Two combinatorial problems from craniosynostosis

by

Mathieu Rundström

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2022

Abstract

The art of cranial vault remodelling surgery is amazing and fascinating, but remains much of that, an art, to this day. In this thesis, we provide two mathematical approaches to tackle cranial vault remodelling surgery, in the hopes of bringing some insights from the perspective of combinatorial optimization. First, an integer programming formulation which we analyze experimentally, and second, a more theoretical approach using the local ratio.

Acknowledgements

First and foremost, I would like to thank my supervisors Ricardo Fukasawa and Jochen Könemann for their invaluable guidance, support, and patience throughout the past two years. It has been a great learning experience.

I also wish to thank my readers William Cook and Kanstantsin Pashkovich for their time and feedback.

Thank you to Dr. John Phillips from the Hospital for Sick Children and Chris Woodbeck from the Hospital for Sick Children and the University of Waterloo for their expertise and insight.

Lastly, I would like to thank Marina Drygala for their inspiring work.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis pertains to developing and employing mathematical models to tackle a medical condition called Craniosynostosis. Craniosynostosis is a birth defect that affects 1 in 2000 infants [15]. It is caused by premature fusing of cranial vault sutures and manifests itself in abnormal skull growth patterns. In turn, this can lead to pressure on the brain as it grows rapidly in infancy, and cognitive impairment. See figure 1.1 for an infant affected by craniosynostosis.

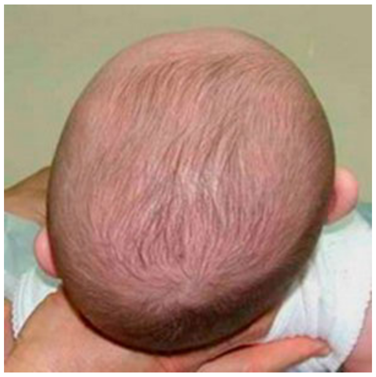

Figure 1.1: An infant affected by craniosynostosis [2].

Craniosynostosis can be treated by surgery. At a high level, in such a surgery, the skull bone is cut into an ensemble of pieces. These pieces are then moved, rotated, and sometimes bent, with the goal of forming a more normal shaped skull. One type of Craniosynostosis surgery is the so called front-orbital reshaping, where the *bandeau* (a horizontal segment of bone right above the eye sockets—see figure 1.2) is bent in locations where kerfs (small cuts) are made to make the bone malleable. The problem of choosing where to cut the bandeau and where to place the resulting pieces was studied by Drygala in their thesis [5]. More specifically, in the Curve Reshaping Problem with Rearrangement, given a so-called ideal piecewise-linear curve g, and a deformed piecewise-linear curve f representing the shape of the infant's bandeau, the problem they studied was to find subsegments of f, and to match these to subsegments of g, minimizing a cost function incorporating how good this matching is. In Drygala's thesis, and in [4], it is shown that this problem is strongly NP-hard using a reduction from the 3-Partition problem, even when cut locations are prescribed. That is, one can only hope to obtain approximate solutions for these problems in polynomial time. However, for the problem of selecting the best incisions with no rearrangement of pieces allowed, i.e., when the bandeau is only bent at cut locations, the authors in [4] provide a dynamic programming algorithm.

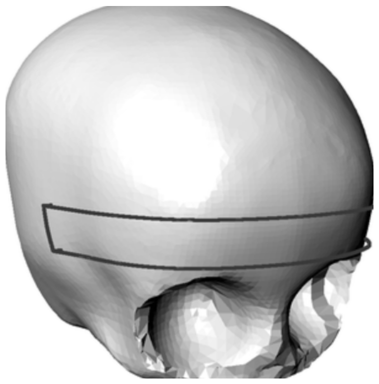

Figure 1.2: The outline of a bandeau [13].

Another type of surgery is the more general so-called cranial vault remodeling surgery. The cranial vault is the area enclosed by the skull, containing the brain. This type of surgery can be used when the defects are not restricted to the bandeau area of the skull. In such a surgery, a greater part of the skull bone is cut into pieces. The pieces are then potentially moved, rotated, or bent, until placed in their final configuration. The problem of choosing how to cut these pieces, and where to place them, is less well studied from a mathematical perspective. This is the problem we aim to tackle in this thesis.

This thesis is divided into two main components, described respectively in chapter 2 and chapter 3. Chapter 2 revolves around modeling the cranial vault reshaping problem as an integer program. Here, we allow for certain types of cuts to form pieces. The goal is then to place these pieces in a best way possible, provided the pieces cover a certain portion of an ideal skull. To know what to strive for, it would be good to have an idea of what a "normal" skull resembles. Indeed, surgeons and researchers at the Hospital for Sick Children have developed a library of normative skulls modelling ideal skull shapes of infants [13]. The second component, covered in chapter 3, is greatly influenced by Drygala's reduction to a scheduling framework of Bar-Yehuda, in which they achieve a $\frac{1}{2}$-approximation in the bandeau setting outlined above [1][5]. The key idea in this reduction is to think of bone pieces cut from the bandeau as time intervals to be scheduled on the real line. This is reasonable because one can think of the bandeau as a curve. Using similar techniques to Drygala in the bandeau setting, but applied to the cranial vault setting, we now "schedule" bone pieces in the plane. Thinking of the skull as the plane might be a greater assumption than considering the bandeau as a curve, but it allows for a simple analysis similar to the bandeau case. Under this simplifying assumption, we show that the approximation ratio depends only on how pieces can overlap. We give some examples of classes of pieces leading to constant approximation ratios.

Chapter 2

An Integer Programming Approach.

In this chapter we are concerned with the following problem. Given an ideal skull, and a deformed skull divided into regions (see figures 2.1, 2.2), cut each region at most once, and place the resulting pieces on the ideal skull in some best possible way. This best possible way will be a balance between covering a certain area on the skull, and minimizing the volume between placed pieces and the target ideal skull. We make this precise in section 2.1. The motivation behind cutting each piece at most once is twofold: it makes the problem simpler to model, and few cuts ensures the surgery does not last too long. The high level idea is inspired by Jessie Yeung's work [17]. They gave a model for the problem of placing pre-cut pieces. Throughout this chapter, we highlight key differences between their approach and ours as they appear. A major contribution of ours is that our model does not require pre-cut pieces. After describing the problem, we give an integer programming (IP) formulation of the problem in section 2.1. In section 2.2 we provide some variants of the base model from section 2.1. Finally, in section 2.3 we present some experimental results.

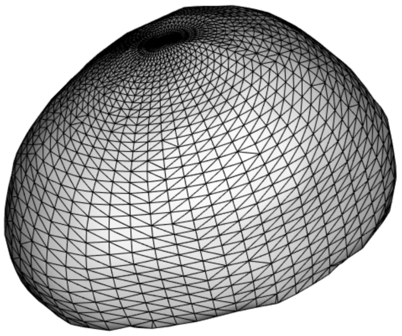

Figure 2.1: An ideal skull.

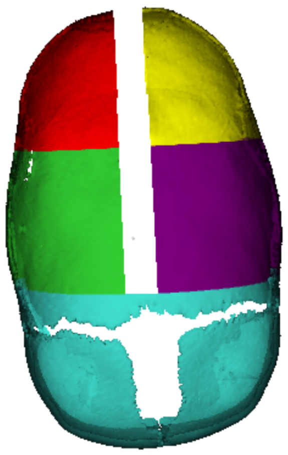

Figure 2.2: Five regions of a deformed skull, face pointing down.

2.1 Problem formulation.

We are given I, an ideal skull, and a collection $\mathcal{R}$ of disjoint contiguous subsets of a deformed skull, which we call regions. These are given in the form of triangulated surfaces, see figure 2.1 for an idea of what the ideal skull looks like, and figure 2.2 for an example the five regions of a deformed skull. The goal is to cut each of these regions in two with a straight line cut, and place the resulting pieces on the ideal skull. We now describe how we cut a region $R \in \mathcal{R}$. Two points x_1 and x_2 on the boundary of R determine a line c. Let $x_3 \in R$ denote a closest point on R to the midpoint $\frac{x_1+x_2}{2}$ of c. The normal n of R at x_3 together with c determine a plane H_c, which divides R in two. This division of R into two contiguous subsets of R is what we mean by cutting R into two pieces. By a cut we mean $H_c \cap R$, but we will refer to a cut by just c. See figure 2.3 for an example of a region cut into two pieces.

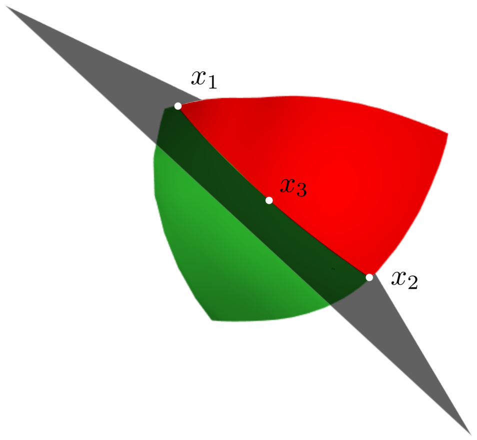

Figure 2.3: A region cut into two pieces, shown in green and red.

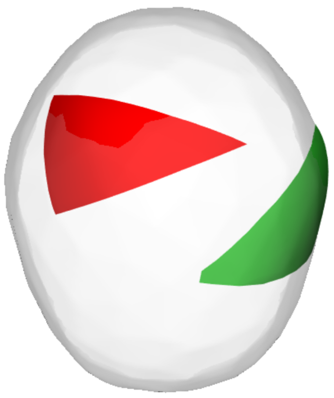

Figure 2.4: The pieces shown in figure 2.3, placed on an ideal skull.

Placing a piece p is defined as identifying the center point of p with the center point t of a face of I, and identifying the normals of the piece with the normal of I at t. By the center of a face we mean the centroid of that face, i.e., the arithmetic mean of the vertices defining said face. By center of p, we mean the nearest point on p to the arithmetic mean of all centers of faces of p. For an example of two pieces being placed on a skull, see figure 2.4. A piece is allowed to rotate in the plane defined by this normal, according to a set of prescribed angles Θ.

We generate multiple potential cuts for each region, and select at most one of these potential cuts per region in a solution. Each cut of a region yields two pieces. Let $\mathcal{P}$ be the set of possible resulting pieces, let Θ be the set of possible rotations, and let $\mathcal{T}$ be the set of possible placement locations, i.e., center points of faces in I. We introduce a binary variable y_c denoting whether we use a cut c. We also introduce a binary variable $x_{p,\theta,t}$ indicating whether piece $p \in \mathcal{P}$ is rotated with rotation $\theta \in \Theta$ and placed at $t \in T$. We also have a variable $x_{R_i,\theta,t}$, corresponding to the placement and rotation of uncut region $R_i \in \mathcal{R}$. We emphasize that we have two fundamentally different types of pieces that we can place, $R_i \in \mathcal{R}$, which correspond to uncut region pieces, and $p \in \mathcal{P}$ which is formed from cutting a region in $\mathcal{R}$ in two. We denote by $\mathcal{P}_i \subseteq \mathcal{P}$ the set of potential pieces formed from region R_i, $\mathcal{C}_i$ the set of cuts in R_i, and $c(p)$ the cut inducing piece p. We denote by $\mathcal{C}$ the union of all $\mathcal{C}_i$. Note that each cut induces two pieces, so there are two distinct pieces p, p' with $c(p) = c(p')$. Finally, we introduce a variable z_g for every point $g \in \mathcal{T}$, indicating whether g is covered by some piece. We now describe the constraints involved in our IP model.

A piece can be placed in at most one location $t \in \mathcal{T}$ with one rotation $\theta \in \Theta$

$$\sum_{\theta \in \Theta} \sum_{t \in T} x_{p,\theta,t} \leq 1, \forall p \in P \cup \mathcal{R}.$$

At most one cut is made per region

$$\sum_{c \in \mathcal{C}_i} y_c \leq 1, i \in [|\mathcal{R}|]^{1}.$$

A piece can be used only if its corresponding cut is made

$$\sum_{\theta \in \Theta} \sum_{t \in T} x_{p,\theta,t} \leq y_{c(p)}, \forall p \in P.$$

There are really two types of pieces, pieces $p \in \mathcal{P}$ formed from cutting a region, and region pieces in $\mathcal{R}$, which are uncut regions. Since we allow for the placement of region pieces, we need a constraint to ensure that we are not using a region and a piece formed by cutting said region. With the following constraint, we encode the fact that an uncut

[1] $[k]$ for $k \in \mathbb{N}$ denotes the set $\{1, 2, \ldots, k\}$.

region piece is never placed if it is cut to form other pieces:

$$\sum_{c \in \mathcal{C}_i} y_c + \sum_{\theta \in \Theta} \sum_{t \in \mathcal{T}} x_{R_i,\theta,t} \leq 1, i \in [|\mathcal{R}|].$$

On the one hand, if $\sum_{\theta \in \Theta} \sum_{t \in \mathcal{T}} x_{R_i,\theta,t}$ equals 1, signifying region piece R_i is placed, then all cut variables y_c of cuts $c \in \mathcal{C}_i$ are zero. On the other hand, if $\sum_{c \in \mathcal{C}_i} y_c$ equals 1, signifying the region R_i is cut, then all $x_{R_i,\theta,t}$ variables are zero.

