratively adds terminal sites into the network, meaning that the overall complexity is $N$ times the complexity within each iteration. At each iteration, a branch must be selected from the existing network from which to create a bifurcation, and the location of this bifurcation must be determined. The standard method is to evaluate all existing branches connecting the new bifurcation (the bifurcation triad, $T$, shown dashed). (b)(i) Optimization: The network topology is fixed, and the geometry of the entire network is altered to minimize the total network volume. (b)(ii) Collision Resolution: All networks are considered, and an attempt is made to resolve any disallowed intersections geometrically. If this fails, the network topology is altered by removing terminals to make space for larger vessels, reducing the network density in regions of unresolvable intersections.

![Diagram](image)

We suggest that the path should bifurcate into these terminals as cost is set to be negative and the suitability flag is set - this is then guaranteed to be returned from the search process. The terminal is rejected if no suitable candidates were found or the cost is negative. For the investigations in this paper, we consider only the Euclidean distance from the terminal node to the branch as the cost function: with prescribed flow through branches and pressure being fixed over the network, we expect that longer paths will require larger radii. The suitability test is whether the triangle formed by the set of nodes around the bifurcation (the bifurcation triad, $T$) is likely to degenerate into a line. Extensions which we do not investigate here (but are briefly touched upon in supplementary material) include augmenting the cost with a flow rate term and providing a maximum permissible flow asymmetry.

Algorithm 1 Counted selection method

**Require:** terminal, skipLimit : int[0,∞], evalFunc

**Function** Select(current, count) :

1. best ← current
2. for all child c of current.Branch do
   1. candidate ← evalFunc(c, terminal)
   2. if candidate.Cost < current.Cost then
      1. downstream ← Select(candidate, 0)
      2. if downstream.Suitable and downstream.Cost < best.Cost then
         1. best ← downstream
   3. else if count < skipLimit then
      1. downstream ← Select(candidate, count + 1)
      2. if downstream.Suitable and downstream.Cost < best.Cost then
         1. best ← downstream
      3. end if
   4. end if
3. end for

return best

2) Performance: The counted selection method will never do worse than testing all existing branches, giving an upper bound on complexity of $O(n)$, where $n$ is the number of terminals currently constructed, since the number of branches is $\Theta(n)$. When a low number of skips, $S$, is permitted, the performance should scale as the mean depth of the branches currently in the network. Furthermore, we can guarantee that the depth of the bifurcation created will be less than or equal to the depth produced by searching all existing branches, which will help maintain the good balance required to achieve better query performance. The minimum scaling would be achieved if the tree had uniform depth balance, which would scale as $O(\log n)$.

We expect that the depth distribution along these extremal paths will be approximately uniform, with a mean that scales with the characteristic number of terminals in any dimension, for the following reasons: (1) the average path to the most extreme terminals of a perfusion space will pass a number of internal terminals that scales with the characteristic number of terminals in any dimension; (2) minimum volume principles suggest that the path should bifurcate into these terminals as
soon as they are passed (observed in [16], [19], [23]–[25]). We therefore expect performance of volumes to scale no better than $O(n^{3/5})$. For perfusion spaces where the terminals are arranged as a shell around the inlet, better performance is expected, whereas for long, thin volumes we expect the worst case performance to be achieved.

Perfect depth balance can be achieved by using a space-filling fractal [26], but the suitability of fractals for large-scale organs is debatable [27]–[29]. Notably, manufactured models have been designed to achieve uniform depth balance have only been shown to supply a line or plane of points [30], [31].

B. Volume optimization

The network volume is approximated as being the sum of cylindrical volumes over the the set of existing branches, $B$,
\[
V(B) = \pi \sum_{b \in B} r_b^2 L_b.
\]

Previous implementations have optimized the position of bifurcations as they are added into the network, which we will refer to as ‘incremental’ optimization.

