Craniosynostosis Surgery: A Study of Rearrangement

by

Marina Drygala

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

Waterloo, Ontario, Canada, 2020

Author's Decloration

Abstract

Cranial vault remodeling, a form of skull surgery is currently performed according to intuition developed over years of experience. The problem of how to best perform this surgery has not yet thoroughly been studied by computer scientists.

In this thesis we provide a mathematical formulation of a simplified version of cranial vault remodeling surgery, and prove $\mathcal{NP}$-hardness for our formulation. We then provide several heuristics and test their performance in an experimental study.

Acknowledgements

I would like to thank my supervisors Ricardo Fukasawa and Jochen Könemann for their valuable guidance throughout this process, that made this work possible.

I would also like to thanks my readers Bertrand Guenin and Levent Tunçel for their time and valuable comments.

Lastly, I would like to thank Dr. John Phillips and Dr. James Drake from SickKids Hospital and Chris Woodbeck from SickKids Hospital and the University of Waterloo for all of their insights.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The motivation behind the work contained in this thesis is to develop algorithmic surgical solutions for children born with a condition called craniosynostosis. Normally, when a child is born, their skull's bones have not yet been fused. During normal development, the cranial bones will fuse between 12 and 18 months [4]. This delay allows the infant's brain to grow rapidly during the first few months of the infant's life. An individual with craniosynostosis has one or more of these sutures fuse prematurely. If left untreated, as an affected infant's brain develops, their skull will become deformed because the closed sutures restrict growth. Also, there could be increased pressure on the brain leading to other complications, including mental development [5]. See Figure 1.1 for images of affected individuals. This condition affects approximately 1 in every 2000 infants [6], and most cases require surgical intervention to treat.

There are different types of surgeries that a craniosynostosis patient may undergo. The decision of which procedure(s) to perform will depend on several factors. These include the age of the patient, the severity of the case, and which sutures have fused [7]. In any such procedure, the general approach is to break or

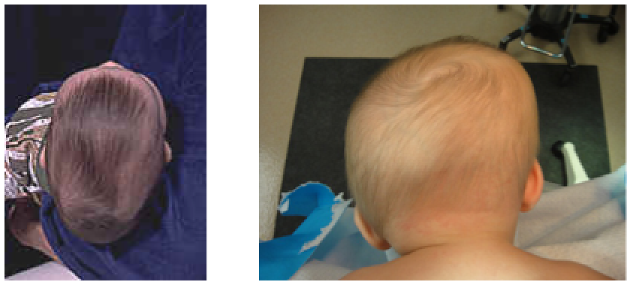

Figure 1.1: Infants affected by craniosynostosis [1].

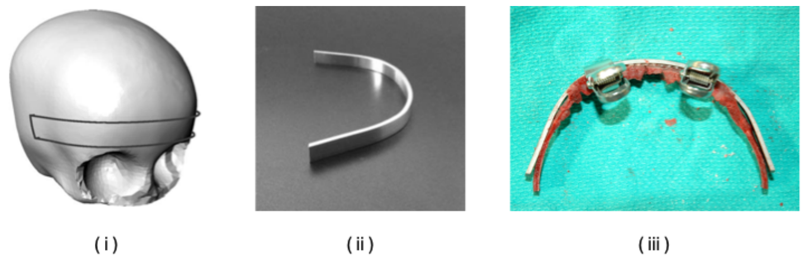

Figure 1.2: (i) Location of the bandeau, (ii) Template of proper bandeau shape, (iii) Patient bandeau attached to template [2].

weaken the bones and remodel them. It is informative to discuss different types of performed surgeries. One such procedure is the remodeling of the front-orbital bandeau, a segment of bone running horizontally across the forehead just above the eyes (see Figure 1.2 (i)). The authors of [2] have developed normative bandeau shapes as in Figure 1.2 (ii), that they use as templates while performing surgery. They remove the bandeau during the procedure, after which incisions are made vertically along the bandeau's length. The skull can then be bent and reshaped into a form as close to that of the template as possible (see Figure 1.2 (iii)). We will call each region of the bandeau between two incision points or between one end of the bandeau and one incision point a *bandeau / bone segment* or *bandeau / bone piece*. Following reshaping, the surgeon places the reshaped bandeau into the infant's skull in the same location from which it was removed. This procedure is often called fronto-orbital advancement or reshaping and can be performed alone or as part of a larger remodeling operation. Note here that surgeons do not break the bandeau into separate pieces at the incision points. The incisions made to the skull make it more malleable, allowing it to be bent into the desired form. We are also interested in the more general cranial vault remodeling surgery. The cranial vault is the area formed by the skull, containing the brain. This procedure involves making incisions over a larger area of the skull, after which bone pieces are created by separating the skull into pieces along the incisions. The surgeon then places a subset of these pieces back into the infant's head in a pattern that attempts to emulate how the baby's skull should look at the current stage of their development [8]. We discuss the required properties of this pattern in the following paragraph. See Figure 1.3 for a visualization of the procedure.