We have a binary variable z_g for every $g \in \mathcal{T}$, indicating whether g is covered by a piece. A point on the ideal skull is covered if and only if a piece covers it, and at most one piece covers a point

$$\sum_{p \in \mathcal{P} \cup \mathcal{R}} \sum_{\theta \in \Theta} \sum_{t \in T} d_{p,\theta,t,g} x_{p,\theta,t} = z_g, \forall g \in \mathcal{T},$$

where $d_{p,\theta,t,g}$ is an indicator parameter for a piece p placed with θ at t covering point g. We describe how we compute $d_{p,\theta,t,g}$ in the next section, section 2.1.1.

To have some control over the proportion of ideal skull surface area the placed pieces cover, we associate with each point $g \in \mathcal{T}$ a weight a_g, equal to the area of the face on I containing g. We then have a constraint enforcing a lower bound γ on the area covered

$$\sum_{g \in \mathcal{T}} a_g z_g \geq \gamma.$$

Finally, $w_{p,\theta,t}$ is a non-negative real number, a cost, representing how poorly a piece p placed with rotation θ at location t fits. We want to minimize this cost w. We describe how w is computed in 2.1.1. Putting this all together, we obtain the following model

$$\min \sum_{p \in P} \sum_{\theta \in \Theta} \sum_{t \in T} w_{p,\theta,t} x_{p,\theta,t} \tag{2.1.1}$$

$$\sum_{\theta \in \Theta} \sum_{t \in T} x_{p,\theta,t} \leq 1, \forall p \in P \cup \mathcal{R} \tag{2.1.2}$$

$$\sum_{c \in \mathcal{C}_i} y_c \leq 1, i \in [|\mathcal{R}|] \tag{2.1.3}$$

$$\sum_{\theta \in \Theta} \sum_{t \in T} x_{p,r,t} \leq y_{c(p)}, \forall p \in P \tag{2.1.4}$$

$$\sum_{c \in \mathcal{C}_i} y_c \leq 1 - \sum_{\theta \in \Theta} \sum_{t \in \mathcal{T}} x_{R_i,\theta,t}, i \in [|\mathcal{R}|] \tag{2.1.5}$$

$$\sum_{p\in\mathcal{P}\cup\mathcal{R}}\sum_{\theta\in\Theta}\sum_{t\in T} d_{p,\theta,t,g}x_{p,\theta,t} = z_g, \forall g \in \mathcal{T} \tag{2.1.6}$$

$$\sum_{g\in\mathcal{T}} a_g z_g \geq \gamma \tag{2.1.7}$$

$$x_{p,\theta,t} \in \{0,1\}, \forall p \in \mathcal{P}\cup\mathcal{R}, \theta \in \Theta, t \in \mathcal{T} \tag{2.1.8}$$

$$y_c \in \{0,1\}, \forall c \in \mathcal{C} \tag{2.1.9}$$

$$z_g \in \{0,1\}, \forall g \in \mathcal{T}. \tag{2.1.10}$$

2.1.1 Parameters and variables.

The model described in 2.1 above comes with three types of parameters, namely $w_{p,\theta,t}$, $d_{p,\theta,t,g}$, and a_g. $w_{p,\theta,t}$ is the cost of placing piece p with rotation θ at location t, $d_{p,\theta,t,g}$ is a 0-1 parameter indicating whether a piece p with rotation θ placed at t covers a point g. Finally, a_g gives a weight to a point. Parameter calculations, as well as various figures throughout this thesis, are made possible by the Python library Visualization Toolkit (VTK) [14].

The cost function w for every piece p, rotation θ and placement t is calculated as follows. From the center point y of the skull, for each vertex of v of the triangulated ideal skull mesh I, generate a ray originating at y and intersecting v. If this ray intersects p placed with θ at t, say at a point t_v, then store the distance between v and t_v. The cost $w_{p,\theta,r}$ is the average of all these distances over all such rays, multiplied by the surface area of p. Since p is a triangulated mesh, the surface area of p is the sum of areas of triangles comprising p. One should think of $w_{p,\theta,t}$ as an estimate for the volume between the piece p and I. This cost function differs slightly from the one used in Yeung's model [17]; they used the average distance, not multiplying by the surface area of p . We made this change in cost function because if makes sense from a medical perspective to minimize volume between the ideal skull and placed pieces.

The parameter $d_{p,\theta,t,g}$ is computed similarly. For each point $t \in \mathcal{T}$, extend the normal n_t of I at t in both directions, forming a line l_t. If l_t intersects p placed with rotation θ at location t, then $d_{p,\theta,t,g}$ is set to 1, and 0 otherwise. If $d_{p,\theta,t,g} = 1$, we take this to mean that piece p with rotation θ at placed at t *covers* point g.

Finally, Yeung's model [17] attempts to cover vertices of the underlying ideal skull mesh. It is reasonable from the point of view of the application to cover the center points of faces instead, as covering the vertices on the bottom edge of the skull might not be very important, since we are interested in covering a certain amount of surface area. These are the points we attempt to cover in our model. Recall that the ideal skull mesh is a triangulated mesh. These triangles are not all uniform; they are smaller near the apex (the top) of the skull, and larger further from the apex, see figure 2.1. Hence, a constraint enforcing a certain number of points to be covered favours placements near the apex of the skull. Instead of covering a certain number of points on the ideal skull, it is more intuitive to cover an amount of surface area. A natural notion of area we can assign to each point on the ideal skull is the area of the mesh cell containing it. This is another reason for favouring the center of faces over vertices of the mesh: the notion of area is more natural. We compute this area a_g for every $g \in \mathcal{T}$, and include constraint 2.1.7

$$\sum_{g \in \mathcal{T}} a_g z_g \geq \gamma.$$

In Yeung's model [17], all points $g \in \mathcal{T}$ are weighted equally, i.e., $a_g = 1, \forall g \in \mathcal{T}$. With our modification, the points are weighted by the corresponding cell area.

We have yet to specify the number of regions, the number of cuts, the number of rotations, and the number of possible placement locations in 2.1. Choosing any of these to be large leads to a large model, which might have greater accuracy, but will take longer to solve. In practice, it makes sense to consider five regions, so $\mathcal{R} = \{R_1, R_2, R_3, R_4, R_5\}$ [12], as seen in figure 2.2. We choose to have approximately 100 cuts per region, i.e., $|\mathcal{C}_i| \approx 100$, $\Theta = \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$, and approximately 250 placement locations, i.e., $|\mathcal{T}| \approx 250$. The 2 points on the boundary inducing the cuts are chosen to be at least some distance apart, so that cuts are not trivial. We often found solutions to the model where rotating a piece p a small amount would decrease the volume between p and I. Now, Θ allows only for rotations in a certain plane, whereas these seemingly natural rotations were often not in this plane. Introducing rotations in other planes to allow for this potential improvement would lead to a very large model that might not be solvable in a reasonable amount of time. Hence, to address this issue, we take a local optimization step once a solution is produced, to try to rotate the placed pieces to fit better. We describe this next.

2.1.2 Kabsch's Algorithm

In this section, we describe how to improve the placement of a piece p in a given position. The idea is to take n points X on a piece p, and compute for each $x \in X$, the closest point y on I. This gives a set Y of n points on I. Seeking a rotation of of X minimizing the distance between X and Y is known as the *constrained orthogonal Procrustes problem*. More formally, in the constrained orthogonal Procrustes problem, one is given two $d \times n$ matrices P and Q. One wants to find a $d \times d$ rotation matrix U minimizing $\sum_{i=1}^{n} \|Uq_i - p_i\|^2$, where q_i and p_i is the ith column of Q and P, respectively, and $\|\cdot\|$ is the d-dimensional Euclidean norm. For a vector $v = (v_1, v_2, \ldots, v_d) \in \mathbb{R}^d$, the d-dimensional Eucliean norm is defined as $\|v\| := \sqrt{v_1^2 + v_2^2 + \cdots + v_d^2}$. In the name of the problem, "constrained" refers to the fact that U is a rotation matrix, i.e., an orthogonal matrix[2] constrained to have determinant 1. In our setting $d = 3$, and the matrices P and Q are the matrices obtained by writing the coordinates of points of X and Y as column vectors. Kabsch's algorithm (sometimes refered to as the Kabsch-Umeyama algorithm) is an algorithm for solving the constrained orthogonal Procrustes problem. It was first introduced in 1976 by Kabsch [8] [9], and then re-introduced by Umeyama in 1991 [16]. The original proof uses lagrange multipliers. A recent and elegant algebraic justification for the algorithm is given by Lawrence, Bernal, and Witzgall in [11]. We give a high level overview of their short argument.

In their paper, the authors first argue that minimizing $\sum_{i=1}^{n} \|Uq_i - p_i\|^2$ is equivalent to maximizing $\text{tr}(UQP^T)$, where $\text{tr}(M)$ is the trace of a matrix M, i.e., the sum of the diagonal entries of M. Since QP^T is fixed, the problem is now to find a matrix U maximizing $\text{tr}(UM)$. This is what Kabsch's algorithm (see below) does, using a singular value decomposition (SVD) of M. The singular value decomposition of a matrix M is a factorization of M into VSW^T, where S is diagonal, and V and W are orthogonal $d \times d$ matrices.

Algorithm 1 Kabsch-Umeyama.

```
Input: $d \times n$ matrices $P, Q$.
1: Compute $d \times d$ matrix $M = QP^T$.
2: Compute SVD of $M$, i.e., identify $d \times d$ matrices $V, S, W$,, so that $M = VSW^T$ in the SVD sense.
3: Set $s_1 = \ldots = s_{d-1} = 1$.
4: if $\det(VW) > 0$ then
5:     Let $s_d = 1$.
6: else
7:     Let $s_d = -1$.
8: Set $\tilde{S} = \text{diag}\{s_1, \ldots, s_d\}$.
9: return $d \times d$ rotation matrix $U = W\tilde{S}V^T$.
```

[2]An orthogonal matrix is a matrix whose columns are orthogonal and have unit norm.

The key to their proof is the following proposition.

Proposition 2.1.1. *If $D = \text{diag}\{\sigma_1, \ldots, \sigma_d\}, \sigma_i \geq 0, i \in [d]$, and W is a $d \times d$ orthogonal matrix, then*

1. $\text{tr}(WD) \leq \sum_{i=1}^{d} \sigma_j$.
2. *If B is a $d \times d$ orthogonal matrix and $S = B^T DB$, then $\text{tr}(WD) \leq \text{tr}(S)$.*
3. *If $\det(W) = -1$ and $\sigma_d \leq \sigma_i, i \in [d-1]$, then $\text{tr}(WD) \leq \sum_{i=1}^{d-1} \sigma_i - \sigma_d$.*

The majority of the work for proving this proposition lies in proving 3. Using the proposition, we have the following theorem, justifying Kabsch's algorithm,

Theorem 2.1.2. *Let M be a $d \times d$ matrix with singular value decomposition VSW^T, where V, S, and W are $d \times d$ matrices. If $\det(VW) > 0$, then $U = WV^T$ maximizes $\text{tr}(UM)$ over all $d \times d$ rotation matrices U. Otherwise, with $\tilde{S} = \text{diag}\{s_1, \ldots, s_d\}, s_1 = \ldots = s_{d-1} = 1, s_d = -1$, then $U = W\tilde{S}V^T$ maximizes $\text{tr}(UM)$ over all $d \times d$ rotation matrices U.*

We use Kabsch's algorithm on an optimal solution our IP model outputs. Ideally, one would run Kabsch's algorithm on all placements and rotations, and update w accordingly. However, this is challenging computationally, as a single Kabsch computation for a single placed piece takes more than one second, and we would have to do $|\Theta||\mathcal{T}| \sum_{i=1}^{5} |R_i||\mathcal{C}_i| \approx 500000$ such computations. Instead, we only run the algorithm on the solutions our IP model outputs. Experimentally, we observe that this yields a decrease in cost of the solution, while maintaining high coverage. However, we cannot prove this holds in general. We run this algorithm on the solutions produced by the IP model, and use the already implemented version in SciPy [6]. This leads to an average improvement of 13.4% in the objective value over all variants of our model analyzed in section 2.3, with some solutions seeing an improvement of over 30%.