1) Accelerated incremental optimization: Karch et al. [32] considered that when a bifurcation is moved, only the reduced resistances and radii fractions in the direct path upstream to the source must be re-evaluated. We introduce a new derived property, the effective length, $L^*$, which allows the approximation to the volume to be re-evaluated in a similar manner, requiring only calculation along the path to the root, avoiding the costly evaluation of the sum. By expanding the sum and replacing radii with the root radius multiplied by their chained fractions from (6) we get:
\[
V(B) = \pi \left( L_0 r_0^2 + L_1 f_1 r_0^2 + L_2 f_2 r_0^2 + L_{1,1} f_{1,1} r_0^2 + \ldots \right),
\]
from which common factors may be recursively extracted:
\[
\pi r_0^2 \left( L_0 + L_1 f_1 + L_{1,1} \ldots + \right) + r_0^2 \left( L_2 + \ldots \right).
\]

We can now define the effective length relationship at terminals and bifurcations:
\[
L_{\text{term}}^* = L_{\text{term}},
\]
\[
L_r^* = L_r + f_1^2 L_1^* + f_2^2 L_2^*.
\]

The total volume downstream of a branch is therefore
\[
V(B) = \pi r_0^2 L_0^*.
\]

The update rule can be generalized to any cost function of the form considered by Schreiner et al. [23]:
\[
C(B, \lambda, \rho) = \sum_{b \in B} r_b^6 L_b^4,
\]
where $\lambda, \rho$ are arbitrary constants. Using the following substitutions: $L_i \rightarrow L_i^*, f_i^2 \rightarrow f_i^*$, the general form of the evaluation becomes:
\[
C(B, \lambda, \rho) = r_0^6 L_0^*.
\]

2) Batch optimization: The fundamental limitation of incremental optimization is that the effect of moving a single bifurcation on the entire network becomes negligible as the total number of nodes becomes large. We will deal with any intersections at a later point in the algorithm, we may interrupt the build at various stages to optimize every single bifurcation at once.

Karch et al. [32] demonstrated that the volume of the whole network when perturbing a single bifurcation has a single minimum somewhere within the bifurcation triad, permitting the use of the simplest method of minimization; gradient descent of the volume with respect to the position of each bifurcation:
\[
x' = x - \eta \nabla V(x),
\]
where $\eta$ is some small parameter. Whilst we could use the accelerated volume calculation method to estimate the gradient at each node, we would pay a computational cost per iteration of $N$ times the mean depth of the whole tree, giving an expected $O(N^{4/3})$ scaling for perfusion spaces.

A constant time approximation to the gradient, giving an overall scaling per iteration of $\Theta(N)$, is justified here. Starting by differentiating (17) with respect to a bifurcation position,
\[
\nabla V(B) = \pi r_0^2 \nabla L_0^* + 2\pi L_0^* r_0^2 \nabla r_0^2,
\]
we note from (9) that the root radius is a function of only root reduced resistance, $R_0^*$, given that the flow rates and total pressure drop are fixed by the logical configuration of the tree. Since changes in $R^*$ propagate upwards from the bifurcation to the root, we see from (11) that there is a scalar gradient chain through the upstream nodes, $U_m; m \in \{1 \ldots M\}$, where 1 denotes the first child of the root and $M$ the immediate upstream node of the bifurcation, and the vector terms are introduced at the bifurcation triad via their physical lengths (using $C$ to denote the set of child branches):
\[
\nabla V(B) = \frac{\partial R_0^*}{\partial R_i^*} \left( \prod_{i=1}^{M-1} \frac{\partial R_{i+1}^*}{\partial R_i^*} \right) \frac{\partial R_{i+1}^*}{\partial R_i^*} \nabla R_{i+1}^*(x),
\]
\[
\nabla V(B) = \frac{8\mu}{r_0^2} \left( \nabla L_p(x) + \sum_{i \in C} \frac{\partial r_i^2}{\partial L_i} \nabla L_i(x) \right).
\]

In order to avoid vanishing gradients due to deep chains, we wish to approximate the gradient direction and work out the step size later. Defining $\beta = r_1/r_2$, the bifurcation relationship yields:
\[
\frac{\partial R_p^*}{\partial R_1^*} = -\left( f_1^4 R_1^* + f_2^4 R_2^* \right)^{-2} \left( \frac{(1 + \beta^{-2})^{-\frac{1}{2} - \frac{1}{2}}}{(1 + \beta^{-2})^{-1}} \right) \left( \frac{1 + \beta^{-2}}{1 + \beta^{-2}} \right)
\]
\[
\nabla R_0^*(x) \text{ therefore points in the same direction as } \nabla R_p^*(x).
\]

The expression for the second child can be found by exchanging the indices. Considering that
\[
\frac{z}{1 + z} - 1 \leq 0 \quad \forall z \geq 0,
\]
we know that the gradient chain elements are all positive.
Disregarding the scaling factor \((8\mu/\pi)\) and evaluating the length gradients in (23), we get \(g(x)\), the approximation to the gradient direction:

\[
g(x) = \frac{x - x_p}{L_p} + \sum_{i \in C} \frac{dR^*_i}{dR^*_i} (x - x_i).
\]

(26)

For the approximation to be suitable requires the terms in (21) from the effective length derivative to be either pointing in a similar direction, or small. The gradient of the parent effective length at a bifurcation with respect to the position of a different bifurcation downstream is:

\[
\nabla L^*_p(x) = \nabla L_p(x) + \sum_{i \in C} \left( f^2_i \nabla L^*_i(x) + 2 f_i L^*_i \nabla f_i(x) \right),
\]

and we claim the following:

1) \(\nabla L_p(x) = 0\) for all terms not in the bifurcation triad. Whilst not in the exact same direction as (26), they have similar components (physical length gradients multiplied by positive coefficients) and arrive at the root expression having been attenuated by every \(f^2_i\) on the way, reducing their effect.

2) One of the \(f^2_i \nabla L^*_i(x)\) terms at each bifurcation is zero, as it has no downstream changes.

3) Each \(f_i L^*_i \nabla f_i(x)\) term has a gradient chain downstream to an expression proportional to (26). A pair is introduced at each bifurcation, and are attenuated by fewer \(f^2_i\) terms than the physical length terms at the root. It is possible for the proportionality to (26) of this term to be negative: a small number of bifurcations will suffer this at enough nodes that the approximation will be pointing in the reverse direction to the true gradient.

The approximation of the gradient direction in (26) seems acceptable when considering the bulk movement of nodes, increasingly so at larger depths. In the case of transient nodes, we can view the parent and child branches together as a single element, meaning that only the length term of the parent reduced resistance is involved:

\[
\nabla R^*_p(x) \approx \left( \nabla L_{total}(x) = \frac{x - x_p}{L_p} + \frac{x - x_c}{L_c} \right),
\]

(28)

which gives a direction that is the mean of the branch unit vectors towards the transient, and the expected result of equilibrium lying on the line between parent and child.

All that remains is to determine the perturbation distance, \(\Delta\), which is set to be a fraction, \(\eta\), of the minimum length in the bifurcation triad, \(T\):

\[
\Delta = \eta \cdot \min(L_p, L_1, L_2).
\]

(29)

However, without a termination condition we may find that bifurcations are pulled into overlap with other nodes, increasing collisions. When the overlap is at a terminal, this can have disastrous consequences for the completeness of perfusion due to terminal culling (see §III-C.6). Movement is prevented when the shortest triad length is less than a termination fraction, \(\tau\), of the minimum distance between surrounding nodes:

\[
\min(L_p, L_1, L_2) \leq \tau \cdot \min(|x_1 - x_p|, |x_2 - x_p|, |x_2 - x_1|).
\]

(30)

Should equilibrium lie outside of the termination regions, low amplitude oscillations or limit cycles may be established due to the increased step size in this region. Convergence can be encouraged in practice by ensuring that \(\eta\) starts small (limiting the potential overstep) and by reducing \(\eta\) over time. A simple implementation is to specify a reduction fraction and block length, with care taken to ensure that the block length is appropriate for the initial \(\eta\) - small strides need more iterations before reduction.

3) Initial bifurcation position: If we do not optimize the network incrementally, we need a method of determining where to place a bifurcation upon creation. Rather than place it at the arithmetic mean position of the nodes in \(T\), we propose that a more suitable (yet still trivial to calculate) approximation to the optimal position, \(\hat{x}\), is a weighted arithmetic mean, with the weighting being the flow:

\[
\hat{x} = \frac{\sum_{n \in T} Q_n x_n}{\sum_{n \in T} Q_n}.
\]

(31)

Considering the extreme case, we see that this reduces the deviation taken by high flow branches:

\[
Q_1 \gg Q_2 \implies \hat{x} \approx \frac{1}{2}(x_p + x_1),
\]

(32)

which helps minimize the length of the main branches.

C. Collision resolution

The collision resolution scheme is an iterative approach with three actions available:

1) Insert a transient in a branch to create an excursion.

2) Move a bifurcation/transient.

3) Remove a terminal.

We aim to minimize the removal of terminals, and care is taken to ensure that this only occurs when a much larger branch has no possible place to be moved - in this case the unit cell is essentially marked as having no useful tissue volume to support, and the loss to total flow does not harm the completeness of perfusion.