2.2 Model variants

We produced different variants of the model, giving alternative perspectives on the problem. We discuss these here before presenting some experimental results in 2.3.

2.2.1 Overlapping pieces

At first, our model had no bound on the number of pieces that could cover a given point on the ideal skull. Intuitively, having more than two pieces overlapping does not increase the amount of points covered. One might think that the model would not favour such solutions. When looking at the solutions the model produced, however, we realized some solutions would not make sense in practice, namely ones where many pieces overlapped at a single point, see figure 2.5. Solutions where many pieces are stacked on top of one another do not resemble what is reasonable in practice, where one wants the end result to approximate a skull bone.

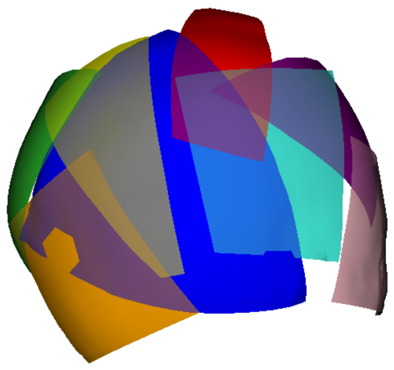
Figure 2.5: Example of a feasible solution with no restriction of overlap. Here, the blue, orange, and yellow pieces all overlap at a point. So do the blue, purple, red, and turquoise pieces. [3]

To rectify this, we introduced constraint 2.1.6

$$\sum_{p \in \mathcal{P} \cup \mathcal{R}} \sum_{\theta \in \Theta} \sum_{t \in T} d_{p,\theta,t,g} x_{p,\theta,t} = z_g, \forall g \in \mathcal{T}.$$

The above constraint enforces that every point covered by a piece is covered by exactly one piece, and that a point is counted as covered if and only if a piece covers it. The addition of constraint 2.1.6 makes the model harder to solve. Perhaps allowing for only one piece to cover a point is too restrictive, in the sense that hoping for such solutions

[3]Note that the colors of the pieces do not have any significance, they are purely there for ease of distinction between the pieces. This is the case for similar figures hereinafter.

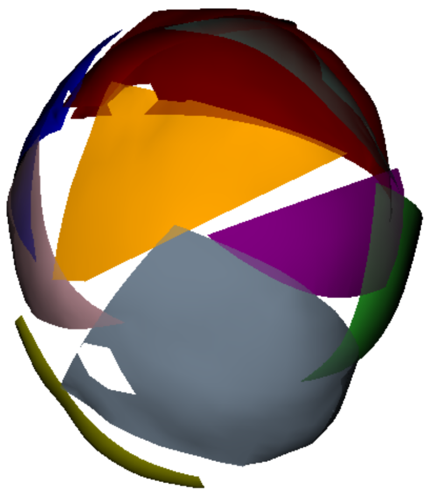
Figure 2.6: Example of solution with overlap 1.

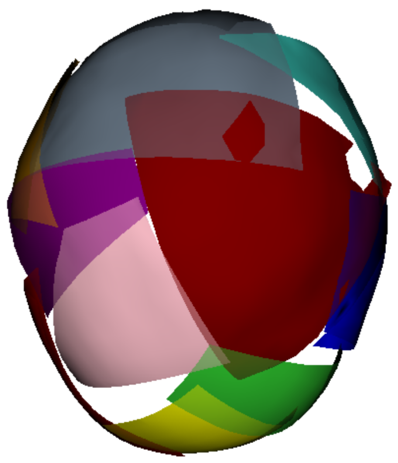
Figure 2.7: Example of solution with overlap 2.

while requiring a lot of area covered might not exist, as seen in our experimental results in section 2.3. Hence, we also consider the variant where at most two pieces can cover a given point. Either form of this overlap condition can be encoded with the following constraints

$$\sum_{p\in\mathcal{P}\cup\mathcal{R}}\sum_{\theta\in\Theta}\sum_{t\in T} d_{p,\theta,t,g}x_{p,\theta,t} \leq \omega z_g, \forall g \in \mathcal{T},$$
$$\sum_{p\in\mathcal{P}\cup\mathcal{R}}\sum_{\theta\in\Theta}\sum_{t\in T} d_{p,\theta,t,g}x_{p,\theta,t} \geq z_g, \forall g \in \mathcal{T},$$

where $\omega \in \{1, 2\}$ depending on how many pieces are allowed to overlap. Prior to introducing this constraint, pieces tended to be clumped near the apex of the skull. We hoped that such a constraint would lead to less clumping near the apex, which is indeed what occurred. See 2.6 and 2.7 for some contrasting examples to figure 2.5.

2.2.2 Disallowing central coverage.

In practice, one avoids cutting across the superior sagittal sinus (see figure 2.8) due to the amount of blood flow it allows for. Hence, the bone covering the sinus normally stays in place during surgery. The uncut rectangular strip in figure 2.2 is due to this. This rectangle

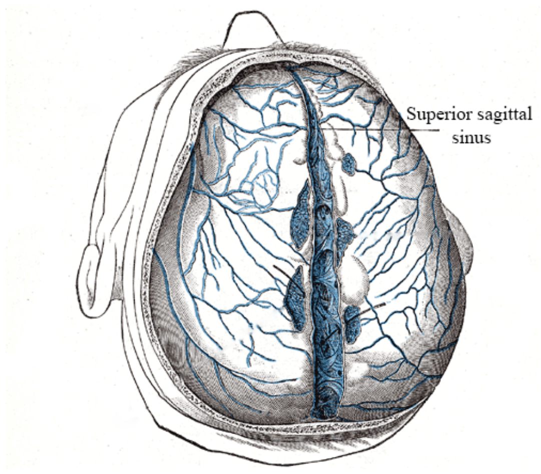

Figure 2.8: Superior sagittal sinus [3].

is exactly in the location of the sinus. It is therefore resonable to not allow pieces to cover this area of the skull in our model, which is precisely what the following modification does. If we denote by $\mathcal{S}$ the set of face center points in the region we do not want to cover, we simply add a constraint forcing these points to be uncovered.

$$\sum_{g \in \mathcal{S}} z_g \leq 0.$$

Figure 2.9 is an example where we force the superior sagittal sinus area to be uncovered.

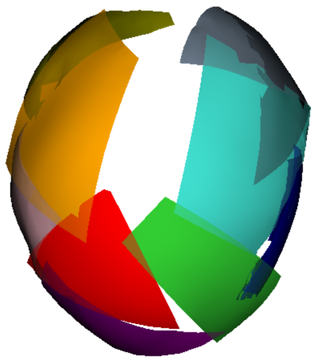

Figure 2.9: Example solution where no placement over the sagittal sinus is allowed.

2.3 Experimental results.

In this section we present the results of various computations of our model and five different skulls. A top down view of these skulls, with the face pointing down, is shown below.

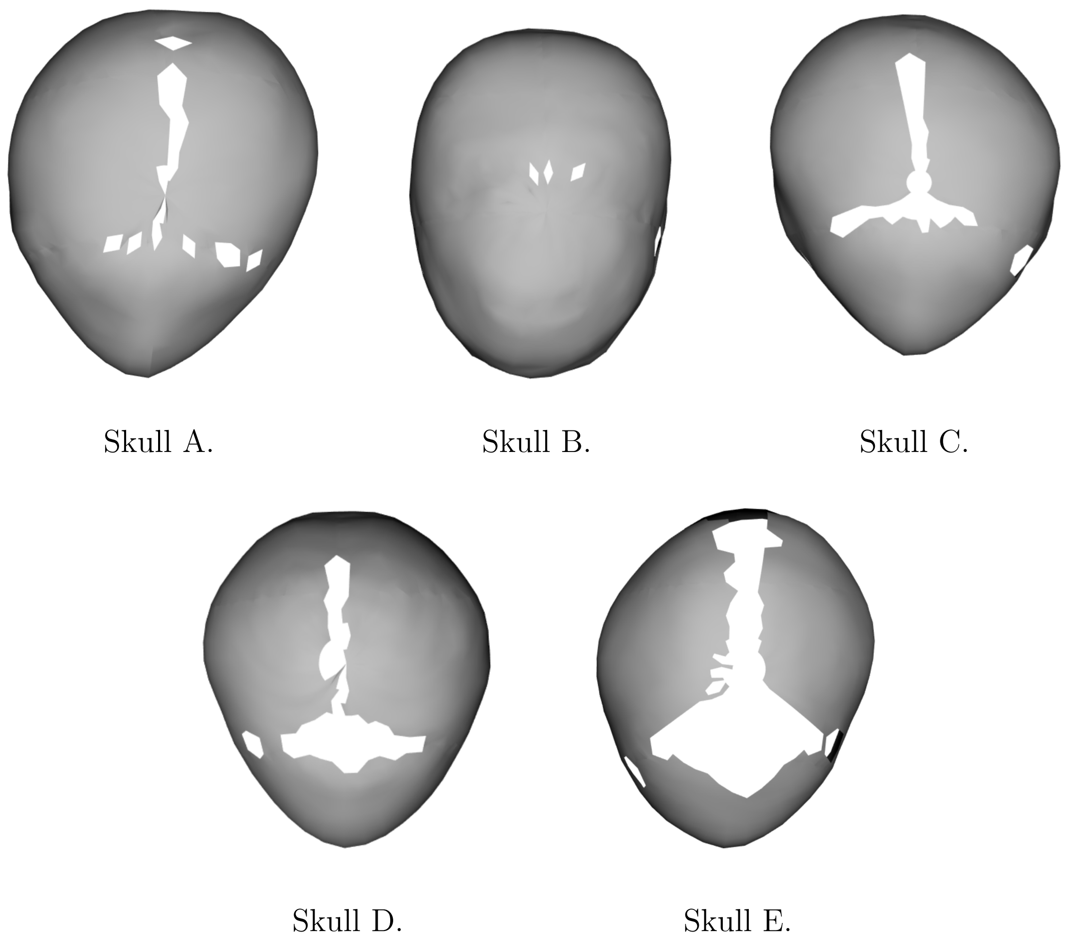

Figure 2.15: Five skulls

These skulls were obtained from real patients, and scaled isotropically, i.e., uniformly in all directions, to match the size of our ideal skull by matching aligning bandeaus. The data on which we base the analysis in this section is given in appendix A. Each model is allowed one wall-clock hour to run, i.e., 3600 seconds on our server, after which we select the best solution reached so far if no optimal solutions is reached. We make use of Gurobi's Python interface in solving our IPs [7].

2.3.1 Comparing overlap.

We compare solutions varying the overlap parameter ω for the five skulls, at 7 different levels of covered area, namely increments of 5% between 70% and 100%. We do this by choosing γ in constraint 2.1.7 accordingly. Since having $\omega = 2$ in our model is a relaxation of having $\omega = 1$, we expect solutions where $\omega = 2$ to have smaller objective values. We also expect the objective value to be monotone increasing with respect to the coverage requirement. Both these expectations are not necessarily true due to the use of Kabsch's algorithm. For instance, the Kabsch step could improve a solution with $\omega = 1$ more than a solution to $\omega = 2$, to the point of having lower objective value. Below is the data in table form, related to skull A. For tables for the other skulls, refer to appendix A.

Table 2.1: Skull A.

ω	Covered area (%)	Objective	Non-region pieces	Region pieces	Run time (s)
1	70	2507.575	7	0	475.37
1	75	3593.693	8	0	1133.25
1	80	4328.770	8	0	1854.34
1	85	3956.226	8	0	2015.37
1	90	6654.603	9	0	3600.00
2	70	2319.224	7	0	155.38
2	75	2700.144	8	0	142.25
2	80	3514.421	9	0	204.26
2	85	4082.187	9	0	354.17
2	90	3852.434	9	0	387.53
2	95	6573.883	10	0	725.50
2	100	8207.805	10	0	3186.44

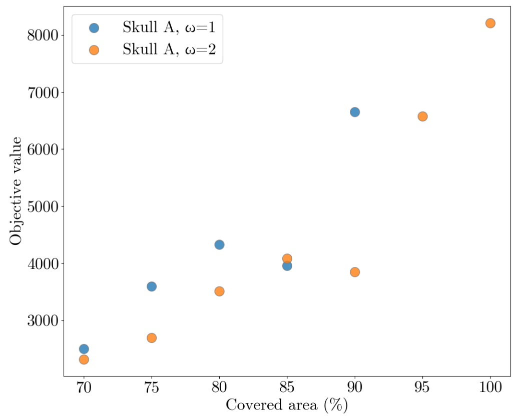

Figure 2.16: Skull A, area vs. objective.