1) Detection: A naive approach to collision detection would be to test every branch against each other, which gives a complexity of \(\Theta(N^2)\). By the use of an appropriate hierarchical bounding tree we can achieve \(O(N \log N + k^{1+\epsilon})\) query, where \(k\) is the number of intersections, by rejecting entire groups using simple tests. In this case, we use Axially Aligned Bounding Boxes (AABBs), which can find a rejection in at most 6 comparisons [33].

Unfortunately, structures optimized for query have a high construction cost [34] – we use the assumption that we have produced a roughly balanced tree, combining sibling bounds at bifurcations into downstream bounds, which are then combined with their parent branch local bounds to produce the global bounds for that branch. This is by no means optimal in query, as at each layer of the tree we have both a leaf element and two child branches. We again expect query performance to scale as \(N\) times the mean depth – we sacrifice optimal query for linear cost of construction, as we must recalculate the bounds for each iteration of resolution. The query routines are demonstrated in Algorithms 2 and 3.
### TABLE IV
INTERSECTION DATA AND THEIR PURPOSE.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indeterminate</td>
<td>Boolean</td>
<td>Whether a single normal mutual direction could not be established.</td>
</tr>
<tr>
<td>NormalAB</td>
<td>Vector3 (Normalised)</td>
<td>If determinate, the normal from branch A to B. If indeterminate and collinear, a randomly generated mutual normal. Otherwise, the in-plane direction from A to B.</td>
</tr>
<tr>
<td>Fraction, Closest (A,B)</td>
<td>Real[0,1], Vector3</td>
<td>The fraction along each branch of the closest point, and the evaluated point itself. Only valid if determinate.</td>
</tr>
<tr>
<td>Start, End (A,B)</td>
<td>Real[0,1], Real[0,1]</td>
<td>If indeterminate, the start and end fractions of intersection.</td>
</tr>
<tr>
<td>Distance2, Overlap</td>
<td>Real, Real</td>
<td>The square separation distance, used for determining intersection, and the overlap if intersecting.</td>
</tr>
</tbody>
</table>

![Algorithm 2](image1)

**Algorithm 2** Collision detection

**Subroutine** TestTree(a, b) :

- `immune ← GenerateImmuneSet(a)`
- `TestDownstream(b, a)`
- **for all** child ca of a **do**
  - **if** TestBounds(ca.Global, b.Local) **then**
    - `TestTree(ca, b)`
  - **else**
    - **for all** child cb of b **do**
      - **if** TestBounds(ca.Global, cb.Global) **then**
        - `TestTree(ca, cb)`
    - **end if**
  - **end if**
- **end for**

![Algorithm 3](image2)

**Algorithm 3** Collision detection downstream search

**Subroutine** TestDownstream(root, check) :

- **if** not immune.Contains(check) **and** TestBounds(root.Local, check.Local) **then**
  - `data ← TestBranches(root, check)`
  - **if** data.Intersecting **then**
    - `intersections.Add(data)`
  - **end if**
- **end if**
- **for all** child c of root **do**
  - **if** TestBounds(check.Local, c.Global) **then**
    - `TestDownstream(c, check)`
  - **end if**
- **end for**

We model branches as capsules, which have simple intersection tests. The intersection data produced are shown in Table IV. Intersections where a single mutual normal direction cannot be established (the branches are either co-directional or collinear) are referred to as *indeterminate*, and are handled separately, as there may not be a single point of closest approach between the centres of the branches.

2) *Immune sets:* In the case of matched networks, the parent branch of a terminal node must intersect with each of the parents of the other matched terminals. An *immune set* is generated whenever the end node of a branch indicates it has matched partners, and intersection tests with branches in the set are skipped. For intersections within a network, each branch must generate an immune set of at least its sibling, children and parent. However, in cases where the branches have low slenderness we allow the set to be expanded: branches are recursively added both upstream and downstream until the logically closest node pair between the test branch and set-generating branch no longer touch.

3) The determinate case: Firstly, a decision is made as to whether the start and end nodes of each branch are involved in the intersection, by comparing the distance between the nodes and the point of closest approach, \( d \), to the branch radius. Using the radial slenderness of the branch, \( s_r \), the fraction along the branch of closest approach, \( \chi = d/L \), and a *radial capture* constant, \( k_r \), the start and end conditions are respectively

\[
\begin{align*}
\frac{d}{r} &= \frac{d}{L} = \chi s_r < k_r, \\
(1 - \chi)s_r &< k_r.
\end{align*}
\]

If either node passes its test, a request is moved to file the intersecting nodes by a perturbation vector, \( v_p \). If both nodes fail, a transient is requested at the closest approach point of the centrelines plus \( v_p \). The perturbation distance is calculated by multiplying the overlap, \( \delta \), defined as the sum of the branch radii minus the separation between the branch centrelines, by an aggression factor \( k_a \), which tries to prevent the potential increase in branch radii after movement from causing the same collision again; this distance is then divided between the branches by their weighting fractions, which must satisfy

\[
w_A + w_B = 1.
\]

The perturbation vector, \( v_p \), is therefore given by

\[
v_p = \delta k_a \hat{w} n
\]

where \( \hat{w} n \) is the weighted normal in the relevant direction for each branch.

After all intersections have been processed, all node perturbation requests are averaged into a single perturbation for this iteration, which is clamped to ensure a minimum step size, \(|v|_{\min}\). This reduces the number of iterations required, but can introduce needless tortuosity if \(|v|_{\min}\) is too large. Transient requests are also averaged to insert a single transient per branch in each iteration. There is no minimum distance for these, as any unresolved intersections will be captured by the newly inserted transient in the next iteration, which will then follow the rules for node perturbations to ensure a solution is found.
4) **The indeterminate case:** In this case the node tests are of the same form as the determinate case, but with the length fraction replaced by the intersection range start and end fractions, respectively. There is now the possibility to request both node perturbations and transient insertion: all node requests are made the same as before; then, should any node in the intersecting set of A be stationary, a transient is requested at the midpoint of B’s intersection range plus the perturbation vector, and vice versa.

5) **Perturbation weighting:** It often makes little sense to perturb a high-flow and low-flow path equally, as volume minimization suggests that the path with lower flow should yield. The simplest way to weight the perturbations to achieve this is using the total flow fraction of the opposing branch. However, between networks where the overall pressure differences are substantially different, the effect of perturbations on the root radius (and subsequently all radii) is amplified in the low-pressure case, so the user can specify a relative compliance, \( C \), for each network to compensate for this. The overall perturbation weights are therefore:

\[
W_A = \frac{Q_B C_A}{Q_A C_B + Q_B C_A},
\]

\[
W_B = \frac{Q_A C_B}{Q_A C_B + Q_B C_A},
\]

which satisfy the constraint in (35).

6) **Terminal culling:** As generated volumes become larger, the probability of situations where root branches grow to sizes similar to the terminal spacing increases, creating potentially unsolvable collisions with the stationary terminal nodes. Even if a solution were to be found, the tortuous structures created by diverting the flow around the terminal would increase volume, and optimization steps would most likely pull the branches back into the terminal, perpetuating the issue. To resolve this, a culling stage was added.

Whenever a request to move a terminal is made, the resolver increases its cull count by a user-specified penalty, \( z_p \). After each iteration of resolution, the resolver checks the count of each node against a threshold value, \( z_t \), and any nodes with counts greater than this are culled from their networks, alongside their matched partners. The resolver then decrements the count of all nodes, clamping values at zero.

Aggressive culling prevents the distortion of branches close to the source at the expense of those downstream, where solutions may be available if more attempts were allowed. To permit less aggressive culling whilst still protecting the root branches a critical ratio, \( \beta_c \), is specified and if the condition

\[
Q_{\text{term}} \beta_c < Q_{\text{branch}}
\]

is met, the terminal is immediately removed from the network.

7) **Ensuring 3D printability:** The radii of vessels are kept above the printer minimum feature size, \( r_{\text{min}} \), by adjusting the overall pressure drop if necessary. To ensure vessel spacing is above the minimum feature size of the printer, their radii are padded for collision resolution purposes by another input parameter, \( r^+ \).

In terminal segments spaces between the narrow vessels are at risk of infill in printing, should the angle at which they meet be too large. We therefore provide the user with the option to smooth terminal segments as a final action before exporting the model to a CAD format. We export to Autodesk® Inventor® 2017 onwards, as well as directly exporting to .STL files.

### IV. Results

A. **Collision resolution**

A demonstration of collision resolution and terminal smoothing working for a simple case is shown in Fig. 2. Whilst not perfect, angles have been mostly reduced to below 90°. An instance of terminal culling can also be seen, as there is a terminal in Fig.1a(ii) (right inset, circled) that is not present in Fig.1b(ii). Regions such as this, where we have one or more high-flow branches, are exactly where we expected to need to cull terminals.