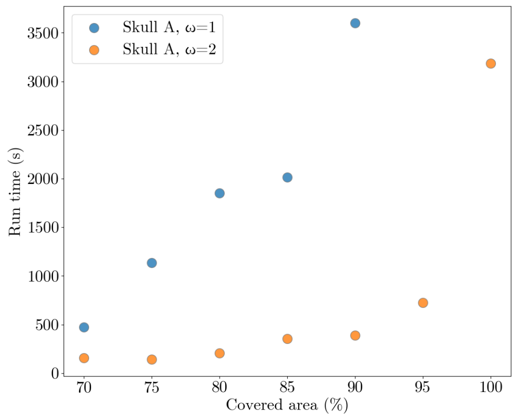

Figure 2.17: Skull A run time.

In figure 2.16, we see that the objective value for $\omega = 2$ is almost always lower than that of $\omega = 1$. Interestingly, this is not the case for the area constraint at 85%. Worth noting is that there is a decrease in objective value for $\omega = 2$ between 85% and 90%. The absence of a blue data point in figure 2.17 for 95% and 100% signifies the model is infeasible for $\omega = 1$ with covered area 95% and 100%. Moreover, the solution for 90% was not necessarily optimal, but the best solution found after an hour. This could potentially explain the sudden increase in objective value. Another observation to be made from figure 2.17 is that the run time for $\omega = 1$ is much greater than that of $\omega = 2$, indicating that, as expected, the model is harder to solve when $\omega = 1$. With $\omega = 2$, all but the model with 100% area covered take less than 1000s for Gurobi to solve.

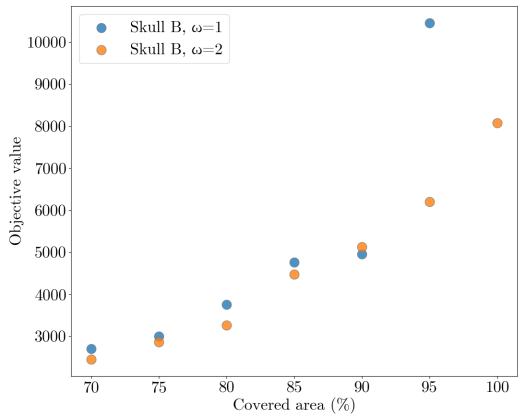

Figure 2.18: Skull B, area vs. objective.

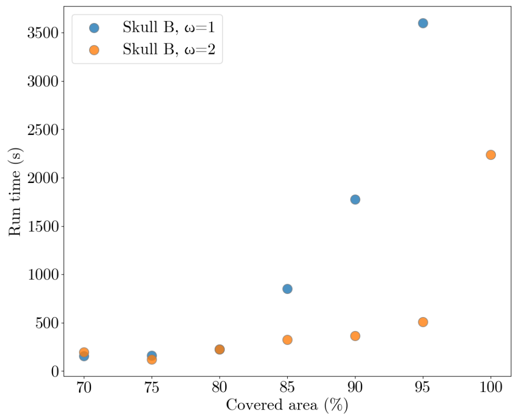

Figure 2.19: Skull B run time.

In figure 2.18, we see that the objective value for $\omega = 2$ is almost always lower than that of $\omega = 1$. This is the case for all covered area percentages except 90%. Compared to Skull A, varying ω makes a smaller difference here. Looking at figure 2.19, the solution for $\omega = 1$ at 95% was not necessarily optimal, but the best one found after an hour, and the model was infeasible for coverage of 100%. Similarly to skull A, the run time for $\omega = 1$ is greater than that of $\omega = 2$, at least for the higher area coverage requirement cases. With $\omega = 2$, all but the model with 100% area covered take less than 1000s for Gurobi to solve.

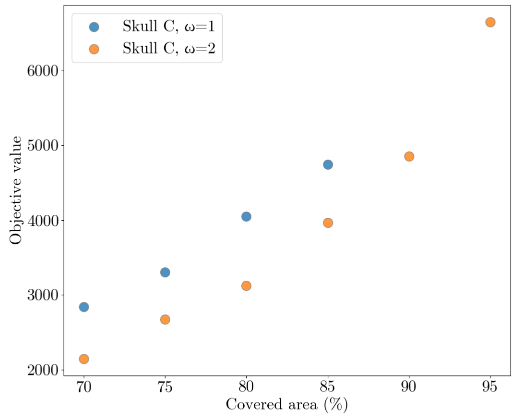

Figure 2.20: Skull C, area vs. objective.

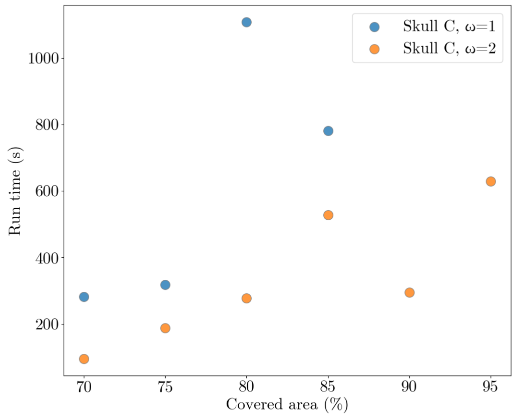

Figure 2.21: Skull C run time.

In figure 2.20, we see that the objective value for $\omega = 2$ is always lower than that of $\omega = 1$. Here, the model was infeasible for coverage greater than 85% with $\omega = 1$, and for $\omega = 2$ with coverage at 100%. Similar observations about run time as for the previous two skulls can be made. To get an idea of what these solutions look like, see figures 2.22 and 2.23. Pieces might still slightly overlap when $\omega = 1$, due to the discrete nature of the IP model, but nowhere near as much as when $\omega = 2$.

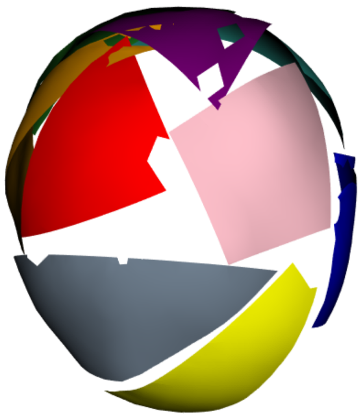

Figure 2.22: Skull C, $\omega = 1$.

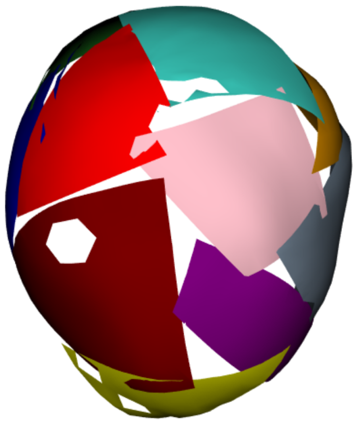

Figure 2.23: Skull C, $\omega = 2$.

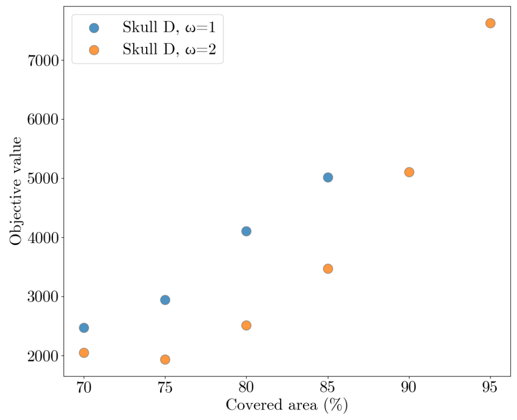

Figure 2.24: Skull D, area vs. objective.

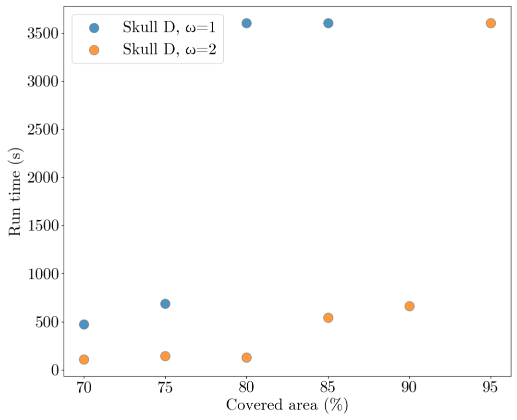

Figure 2.25: Skull D run time.

The situation for skull D, as seen in figure 2.24, is similar to that of skull C, probably due to the similarity of the skull shapes, as seen in 2.15. The values are similar, and again the model was infeasible for coverage greater than 85% with $\omega = 1$, and for $\omega = 2$ with coverage at 100%. For skull D, the solutions for $\omega = 1$ when coverage is at 80% and 85% were the best found after an hour. The same holds for the solution for $\omega = 2$ at 95%.

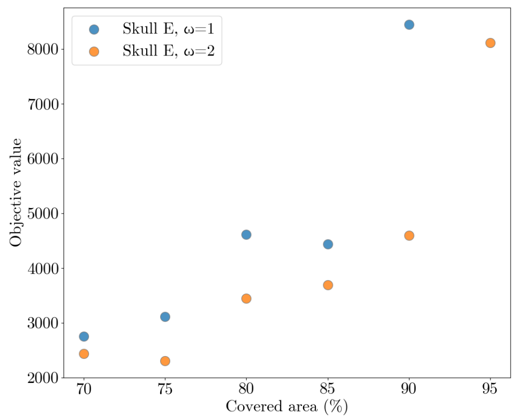

Figure 2.26: Skull E, area vs. objective.

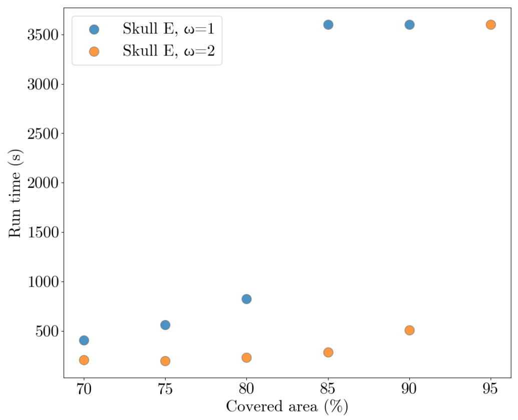

Figure 2.27: Skull E run time.

For skull E, the model is infeasible with $\omega = 1$ for 95% and 100% coverage, and with $\omega = 2$ for 100% coverage. Moreover, the solutions with $\omega = 1$ for 85% and 90% were the best found after an hour, and not necessarily optimal. The same is true for $\omega = 2$ at 95%.

Overall, the results align reasonably well with our expectations, in the sense that solutions with $\omega = 1$ have larger objective values than solutions with $\omega = 2$, when fixing a skull and coverage percentage. Additionally, the objective value almost always increases with covered areas. Unfortunately, 2 of the 5 skulls fail to have solutions with $\omega = 1$ for 90% and above, and 4 of the 5 skulls fail to have solutions with $\omega = 1$ for 95% and above, In contrast, when letting $\omega = 2$, all skulls allow for a solution at 95%, two of which also allow for a solution at 100%. Many of the solutions found at the higher coverage requirements were not necessarily optimal, and more time than an hour was needed to solve these to optimality. However, all but one solution that were found to be optimal for $\omega = 2$ were found in less than 1000s, which is a very reasonable amount of time. It thus seems reasonable to use the $\omega = 2$ variant of the model.

2.3.2 Disallowing middle placements.

In this section, similarly to the previous section, we compare solutions with disallowed middle placements, varying the overlap parameter ω for the five skulls, at same 7 levels of covered area percentages. We note that the percentage of the area is the percentage of the allowed placements. Again, we expect solutions where $\omega = 2$ to have smaller objective values and the objective value to be monotone increasing with respect to the coverage requirement. We expect the models in this section to be easier and quicker to solve, since we have decreased the size of the model.

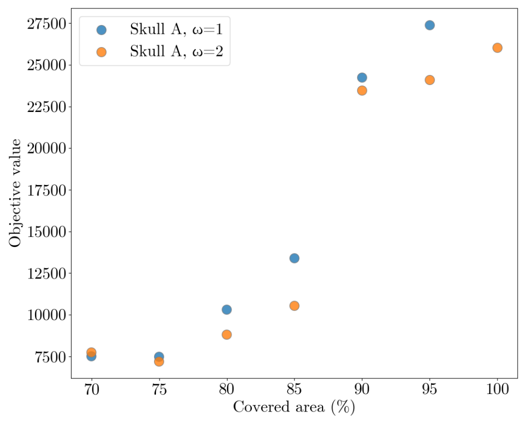

Figure 2.28: Skull A, area vs. objective.

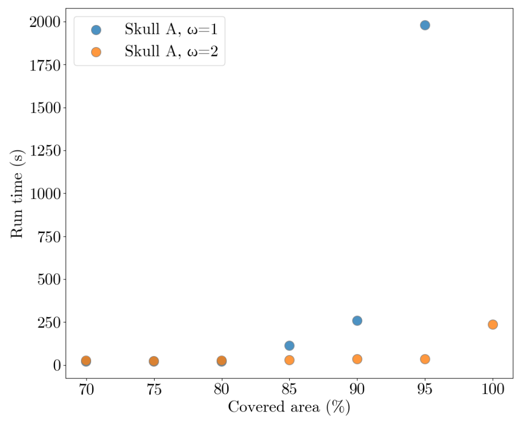

Figure 2.29: Skull A run time.