B. **Computational scaling**

Performance of construction was tested by generating networks for an *Edge-fed Cubic Volume* (ECV), where the source node was placed above the center of one face of a cubic terminal layout; a *Centre-fed Spherical Shell* (CSS), where the terminals were placed uniformly over the surface of a sphere, and the source node placed at the center; and an *Above-fed Planar Surface* (APS), where a square surface of terminals is supplied by a network descending from a central out-of-plane source node. An example of each network is shown in Fig. 3. An important note is that for the CSS case the skip limit, \( S \), must be increased. If \( S = 0 \), the order of construction becomes important: if the children of the first bifurcation point away from the target site, the bifurcation will be created at the root. Should the next terminal have a similar relationship to the new initial bifurcation, the process will repeat and the output will be a spiky sphere. For each setting 100 tests were performed with a fresh random number generator, seeded by the system tick count.

Fig. 4 shows a polynomial scaling estimate for each case. It can be seen that the mean performance on the ECV scales faster than previous algorithms by roughly a factor of \( N^{1/2} \), with a number of outliers seeing far better performance. The CSS case scales best, despite the fact that \( S \) had to be increased for this geometry. Whilst we might have expected the APS to have seen better performance again, due to the ability to use lower values of \( S \), the minimum volume structure does not bifurcate out-of-plane very often: the optimal volume solution is to move rapidly to the plane and then branch out in-plane (see Fig. 3), which we would expect to scale as \( O(N^{3/2}) \), using the argument that the extremal paths pass a number of terminals scaling as \( O(N^{1/2}) \).

1) **Terminal depth distributions:** Comparing terminal depth distributions (which we will denote as \( d(i) \) to represent the fraction of terminals at depth \( i \)) across sizes is meaningless without a normalization: in this case, we know that the perfectly balanced tree will have all of its terminals at a depth of \( \log N \), and we therefore compare distributions normalized...
against this. These comparable distributions are referred to as $d(i')$, where $i' = i/\log N$. For notational convenience, we do not explicitly mention the dependence of $d$ on $N$ and geometric factors.

Fig. 5 shows that, as expected, the depth distribution mean and standard deviation for an ECV is seen to grow almost exactly as $N^{1/3}$, suggesting that we could do no better than $O(N^{4/3})$ performance overall, if we were able to find a cost function to use in the construction phase that prevents needless downstream searches more effectively than the Euclidean distance. The distribution for the CSS case shows a very slight decay (Fig. 6), suggesting that the performance floor would be closer to $O(N \log N)$ if we were to supply a shell-like geometry for which we may use $S = 0$.

2) Collision resolution: The number of iterations required for total collision resolution (Fig. 7) is seen to have a weak increase with the number of terminal nodes, but remains fairly constant over the range of problem sizes that we are currently interested in. The position of source nodes determines the shape of the distribution: when inlet and outlet are on opposite sides of the perfusion volume, low-flow branches of each network will be intersecting high-flow branches of the other,
Fig. 3. Single sample networks for the 3 configurations used for testing performance: edge-fed cubic volume (ECV), centre-fed spherical shell (CSS) and above-fed planar surface (APS), with their bounding geometries shown.

Fig. 4. Estimate of polynomial scaling of build time, $t$, for an edge-fed cubic volume (ECV), centre-fed spherical shell (CSS) and above-fed planar surface (APS). Note that the CSS case required $S = 1$ due to the geometry of the problem, whereas the others were able to use $S = 0$. 100 tests performed at all settings.

which is more likely to require many iterations to fully resolve.

An estimate of the computational scaling of a single iteration of collision resolution is shown in Fig. 8. Whilst there is a lot of variability in the time taken, it is seen to scale better than construction. However, for sizes that we are currently generating, the time taken is similar to, if not greater than, the time taken in construction.

C. Volume of generated networks

The mean volumes of ECV networks with 1000 terminals generated using various selection methods, $\bar{V}$, are shown in

$\begin{array}{c|c|c|c|c}
 S & 0 & 1 & 2 & \infty \\
 \hline
 \bar{V} / \bar{V}_\infty & 0.8869 & 0.9074 & 0.9193 & 1.0000 \\
 s_{\bar{V}} / \bar{V}_\infty & 0.0243 & 0.0245 & 0.0245 & 0.0383 \\
\end{array}$

Fig. 6. Terminal depth distributions for the spherical shell (CSS). (a) The observed comparable distributions $d(i')$ for a range of terminal counts $N$, which remain similar despite large changes in $N$. (b) The normalized mean, $d$, and standard deviation, $s_{d}$, of terminal depth distributions are seen to vary much less with $N$ than the ECV case. 100 tests performed at all settings.
Fig. 7. The number of iterations required for collision resolution for an ECV network with (a) neighbouring and (b) opposing inlet and outlet. Box plot shows minimum, median, maximum, and interquartile range. Outliers (determined as being more than 1.5 interquartile ranges beyond the upper and lower quartiles) are plotted as crosses. 100 tests performed at all settings.

Fig. 8. An estimate of a polynomial scaling law for the time required per iteration of collision resolution for an ECV network. 100 tests performed at all settings.

Table V, normalized against the reference volume of the traditional ‘full’ search, $\bar{V}_\infty$. The standard deviations, $s_V$, seen in this are also shown, again normalized by $\bar{V}_\infty$, to give an idea of the susceptibility of each method to the randomness in the build order. The optimization scheme used for these was incremental and flow weighted. It can be seen that the minimum volume is achieved using the most aggressive scheme (skip limit, $S = 0$) and that the standard deviation of volumes is lower when using counted selection compared to the full search case, $S = \infty$. Due to the fact that it is not possible to use $S = 0$ with all geometries, we recommend using $S = 1$ as a default.

1) Verifying the gradient approximation: Fig. 9 shows the distribution of alignment between the fast gradient approximation, $\tilde{g}(x)$, and a finite-difference estimation of the gradient, $\nabla x V(x)$, with stride size $\epsilon = 0.01$ mm, measured as the dot product between their directions:

$$\text{Alignment} = \frac{\tilde{g}(x)^T \nabla x V(x)}{|\tilde{g}(x)||\nabla x V(x)|}.$$  \hspace{1cm} (39)

This was measured for all bifurcations in 100 instances of an 8000 terminal ECV as a function of bifurcation depth. Whilst we see somewhat poor alignment at lower bifurcation depths (up to 10% of nodes being moved in the wrong direction), at depths at which the majority of the bifurcations reside (25 to 75) we find that 80% of nodes show alignment above 0.8, showing that in almost all cases, the $O(N^{1/3})$ saving in complexity does not excessively sacrifice optimality.

2) Effect of optimization schemes: The effect of various optimization schemes is shown in Fig. 10. At each size, the mean volumes and confidence intervals were normalised against the flow-weighted, incrementally optimized scheme for that size, denoted $V_{Q,I}(N)$. Our implementation of an incremental optimizer terminated when it failed to reduce the volume by at least a certain fraction (in this case $10^{-4}$) of the current volume. Once the network is large enough, it is not possible for the newly created bifurcations to make such a significant impact on the volume, and the optimizer terminates almost immediately. This is most likely the reason for flow weighting having such an effect when using the incremental optimizer; when many bifurcations are deemed too insignificant to spend time optimizing individually, the effects of better initial position for each have a large impact on total volume.

Batch optimization is a clear improvement over incremental, giving a mean volume reduction greater than 10% as well as being more consistent. In this example, we interrupted
the build at fractions of (0.1, 0.2, 0.5, 1) to perform 10 iterations of optimization, which gave similar total time to the incrementally optimized case at the intermediate value of \( N = 8000 \). Even with batch optimization, flow-weighting is still relevant - unit-weighting gives worse performance and higher uncertainty at all terminal counts, and worsens as terminal count increases. This is due to the fact that geometric and topological optimization are not actually separable: flow-weighting provides a better guess of the optimal geometric configuration, which leads to better topology (we select candidates based on distances, a geometric property), which in turn is more responsive to geometric optimization.

Using finite-difference gradient estimation at depths below 10 produced a small (<1%) improvement to mean volume in the case of \( N = 8000 \), suggesting that the power of batch optimization arises from the bulk movement of nodes that are individually insignificant, where the approximation to the gradient is better.

**D. Pseudo-biomimetic 3D printable vascular systems**

One of the key aims in manufacturing networks is that the inlet vessel is of a realistic diameter, to allow future anastomoses to native vessels. To achieve this, we adjusted pressures after construction to give correct dimensions at the inlet. Since the focus is on the higher flow branches, Murray’s exponent was set to \( \gamma = 2 \) to give more realistic radius decay in these vessels [3]. We have aimed to produce the most volume-optimal models, which can result in highly asymmetric bifurcations near the root. The most extreme bifurcation ratio, \( \beta = r_1 / r_2 \) where \( r_2 > r_1 \), we have recorded is \( \beta = 0.082 \), which is within the reported range in real tissue, \( \beta > 0.05 \) [35], but real tissue does not show as many large asymmetries as the volume-optimal designs. A short discussion on the distribution of vessel lengths and radii, as well as a method to improve biomimeticism and its impact on performance, is provided as supplementary material. For these models, \( r_{\text{min}} = r^+ = 125 \, \mu m \), the current 3D printable limit in removable material.