In figure 2.28, we see that the objective value for $\omega = 2$ is always lower than that of $\omega = 1$, except at 70% . Optimal solutions were found in all cases except for $\omega = 1$ at 100%, as seen in 2.29. Remarkably, all solutions with $\omega = 2$. were found in under 500 seconds.

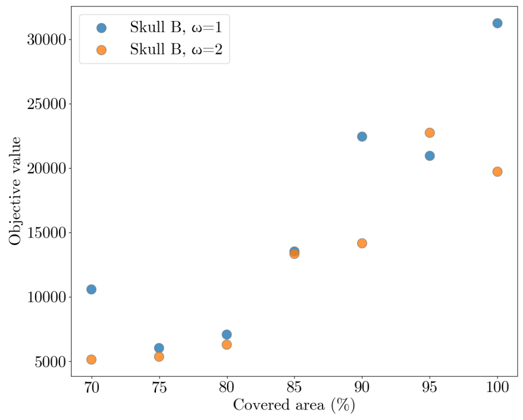

Figure 2.30: Skull B, area vs. objective.

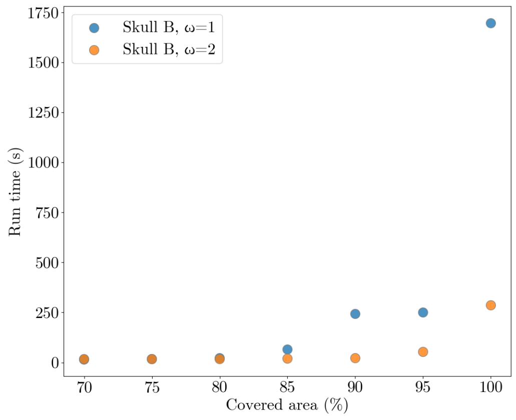

Figure 2.31: Skull B run time.

In figure 2.30, we see that the objective values agree roughly with what we would expect, except at 95%, and except for the decrease in objective value. Optimal solutions were found in all cases, as seen in 2.31. Again, as the case for skull A, all solutions with $\omega = 2$. were found in under 500 seconds.

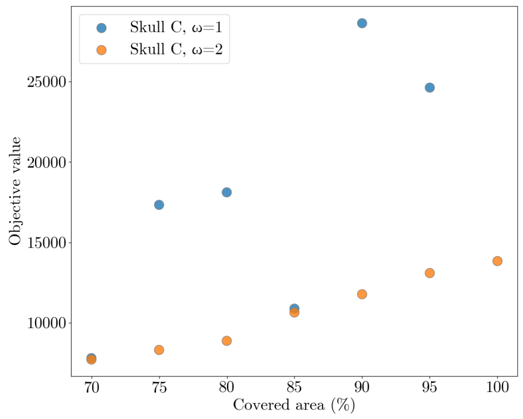

Figure 2.32: Skull C, area vs. objective.

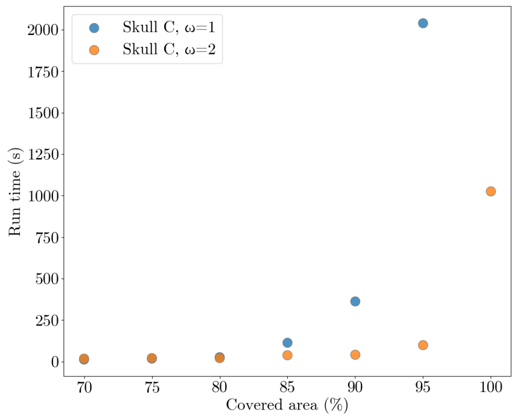

Figure 2.33: Skull C run time.

In figure 2.32, we see that the objective value for $\omega = 2$ is significantly lower than that of $\omega = 1$ for four coverage requirements, namely 75%, 80%, 90%, and 95%. The variation

in objective value for $\omega = 2$ is very smooth, in contrast with other skulls. Optimal solutions were found in all cases except for $\omega = 1$ at 100%, as seen in 2.33.

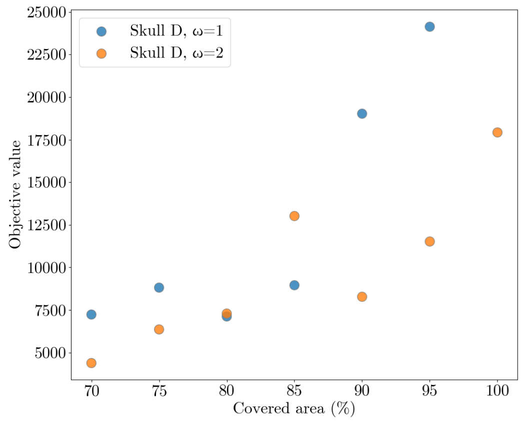

Figure 2.34: Skull D, area vs. objective.

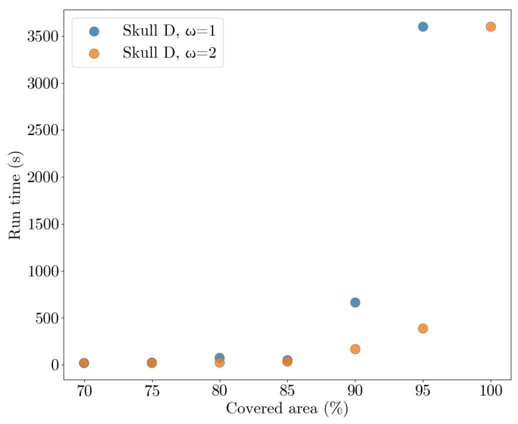

Figure 2.35: Skull D run time.

In figure 2.34, we see that the objective value for $\omega = 2$ is significantly lower than that of $\omega = 1$ for four coverage requirements, namely 75%, 80%, 90%, and 95%. The variation in objective value for $\omega = 2$ is very smooth, in contrast with other skulls. Optimal solutions were found in all cases except for $\omega = 1$ at 100%, as seen in 2.33. Again, as for skulls A, B, and C, all optimal solutions were found in much less than 1000s.

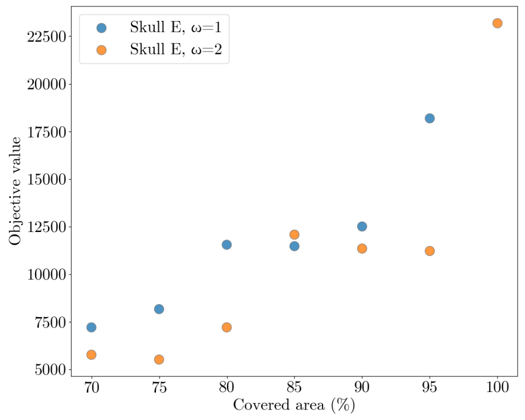

Figure 2.36: Skull E, area vs. objective.

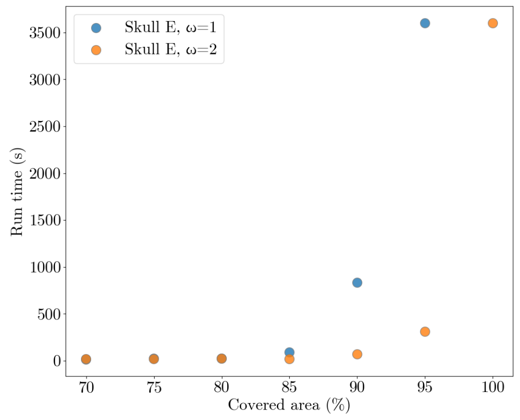

Figure 2.37: Skull E run time.

Similarly to the model variant where we allow middle placements, the results here align reasonably well with our expectations, in the sense that solutions with $\omega = 1$ have larger objective values than solutions with $\omega = 2$, when fixing a skull and coverage percentage. Additionally, the objective value almost always increases with covered areas. In contrast with the case where middle placements were allowed, solutions for all coverage requirements were found for $\omega = 2$. Moreover, 3 out of 5 of these solutions were optimal. Even for $\omega = 1$, a greater number of feasible solutions were found. The objective is interestingly 2 to 3 times higher for all skulls when disallowing for the middle placements. The run time for both values of ω were most times below 1000s, and many times even below 500s. Many times, the run times were twice as fast as the corresponding model where middle placements were allowed. This suggests that the model when disallowing middle placements is easier to solve, which is reasonable, since we have limited the possible placement locations.

As seen in our experiments, there is a trade-off between low objective value and high coverage; trivially, when no coverage is required, the empty solution is optimal. Of course, this is not a solution we are interested in. Among all our variants, the price paid in the objective for increased coverage is the smallest in the disallowed middle setting with $\omega = 2$. In this setting, the solutions to the 2 instances where the objective increased drastically at 100 can most likely be improved given more time, as these solutions were not guaranteed optimal. We believe that this variant of the model could make the most sense in practice. Naturally, the surgeon performing on operation always has the final say when performing a surgery. A surgeon could also potentially consider a few different solutions at different coverage levels, and draw inspiration from these.

2.4 Conclusion

To conclude, we gave an IP formulation to solve a problem inspired by cranial vault remodeling in 2.1. We described a few variants of the model in 2.2, and ran these variants on 5 skulls, analyzing the results in 2.3. Our main contribution is that we show it is possible to solve such a problem, even without pre-cut pieces, in most cases within the reasonable time of an hour, and many times much quicker. In practice, one might have weeks between taking a scan of a patient, and surgery. Hence, it is sensible to run a model of the same kindred as ours in practice; one might even try to make the model larger. Moreover, the problem was easier to solve when allowing for pieces to overlap slightly, and when not allowing for placements of pieces to cover the sagittal sinus.

Many potential improvements and questions remain. If one has access to more computational resources, does increasing the number of cuts, rotations, and placement locations significantly improve the quality of optimal solutions? Does precomputing the local Kabsch optimization for all variables lead to significant improvement? Alternatively, and less costly, one could potentially run the local Kabsch optimization on a few near optimal solutions and select the best one. We do not have any theoretical guarantees for the Kabsch step, but can one formulate guarantees for some set of surfaces? Our entire analysis uses one ideal skull, but it might make sense to consider different ideal skull depending on the type of craniosynostosis. Our model only allows for at most one cut per region. Does a model where more cuts per regions are allowed change things?

Another way of tackling the cranial vault remodeling problem could be to adapt the already existing dynamic programming approach given in [4]. This is yet to be done.

Our initial goal with this model was for our algorithm to be used as inspiration by surgeons working on craniosynostosis, but even if this does not happen, the objective value of our model could be used as some form of severity measure for craniosynostosis.

Chapter 3

Approximation and the Local Ratio.

In the Curve Reshaping Problem with Rearrangement, introduced by [5], one is given an ideal piecewise-linear curve g, and a deformed piecewise-linear curve f representing the shape of the infants bandeau. One wants to find subsegments of f, and to match these to subsegments of g, minimizing a cost function incorporating how good this matching is. This problem was shown to be NP-hard using a reduction from the 3-Partition problem, even when cut locations are prescribed. Nonetheless, Drygala, using the local ratio technique due to Bar-Yehuda et. al., gave a half approximation to the problem arising from a reduction to a scheduling problem [1][5]. In this section, we attempt to take these local ratio analyses one step further, to get closer to something that resembles reshaping the entirety of the cranial vault. More specifically, we study the naturally arising analogue of the scheduling problem, now in the plane instead of on the real line.

3.1 Local Ratio

This section is based on Bar-Yehuda's work on the local ratio technique, and more specifically on the work relating to approximating resource allocation and scheduling problems [1]. We give a summary of the one dimensional scheduling, before introducing our contribution of scheduling in the plane. Suppose we have some maximization problem whose solutions $x \in \mathbb{R}^n$ must satisfy a set of feasibility constraints F. A solution x has value $p \cdot x$, where $p \in \mathbb{R}^n$ is a profit vector. We say that x^* is optimal if it maximizes $p \cdot x$ among all feasible $x \in \mathbb{R}^n$. Sometimes one can not hope to obtain an optimal solution, but only an approximation of an optimal solution. A feasible solution x is an r-approximation if $p \cdot x \geq rp \cdot x^*$, where x^* is an optimal solution, and $r \in (0, 1]$. Note that this setting is in terms of maximizing an objective since the problems we study in this chapter are of such form; however, everything works similarly in the minimization setting. The result after which this section is named is the following.