![Fig. 10. Relative volumes at various sizes with \( S = 2 \) under various initial bifurcation weightings (unit, \( w = 1 \), and flow, \( w = Q \)) and optimization schemes, with 95% confidence bounds. Note that the points for \( w = 1 \) with batch optimization have been offset for clarity. 100 tests performed at all settings with ECV configuration.](image)

Whilst we have not yet implemented the accurate geometry of organs, we have used simple approximations with realistic dimensions to demonstrate 3D printable models at large scales. For a method to adapt the collision resolution technique to approximate real organ geometry, the reader is referred to supplementary material. Fig. 11 shows an approximation to a human-sized kidney model as a cuboid with a matched arterial/venous network pair. Provided that there is sufficient space to resolve the intersections without excessively culling terminals (dependent upon the ratio between unit cell spacing and minimum vessel diameter, as well as the number of networks), the Accelerated CCO approach can be successfully deployed over a wide range of unit cell densities. The volume of the kidney was taken to be 5 cm \( \times \) 5 cm \( \times \) 10 cm, with arterial diameter 5 mm and venous diameter 6 mm [36]–[38].

A human-sized liver (Fig. 12) was approximated as a triangular prism supplied and drained by a quadruply-matched network. The volume was taken to be 20 cm \( \times \) 15 cm \( \times \) 5 cm, with inlet diameters of 12 mm (hepatic vein), 10 mm (portal vein), 6 mm (hepatic artery), and 4.5 mm ( bile duct) [39]–[42].

The model displays several biomimetic features; these include the four networks present in the liver, alongside comparable inlet sizes and overall dimensions. However, there are currently some limitations to the level of biomimicry using this algorithm. In the liver, the hepatic artery, portal vein and biliary tree follow the same path through to the lobe, segmental and sub-segmental regions. In our model, the three networks follow independent paths through the structure, and the sub-unit regions are not defined at any length-scale. However, these characteristic elements of the physiological liver could be incorporated into our models through subdivision of the perfusion space, and by initially creating a single network which is split into three. The inability to 3D print capillary-
those previously reported on vascular trees. Computational
prevention). Construction times are significantly faster than
and manufacturing constraints (minimum feature sizes, infill
network volume, Murray's law, no short-circuit intersections),
pressure consistency), physiological constraints (minimum
method accounts for physical constraints (flow conservation,
material.
A discussion and close-up renderings of possible unit cell structures is included as supplementary material.
It is of note that each unit cell in the CCO model is supported with its own terminal, therefore producing a uniform pressure distribution in the perfusion space, which leads to biomimetic vascular networks. This is in contrast to networks fabricated via 3D printing methods where the perfusion space is typically supplied through a series of one-dimensional channels [30], [43], [44] leading to variation in pressure across the perfusion space. Further, the space occupied by connections between inlet and terminal one-dimensional channels will produce unsupported regions within the tissue construct.

V. CONCLUSION

The success of tissue engineered constructs largely depends on the incorporation of perfusable vascular networks which can support the biological functions of the embedded cells. While 3D printing techniques have evolved to create complex structures with geometric precision, currently there is no method for automatic generation of 3D printable vascular networks for arbitrary scales and spaces, and for interpenetrating networks. Here, we report a method for generating 3D printable vasculature templates, based on Constrained Constructive Optimization, by dividing the process into distinct stages of construction, optimization and collision resolution. The method accounts for physical constraints (flow conservation, pressure consistency), physiological constraints (minimum network volume, Murray’s law, no short-circuit intersections), and manufacturing constraints (minimum feature sizes, infill prevention). Construction times are significantly faster than those previously reported on vascular trees. Computational complexity is observed to scale as \(O(N^{1.57})\), a factor of \(N^{0.43} \log N\) better than the previous best implementation (\(N\): number of terminal nodes). The conversion of these networks into perfusable channel systems, embedded in biomaterials, requires the use of sacrificial 3D printing techniques; however, discussion of this is beyond the scope of this computational work.

REFERENCES