Theorem 3.1.1. *(Local Ratio) Let F be a set of constraints, p_1, p_2, p be profit vectors satisfying $p = p_1 + p_2$. Then, if x is an r-approximation with respect to both (F, p_1) and (F, p_2), then x is an r-approximation with respect to (F, p).*

We include the proof since it is extremely short and provides some intuition.

Proof. Let $p = p_1 + p_2$ and consider x^*, x_1^*, x_2^* optimal solutions to (F, p), $(F, p_1), (F, p_2)$ respectively. We have by linearity $p \cdot x = (p_1 + p_2) \cdot x \geq r p_1 \cdot x_1^* + r p_2 \cdot x_2^* \geq r(p_1 \cdot x^* + p_2 \cdot x^*) = r \cdot p \cdot x^*$. □

At an intuitive level, an outline on how to use this technique to solve problems is as follows. Start with an empty solution, and search for a decomposition of p into $p_1 + p_2$, such that p_1 is "easy to handle" in some sense. Then, recurse on (F, p_2); assuming a good approximation for (F, p_2), we would like it to be a good approximation for (F, p). By the Local Ratio theorem, that would involve showing that the recursive call on (F, p_2) yields a good approximation to (F, p_1).

The authors introduce the following general framework to deal with resource allocation and scheduling problems: we are given activities $\mathcal{A}_1, \ldots, \mathcal{A}_m$, each being a set of time intervals. If an interval $I \in \mathcal{A}_j$, we say I is an instance of activity j. Specifically, an instance I consists of a time interval $[s(I), e(I))$, with $0 \leq s(I) \leq e(I)$. Every instance I has an associated profit $p(I) \in \mathbb{R}$, and a weight $w(I) \in [0, 1]$. We wish to schedule these instances to form a feasible schedule $\mathcal{S}$. A feasible schedule $\mathcal{S}$ is a collection of instances such that at most one instance of each activity is present, and for all time t the sum of the widths of instances scheduled at t is at most 1. We introduce next a very natural problem which falls into this framework: throughput maximization.

3.1.1 Throughput Maximization

In the throughput maximization problem, we simply seek to maximize the total profit of scheduled instances $p(\mathcal{S}) := \sum_{I \in \mathcal{S}} p(I)$. Below is a linear programming formulation of the problem. We introduce a variable x_I for each $I \in \cup_{j=1}^{m} \mathcal{A}_j$.

$$\max \sum_{I} p(I) x_I \tag{3.1.1}$$

$$s.t. \sum_{I: s(I) \le t \le e(I)} w(I) x_I \le 1, \forall t \in \mathbb{R}^+ \tag{3.1.2}$$

$$\sum_{I \in \mathcal{A}_j} x_i \le 1, \forall j \in [m] \tag{3.1.3}$$

$$x_I \in \{0,1\}, \forall I \in \mathcal{A}_j, \forall j \in [m]. \tag{3.1.4}$$

Here, constraint 3.1.2 ensures that the sum of the widths of instances schedule at t is at most 1. This can be encoded using a finite number of constraints, since the points $t \in \mathbb{R}^+$ that matter are just start and end times of instances. In other words, we can partition the subset of positive reals from the earliest start time to the latest end time into a finite number of intervals so that no interval contains a start or end time in its interior. Constraint 3.1.3 ensures at most one instance is chosen from every activity. To solve this problem, the authors propose the following algorithm.

Algorithm 2 Local Ratio: Choosing $\tilde{I}$ with earliest end time.

```
Input: $F, p, \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m$.
1: Delete all instances with non-positive profit.
2: if no instances remain then
3:     return $\mathcal{S} = \emptyset$.
4: Choose $\tilde{I} \in \cup_{j=1}^{m} \mathcal{A}_j$ with $e(\tilde{I})$ minimized.
5: Decompose $p$ by $p = p_1 + p_2$. It is enough to define $p_1$, which we do as follows
```

$$p_1(I) := p(\tilde{I}) \cdot \begin{cases} 1 & I \in \mathcal{A}(\tilde{I}) \\ \alpha w(I) & I \in \mathcal{I}(\tilde{I}) \\ 0 & \text{otherwise,} \end{cases}$$

where $\mathcal{A}(\tilde{I})$ is the activity to which $\tilde{I}$ belongs, and $\mathcal{I}(\tilde{I})$ the set of instances intersecting $\tilde{I}$ from other activities.

```
 6: Solve the problem recursively on input $(F, p_2)$, and return $\mathcal{S}'$.
 7: if $\mathcal{S}' \cup \{\tilde{I}\}$ is a feasible schedule then
 8:     return $\mathcal{S} := \mathcal{S}' \cup \{\tilde{I}\}$
 9: else
10:     return $\mathcal{S} := S'$
```

A feasible schedule $\mathcal{S}$ is said to be I*-maximal* if $I \subseteq \mathcal{S}$ or $\mathcal{S} \cup \{I\}$ is infeasible. The authors [1] introduced algorithm 2 to solve this problem approximately. We have the following two propositions pertaining to algorithm 2.

Proposition 3.1.2. *If for all profit functions $p = p_1 + p_2$ with $p_2(\tilde{I}) = 0$, every $\tilde{I}$-maximal schedule is an r-approximation with respect to p_1. Then the schedule $\mathcal{S}$ returned by algorithm 2 is an r-approximation.*

Denote by $w_{\max}$ and $w_{\min}$ the maximum and minimum width of all instances in the input respectively. Hereinafter, α is a parameter in the interval $[0, 1]$ that we choose in order to get a good approximation ratio.

Proposition 3.1.3. *Every $\tilde{I}$-maximal schedule is an r-approximation with respect to p_1, where*

$$r = \frac{\min\{1, \alpha \max\{w_{\min}, 1 - w_{\max}\}\}}{1 + \alpha}.$$

Next, we reproduce the proofs of these two propositions due to Bar-Yehuda et. al., as our arguments later on are very similar. As a consequence of these two propositions, if we show that every $\tilde{I}$-maximal schedule is an r-approximation, then the approximation ratio of the algorithm is at least r.

Proof of 3.1.2. Deleting instances with non-positive profit does not change the optimal value. It thus suffices to prove $\mathcal{S}$ is optimal with respect to the remaining instances. We proceed by induction on the number of recursive calls. For the base case, at the bottom of the recursion, there are no instances left. Then $\mathcal{S}$ is optimal, so indeed an r-approximation. For the induction step, suppose $\mathcal{S}'$ is an r-approximation with respect to p_2. Then, $\mathcal{S}$ is an r-approximation with respect to p_2, since $p_2(\tilde{I}) = 0$ and $\mathcal{S} \subseteq \mathcal{S}' \cup \{\tilde{I}\}$. By definition, $\mathcal{S}$ is $\tilde{I}$-maximal, so $\mathcal{S}$ is an r-approximation with respect to p_1 by assumption. By applying the local ratio theorem, we conclude $\mathcal{S}$ is an r-approximation with respect to p. □

Worth noting is that in the above proof we do not use any structure of the problem instance, or the choice of $\tilde{I}$. This makes adapting this proof, as we do later on, quite simple.

Proof of 3.1.3. We bound the optimum p_1 profit b_{opt} from above, and bound the profit of an $\tilde{I}$-maximal schedule $b_{\max}$ from below (where we normalize the profit by $p_1(\tilde{I})$). Then $\frac{b_{\max}}{b_{\text{opt}}}$ is a lower bound on the approximation ratio of the algorithm. We proceed to bound b_{opt}. Consider an optimal schedule $\mathcal{S}$. By definition of p_1, only instances in $\mathcal{A}(\tilde{I})$ or $\mathcal{I}(\tilde{I})$ contribute to the p_1-profit of $\mathcal{S}$. Since $\tilde{I}$ is chosen to have $e(\tilde{I})$ minimized, all instances I in $\mathcal{I}(\tilde{I})$ have $s(I) < e(\tilde{I}) \le e(I)$. Hence, all instances in $\mathcal{S} \cap \mathcal{I}(\tilde{I})$ intersect, and thus have total width at most 1 by feasibility. Hence, at most $\alpha p_1(\tilde{I})$ profit comes from such instances. Any feasible schedule contains at most one instance of $\mathcal{A}(\tilde{I})$. Thus, $b_{\text{opt}} \le 1+\alpha$.

Consider now a $\tilde{I}$-maximal schedule $\mathcal{S}$. $\mathcal{S}$ contains an instance in $\mathcal{A}(\tilde{I})$ or a collection of instances $X \subseteq \mathcal{I}(\tilde{I})$ such that $X \cup \{\tilde{I}\}$ is infeasible. The width $w(X)$ of instances in X is at least $1-w(\tilde{I}) \ge 1-w_{\max}$. We also have that $w(X) \ge w_{\min}$. Thus, $b_{\max} \ge \min\{1, \alpha \max\{1-w_{\max}, w_{\min}\}\}$. We conclude that every $\tilde{I}$-maximal schedule is an r-approximation. $\square$

Note that our choice as $\tilde{I}$ is key in getting the bound on b_{opt}, where the additive factor of α comes from the fact that all intervals intersecting $\tilde{I}$ intersect. The bound on $b_{\max}$ does not use this choice. From these observations, we have the following corollary.

Corollary 3.1.4. *If $\tilde{I}$ is chosen such that there exists a set of k points X contained in $\tilde{I}$ such that all $I \in \mathcal{I}(\tilde{I})$ contain some $x \in X$, then the algorithm has an approximation ratio of at least*

$$r = \frac{\min\{1, \alpha \max\{w_{\min}, 1-w_{\max}\}\}}{1+k\alpha}.$$

We apply the corollary in the following example, where we choose $\tilde{I}$ of minimal length. Though choosing $\tilde{I}$ in such a way does not improve the performance guarantee of the algorithm, it demonstrates how we arrive at a certain approximation factor.

Example 3.1.5 (Smallest interval)**.** In step 4 of the previous algorithm, instead of choosing $\tilde{I}$ with $e(\tilde{I})$ minimized, suppose we choose $\tilde{I}$ with $|e(\tilde{I}) - s(\tilde{I})|$ minimized. We use the same decomposition of p into p_1 and p_2. Any $I \in \mathcal{I}(\tilde{I})$ contains $s(\tilde{I})$ or $e(\tilde{I})$. By 3.1.4, this algorithm has an approximation ratio of

$$r = \frac{\min\{1, \alpha \max\{w_{\min}, 1-w_{\max}\}\}}{1+2\alpha}.$$

3.1.2 Craniosynostosis and throughput maximization.

The relevance of throughput maximization and the local ratio to craniosynostosis is due to Drygala [5]. They study deformities in the bandeau. In their thesis, they study the so called Curve Reshaping Problem with Rearrangement and Pre-cut Segments (CRPRPS). In this problem, one is given an ideal and a deformed curve, representing the ideal and deformed bandeau. The deformed curve is cut into segments. The problem is to place these segments onto the ideal curve so as to maximize the profit (some measure of fit and covereage) of placing a segment at a certain location. Drygala provides a reduction from the CRPRPS to the throughput maximization problem. At a high level, each segment J becomes an activity $\mathcal{A}_J$, and an instance I of activity $\mathcal{A}_J$ is a placement of segment J at a location on the ideal curve. The width of each instance is 1. This reduction thus gives an r-approximation to CRPRPS, where

$$r = \frac{\min\{1, \alpha \max\{w_{\min}, 1 - w_{\max}\}\}}{1+\alpha} = \frac{\min\{1, \alpha\}}{1+\alpha}.$$

Since α is a parameter to be set, we can choose α to maximize r, giving $r = \frac{1}{2}$ when $\alpha = 1$.

When considering cranial vault reshaping, it no longer make sense to consider ideal and deformed curves, and pieces as segments. It no longer makes sense to talk about scheduling intervals on the real line. Instead, we generalize the throughput maximization idea to scheduling two dimensional objects in the plane.

3.2 In the plane.

In the following, we reuse terminology from the one dimensional setting. We are given activities $\mathcal{A}_1, \dots, \mathcal{A}_m$, each consisting of instances. Instances are two dimensional objects, such as polygons, placed at a certain location in the plane. A feasible schedule schedule is a collection of instances where at most one instance is selected from each activity, and for every point t in the $\mathbb{R}^2$, the sum of widths of instances covering t is at most 1. We illustrate the idea with the example below.

Example 3.2.1 (Axis parallel squares.)**.** An instance I is specified by

1. The center point of the square I, and the side length $s(I)$ of I.

2. The amount $w(I)$ of resource used, where $0 \le w(I) \le 1$.

3. A profit $p(I)$ collected when I is scheduled.

In step 1. of Algorithm 1, if we now choose a square $\tilde{I}$ with $s(\tilde{I})$ minimized, then we obtain an approximation ratio of

$$r = \frac{\min\{1, \alpha \max\{w_{\min}, 1 - w_{\max}\}\}}{1 + 4\alpha}.$$

This follows by an application of corollary 3.1.4, and the observation that any instance $I \in \mathcal{I}(\tilde{I})$ contains one of the four corners of $\tilde{I}$. If all widths are 1, taking $\alpha = 1$, maximizing r, we have that $r = \frac{1}{5}$.

Remark 3.2.2. In general, if all widths are 1, $r = \frac{\min\{1,\alpha \max\{w_{\min},1-w_{\max}\}\}}{1+k\alpha}$. is maximized by letting $\alpha = 1$, and so $r = \frac{1}{k+1}$.

3.2.1 Overlap number and kissing number.

So far, we have observed that a key detail in how we obtain bounds on our approximation ratio r relies on bounding the denominator using some structure of the set of instances we have, namely how instances overlap geometrically. In an attempt to make this formal, we have the following definition.

Definition 3.2.3. (Overlap number.) Let $\mathcal{U}$ be a collection of sets in $\mathbb{R}^2$ and $I \in \mathcal{U}$. The overlap number of I, denoted $\omega^{\circ}_{\mathcal{U}}(I)$, is the largest collection $\mathcal{T} \subseteq \mathcal{U}$ of pairwise disjoint sets, which all intersect the interior I° of I.

We write $\omega(I)$ for $\omega_{\mathcal{U}}(I)$ when $\mathcal{U}$ is clear from context. Below are some examples illustrating this definition.

Example 3.2.4. If $\mathcal{U}$ is a collection of intervals of $\mathbb{R}$, and $I \in \mathcal{U}$ is chosen with the smallest right endpoint, then all sets in $\mathcal{I}(I)$ intersect, giving $\omega^{\circ}(I) \leq 1$.

Example 3.2.5. If $\mathcal{U}$ is a collection of closed intervals of $\mathbb{R}$, and I is chosen to have minimal length, then $\omega^{\circ}(I) \leq 2$, as for any three intervals in $\mathcal{I}(I)$, two intersect the same endpoint.

Example 3.2.6. If $\mathcal{U}$ is the collection of axis parallel squares of side length at least one, and I is a unit square, then $\omega^{\circ}(I) = 4$.

Relaxing the requirement of intersecting the interior of I, we have the following definition. It is perhaps less natural with the application to craniosynostosis in mind; however, as we will soon see, some work on the closely related kissing number notion exists.

Definition 3.2.7. (Relaxed overlap number.) Let $\mathcal{U}$ be a collection of sets in $\mathbb{R}^2$ and $I \in \mathcal{U}$. The relaxed overlap number of I, denoted $\omega_{\mathcal{U}}(I)$, is the largest collection $\mathcal{T} \subseteq \mathcal{U}$ of pairwise disjoint sets, which all intersect I.

Note that $\omega^\circ(I) \leq \omega(I)$. We now define the notion of kissing number. First, we say that two 2-dimensional figures *kiss* if their interiors have empty intersection and their boundaries have non-empty intersection. The kissing number is generally defined as follows, as seen in *The kissing number of the regular polygon* by Likuan Zhao [18], for instance.

Definition 3.2.8. (Kissing Number.) Let P_n be a regular n-gon. The kissing number $k(P_n)$ is the maximum number of pairwise disjoint copies of P_n that can be arranged so that each kiss P_n.

Since it is a notion seemingly quite closely related to overlap number, we give a brief overview of some results on kissing number.

Theorem 3.2.9. *(Kissing numbers.) The following kissing numbers are known:*

- $k(P_3) = 12$
- $k(P_4) = 8$
- $k(P_n) = 6$ *for all* $n \geq 6$*.*

Some work has also been done on rectangles. Let R be a rectangle with sides a and b, with $\frac{a}{b} = n + r$ with $n \in \mathbb{N}$, and $r \in (0, 1)$, then $2n + 6 \leq k(R) \leq 2n + 7$. The lower bound is tight if $r \leq \frac{1}{2}$, and the upper bound is tight if $r \geq \frac{\sqrt{3}}{2}$, as shown in [10].
We can generalize the kissing number definition in 3.2.8 to the following, so that it aligns with our definition of overlap number.

Definition 3.2.10. (Kissing number.) Let $\mathcal{U}$ be a collection of sets in the plane, and $I \in \mathcal{U}$. The kissing number of I with respect to $\mathcal{U}$, denoted $\kappa_{\mathcal{U}}(I)$, is the size of the largest collection $\mathcal{T} \subseteq \mathcal{U}$ of pairwise disjoint sets, which all intersect the boundary ∂I but not the interior I° of I.

To see how this definition of kissing number is more general, we can recover the old definition of $k(P_n)$ by letting $\mathcal{U}$ be the collection of all copies of a regular n-gon of a given size in the plane, and letting $I \in \mathcal{U}$ be one such n-gon. The notion of overlap and kiss are related by the inequality,

$$\kappa(I) \leq \omega(I), \tag{3.2.1}$$

since any kissing configuration is an overlapping one in the relaxed sense. The converse is not true in general.

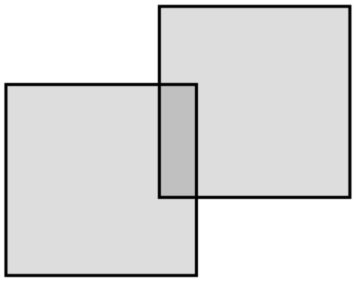
Figure 3.1: Overlapping squares.

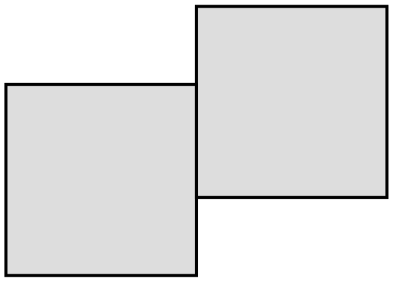
Figure 3.2: Kissing squares.

To tie this back to our performance guarantee of Algorithm 1, observe that if $\mathcal{U}$ is the collection of all instances, then corollary 3.1.4 holds with $k = \max_{I \in \mathcal{U}} \omega^{\circ}_{\mathcal{U}}(I)$. Furthermore, consider running the local ratio algorithm, scheduling regular n-gons in the plane, and where we used the relaxed version of overlapping. Then, the analogous analysis to the one in handling the axis paralell squares in example 3.2.1, together with inequality 3.2.1, gives us the following bound on the approximation ratio

$$r \leq \frac{1}{1 + k(P_n)}.$$

3.2.2 Order matters.

Using the overlap number to find a performance guarantee can be pessimistic. Perhaps there is a certain ordering on the instances such that $k = \max_{I \in \mathcal{U}} \omega^{\circ}_{\mathcal{U}}(I)$ is never attained.

Indeed, this is already the motivation behind choosing the interval with earliest end time in the original paper.

As a motivating example, there exists a collection $\mathcal{U}$ of $1 \times k$ rectangles, such that for one rectangle $R \in \mathcal{U}$, $\omega^\circ(R) = 2k + 2$, and for all other rectangles $R' \in \mathcal{U}$, $\omega^\circ(R') = 1$. An instance where $k = 6$ is seen in figure 3.3, where $k = 6$.

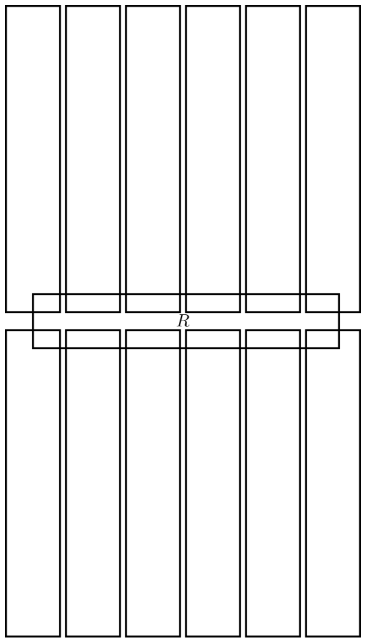

Figure 3.3: A set of overlapping rectangles with $\omega^\circ(R) = 12$, and $\omega^\circ(R') = 1$.

In spirit of the Bar-Yehuda algorithm, if we choose $\tilde{I} = R$, then we can only choose one rectangle, and the area covered is much smaller then if we sequentially choose all but R. Here the optimal solutions covering most area is choosing the twelve vertical rectangles. It is clear that knowing $\max_{I \in \mathcal{S}} \omega^\circ(I)$ gives us a performance guarantee, but it is not always the best performance guarantee we can have, as we soon show. An observation worth making is the following.

Remark 3.2.11. If $|\mathcal{U}| = n$ and some $I \in \mathcal{U}$ has $\omega^\circ(I) = k \leq n$, then for all $I' \in \mathcal{U}, I' \neq I$ we have $\omega^\circ(I') \leq n - k$.

Recalling figure 3.3, we have that $\omega(R) = 12$ for one rectangle, and $|\mathcal{U}| = 13$, so all other rectangles R' have $\omega(R') \leq 1$. It is then natural to define $\omega^\circ_\prec(I)$, the ordered overlap number, where $\prec$ is an ordering on $\mathcal{U}$, as the overlap of I with instances I' with $I \prec I'$.

Again, considering the example of figure 3.3, letting R be last under $\prec$, we have that the maximum ordered overlap number is 1. The order in which one considers the instances is thus key in the sense that one can take $k = \min_{\prec} \max_I \omega^{\circ}_{\prec}(I)$ in corollary 3.1.4.

3.2.3 Algorithm for fixed size axis parallel squares.

The following illustrates the idea that order matters, and what generalizing algorithm 2 to the plane looks like. From a medical perspective, simple shapes that are easy to cut, such as squares, might be desirable. We have an $n \times n$ grid of unit squares, where by (i, j) we denote the square in row i and column j in the grid, with $0 \le i \le n-1, 0 \le j \le n-1$. We are also given a collection of activities $\mathcal{A}_1, \dots, \mathcal{A}_m$, where each activity represents all possible placements of a square. All squares are of fixed side length $1 \le k \le n$, and $s(I) = (i, j)$ denotes the coordinates of the bottom leftmost vertex of I. At most one square can cover any entry in the grid, and we can place at most one square from every activity. The goal is to maximize the profit of a feasible placement. We give an algorithm (see algorithm 3) yielding a $\frac{1}{3}$-approximation.

Algorithm 3 Local Ratio: Choosing bottom leftmost $\tilde{I}$.

```
Input: $F, p, \mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_m$.
1: Delete all instances with non-positive profit.
2: if no instances remain then
3:     return $\mathcal{S} = \emptyset$.
4: Choose $\tilde{I}$ such that $s(\tilde{I}) = (i, j)$ has $i$ minimized, and subject to that, $j$ is minimized.
5: Decompose $p$ by $p = p_1 + p_2$. It is enough to define $p_1$, which we do as follows
       $$p_1(I) := p(\tilde{I}) \cdot \begin{cases} 1 & I \in \mathcal{A}(\tilde{I}) \\ \alpha w(I) & I \in \mathcal{I}(\tilde{I}) \\ 0 & \text{otherwise} \end{cases}$$
   where $\mathcal{A}(\tilde{I})$ is the activity to which $\tilde{I}$ belongs, and $\mathcal{I}(\tilde{I})$ the set of instances intersecting $\tilde{I}$ from other activities.
6: Solve the problem recursively on input $(F, p_2)$, and return $\mathcal{S}'$.
```

```
7: if $\mathcal{S}' \cup \{\tilde{I}\}$ is a feasible schedule then
8:     return $\mathcal{S} := \mathcal{S}' \cup \{\tilde{I}\}$
9: else
10:    return $\mathcal{S} := S'$
```

Again, proposition 3.1.2 holds, reworded here below to fit the vocabulary of the two dimensional situation.

Proposition 3.2.12. *If for all profit functions $p = p_1 + p_2$ with $p_2(\tilde{I}) = 0$, every $\tilde{I}$-maximal placement is an r-approximation with respect to p_1. Then the placement $\mathcal{S}$ returned by algorithm 3 is an r-approximation.*

Let $\prec$ be the ordering on $X := \cup \mathcal{A}_i$ in step 4 of the algorithm.

Claim 1. $\max_{I \in X} \omega_{\prec}(I) \leq 2$.

Proof. Consider a square $I \in X$. Let $I' \cap I \neq \emptyset$, with $I \prec I'$. Then, by definition of $\prec$, we have two (non-exclusive) cases:

1. I' intersects the bottom right corner of I
2. I' intersects the top right corner of I.

Hence, for any I'' with $I'' \cap I \neq \emptyset$ and $I' \cap I'' = \emptyset$, if I' is in case 1, then I'' is in case 2 and vice versa. Therefore, $\omega_{\prec}(I) \leq 2$. The claim follows. □

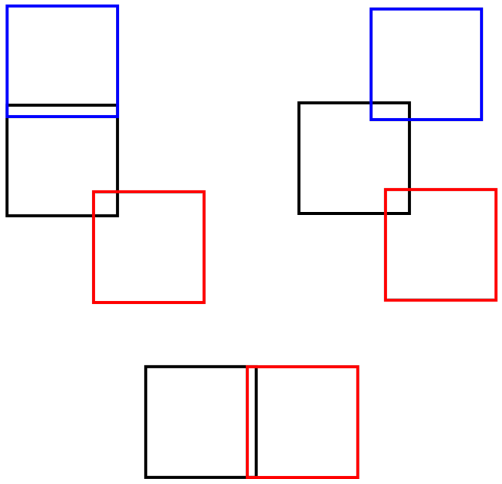
Figure 3.4: Examples of possible combinations of placements for I (in black), I' (in red), and I'' (in blue).

Now, we proceed analogously to the one dimensional setting. In other words, we give an upper bound b_{opt} on the optimum p_1 profit and a lower bound $b_{\max}$ on the p_1 profit of $\tilde{I}$-maximal placements.

Consider an optimal placement. By the choice of decomposition of p, the only instances contributing profit are instances in $\mathcal{I}(\tilde{I}) \cup \mathcal{A}(\tilde{I})$. By the above claim, instances in $\mathcal{I}(\tilde{I})$ contribute at most at most $2\alpha p(\tilde{I})$. The contribution of instances in $\mathcal{A}(\tilde{I})$ is at most one, as any placement contains at most one instance of each activity. Thus, $b_{opt} \leq 1 + 2\alpha$.

Consider now a $\tilde{I}$-maximal placement. Any such placement contains an elements of $\mathcal{A}(\tilde{I})$ or a set instances preventing the addition of $\tilde{I}$. Thus, $b_{\max} \geq \min\{1, \alpha\}$. Thus, the above algorithm is a $\frac{1}{3}$-approximation. Note that, the naive analysis, when not caring about the ordering, gave a $\frac{1}{5}$-approximation, so the ordering gave a significant improvement. Moreover, an analogous argument holds for identical axis parallel rectangles, oriented the same way. If we consider axis parallel rectangles of dimension $k_1 \times k_2$ with $k_1 < k_2$ and the two possible orientations are allowed, then $\omega_{\prec}(I) \leq \lceil \frac{k_2}{k_1} \rceil + 1$ by a similar argument to claim 1. This, in turn gives an approximation ratio of $\frac{\min\{1,\alpha\}}{1+\alpha(\lceil \frac{k_2}{k_1} \rceil + 1)}$. For instance, if $k_1 = 1$ and $k_2 = 10$, this gives a $\frac{1}{12}$-approximation.

3.3 Conclusion

In this chapter, we generalized scheduling intervals on the real line to geometric objects in the plane. We analyzed our problems using the local ratio and gave examples of instances with simple geometric objects, where we achieve constant approximation ratios.

We stumbled upon kissing number, and found it intriguing, that to our knowledge, nothing in the literature mentions the generalization of kissing number where intersection of interiors is allowed, as this seem to be a fundamental geometric problem. We used the notion of overlap number to give bounds on the approximability of our problems, and also noticed that the order in which the objects are processed play a significant role. How big is the gap when not caring about order versus when you do take order into account is an interesting question. How slim our objects were played a significant role in the approximability; it would be nice to have some form of general result relating the approximately to some high level parameter of a class of geometric object. Perhaps the ratio of the perimeter to the area is of interest?

The work in this chapter is somewhat removed from the original application of craniosynostosis; the plane is quite different from an ideal skull. A skull usually has non-zero curvature, for instance. If one is thinking of these geometric objects as skull pieces to be placed on a skull, perhaps the profit of a piece at a certain location in the plane could be chosen as to account for this skull-plane disparity. Further research into bridging the gap between the plane, and a model of the skull would be great to see.

Other, more general, questions that arose are: can one say more about the relation between kissing and overlap numbers? What are the overlap numbers for simple shapes like regular polygons? Are optimal overlapping and kissing configurations related?

This is part of what makes mathematics wonderful: we started with an optimization problem motivated by medicine, and are now asking questions about Euclidean geometry.

References

[1] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph (Seffi) Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48(5):1069–1090, 2001.

[2] Wikimedia Commons, 2020. Online; accessed 25-May-2022, at https://en.wikipedia.org/wiki/Craniosynostosis.

[3] Wikimedia Commons. File:gray566.png — wikimedia commons, the free media repository, 2022. [Online; accessed 8-July-2022].

[4] James Drake, Marina Drygala, Ricardo Fukasawa, Jochen Koenemann, Andre Linhares, Thomas Looi, John Phillips, David Qian, Nikoo Saber, Justin Toth, Chris Woodbeck, and Jessie Yeung. Optimized cranial bandeau remodeling, 2019. https://arxiv.org/abs/1912.10601.

[5] Marina Drygala. Craniosynostosis surgery: A study of rearrangement. Master's thesis, "University of Waterloo", 2020.

[6] Virtanen Pauli et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[7] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022.

[8] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[9] Wolfgang Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 34(5):827–828, September 1978.

[10] Arnfried Kemnitz and Meinhard Möller. On the newton number of rectangles. In *Bolyai Society Mathematical Studies, Proceedings of "The International Conference of Intuitive Geometry"*, pages 373–381, 1995.

[11] James Lawrence, Javier Bernal, and Christoph Witzgall. A purely algebraic justification of the Kabsch-Umeyama algorithm. *Journal of research of the National Institute of Standards and Technology*, 124, 10 2019.

[12] John Phillips. private communication with, 2021.

[13] N R Saber, J Phillips, T Looi, Z Usmani, Jonathan B, J Drake, and P CW Kim. Generation of normative pediatric skull models for use in cranial vault remodeling procedures. *Child's Nervous System*, 28(3):405–410, 2012.

[14] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit–An Object-Oriented Approach To 3D Graphics*. Kitware, Inc., fourth edition, 2006.

[15] B J Slater, K A Lenton, M D Kwan, D M Gupta, D C Wan, and M T Longaker. Cranial sutures: a brief review. *Plastic and reconstructive surgery*, 121(4):170e–178e, 2008.

[16] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.

[17] Jessie Yeung. private communication with, 2021.

[18] Likuan Zhao. The kissing number of the regular polygon. *Discrete mathematics*, 188(1):293–296, 1998.

APPENDICES

Appendix A

Data

A.1 Allowing middle placements.

Table A.1: Skull A.

ω	Covered area (%)	Objective	Non-region pieces	Region pieces	Run time (s)
1	70	2507.575	7	0	475.37
1	75	3593.693	8	0	1133.25
1	80	4328.770	8	0	1854.34
1	85	3956.226	8	0	2015.37
1	90	6654.603	9	0	3600.00
2	70	2319.224	7	0	155.38
2	75	2700.144	8	0	142.25
2	80	3514.421	9	0	204.26
2	85	4082.187	9	0	354.17
2	90	3852.434	9	0	387.53
2	95	6573.883	10	0	725.50
2	100	8207.805	10	0	3186.44

Table A.2: Skull B.

ω	Covered area (%)	Objective	Non-region pieces	Region pieces	Run time (s)
1	70	2704.192	8	0	155.01
1	75	3004.386	8	0	162.83
1	80	3752.150	9	0	227.27
1	85	4758.850	9	0	854.08
1	90	4960.380	9	0	1774.93
1	95	10452.819	10	0	3600.00
2	70	2454.839	8	0	196.42
2	75	2868.489	9	0	121.71
2	80	3270.785	9	0	224.35
2	85	4476.344	10	0	323.38
2	90	5125.455	10	0	363.32
2	95	6197.218	10	0	507.76
2	100	8081.915	8	1	2240.46

Table A.3: Skull C.

ω	Covered area (%)	Objective	Non-region pieces	Region pieces	Run time (s)
1	70	2838.462	8	0	282.52
1	75	3304.546	8	0	317.49
1	80	4050.125	9	0	1107.77
1	85	4746.464	9	0	780.92
2	70	2144.863	8	0	95.27
2	75	2673.033	8	0	187.92
2	80	3120.331	9	0	277.21
2	85	3962.262	9	0	527.27
2	90	4852.374	10	0	294.21
2	95	6647.937	10	0	628.97

Table A.4: Skull D.

ω	Covered area (%)	Objective	Non-region pieces	Region pieces	Run time (s)
1	70	2475.588	8	0	474.15
1	75	2940.906	9	0	689.80
1	80	4105.957	9	0	3600.00
1	85	5015.818	10	0	3600.00
2	70	2049.013	9	0	108.52
2	75	1937.468	9	0	141.72
2	80	2511.720	10	0	129.32
2	85	3471.584	10	0	545.08
2	90	5108.069	10	0	661.89
2	95	7623.566	10	0	3600.00

Table A.5: Skull E.

ω	Covered area (%)	Objective	Non-region pieces	Region pieces	Run time (s)
1	70	2749.418	8	0	406.68
1	75	3116.832	8	0	563.27
1	80	4609.550	9	0	825.31
1	85	4435.781	9	0	3600.00
1	90	8449.245	10	0	3600.00
2	70	2437.279	8	0	207.68
2	75	2305.033	9	0	196.58
2	80	3445.759	9	0	230.26
2	85	3690.297	10	0	284.93
2	90	4597.658	10	0	507.06
2	95	8116.526	10	0	3600.00

A.2 Disallowing middle placements.

Table A.6: Skull A.

ω	Covered area (%)	Objective	Non-region pieces	Region pieces	Run time (s)
1	70	7535.518	5	1	21.69
1	75	7483.207	5	1	22.80
1	80	10323.443	3	2	22.21
1	85	13393.852	6	1	115.37
1	90	24250.883	7	1	258.34
1	95	27390.623	9	0	1980.75
2	70	7760.255	5	1	26.01
2	75	7204.546	5	1	25.62
2	80	8818.526	6	1	27.07
2	85	10560.802	7	1	29.19
2	90	23457.060	10	0	35.32
2	95	24103.856	8	1	35.36
2	100	26041.755	8	1	237.48

Table A.7: Skull B.

ω	Covered area (%)	Objective	Non-region pieces	Region pieces	Run time (s)
1	70	10592.282	3	2	15.70
1	75	6064.486	6	1	17.87
1	80	7099.673	6	1	22.37
1	85	13536.013	5	2	66.88
1	90	22477.605	9	0	245.07
1	95	20958.717	10	0	251.16
1	100	31258.425	8	1	1697.08
2	70	5164.049	5	1	18.55
2	75	5378.347	6	1	18.51
2	80	6334.648	7	1	18.52
2	85	13352.410	6	1	19.72
2	90	14192.971	7	1	22.42
2	95	22766.010	8	1	53.61
2	100	19740.513	10	0	287.80

Table A.8: Skull C.

ω	Covered area (%)	Objective	Non-region pieces	Region pieces	Run time (s)
1	70	7816.304	3	2	14.00
1	75	17343.216	4	2	19.66
1	80	18140.420	5	2	27.99
1	85	10891.765	7	1	115.11
1	90	28642.257	8	1	365.18
1	95	24632.978	10	0	2040.62
2	70	7732.713	3	2	19.26
2	75	8324.796	4	2	21.34
2	80	8904.369	5	2	22.06
2	85	10668.587	6	2	39.96
2	90	11781.942	8	1	42.87
2	95	13121.046	8	1	102.29
2	100	13859.205	10	0	1026.25

Table A.9: Skull D.

ω	Covered area (%)	Objective	Non-region pieces	Region pieces	Run time (s)
1	70	7244.307	2	3	21.55
1	75	8841.618	3	3	23.77
1	80	7130.499	7	1	76.84
1	85	8959.342	8	1	52.94
1	90	19030.084	10	0	666.57
1	95	24143.828	10	0	3600.00
2	70	4399.714	4	2	20.76
2	75	6378.661	3	3	23.08
2	80	7295.066	8	1	26.68
2	85	13022.260	6	2	38.31
2	90	8303.358	8	1	169.62
2	95	11545.489	10	0	388.21
2	100	17936.708	10	0	3600.00

Table A.10: Skull E.

ω	Covered area (%)	Objective	Non-region pieces	Region pieces	Run time (s)
1	70	7223.498	8	0	15.58
1	75	8187.419	9	0	27.76
1	80	11571.038	9	0	26.65
1	85	11489.222	9	0	92.45
1	90	12509.854	10	0	835.43
1	95	18203.098	10	0	3600.00
2	70	5779.015	6	1	18.82
2	75	5531.696	7	1	18.33
2	80	7217.611	7	1	24.73
2	85	12087.947	10	0	20.42
2	90	11369.308	10	0	73.13
2	95	11223.301	10	0	310.85
2	100	23192.389	10	0	3600.00