When performing either remodeling procedure, the surgeon will have two objectives: First, limit the time required to complete the surgeries. This objective is necessary because we would like to minimize the duration of the infant's anesthetization period, leading to reduced risks for the baby and financial costs. This restriction is captured by cutting and replacing only a small number of bone segments into the head.

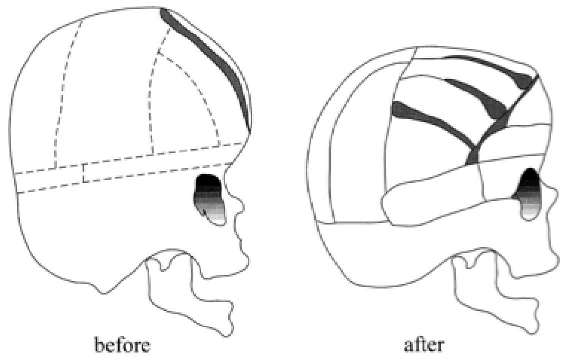

Figure 1.3: Before and after sketches of the cranial vault remodelling surgery. [3]. The dotted lines in the before image display incisions.

It should be noted that some pieces will take slightly longer than others to cut so that our model will not capture our objective precisely. However, in practice, this is a good enough approximation because most cuts the surgeons make will be straight. The second objective is to replace bone segments onto the infant's skull in a pattern that best models a normal skull's shape. The authors in [2] have also created a library of normative skull shapes for different age categories. How well a pattern of replaced bone segments matches a normative shape is judged along several criteria. Each placed skull piece should closely match the form of a normal infant's skull over the entire location of the brain that it covers. Also, we want to place the pieces in such a fashion that, together, they cover enough of the brain to fuse later, forming a complete skull. In practice, any areas of the vault left exposed should be small so that the pieces can grow later, covering these gaps. We will refer to these two criteria as fit and coverage.

The authors in [9] provide a dynamic programming approach for surgeons performing a fronto-orbital advancement procedure. As noted above, surgeons do not separate the bone segments during this procedure. As a result, they do not reorder segments along the length of the bandeau. However, the surgeons may alter the created bone segments' order during the cranial vault remodeling procedure. This change does not allow us to extend the dynamic programming approach given in [9] for the cranial vault remodeling procedure. In this thesis, we study a hybrid of the two surgeries of interest to understand the cranial vault remodeling problem, namely *fronto-orbital advancement with rearrangement.* This procedure is the same as the fronto-orbital advancement procedure, except that the surgeon may separate the bone segments and rearrange their order along the bandeau's length.

1.1 Thesis Outline

In chapter 2 we give a mathematical formulation which we call (CRPR) for the fronto-orbital advancement with rearrangement procedure. We also formalize a related problem which we call (CRPRPS), where the skull pieces have already been cut, and the surgeon need only place them. Our formulation is a minimization problem, penalizing a lack of fit between pieces of the deformed bandeau and the segment of the template they cover and penalizing leaving areas of the template uncovered or covered multiple times by bone. We then review fundamental concepts from complexity theory and combinatorial optimization such as greedy algorithms, local search algorithms, and primal-dual algorithms.

In chapter 3, using a pseudo-polynomial reduction from 3-Partition, we prove that (CRPR) is strongly $\mathcal{NP}$-complete, meaning that we cannot likely find an efficient algorithm to solve it. By the same result, we show (CRPRPS) is also strongly $\mathcal{NP}$-complete. Without further simplification, a corollary gives us the result that we cannot likely find an efficient algorithm with an efficiently computable approximation guarantee for instances of (CRPR) or (CRPRPS).

To overcome our inapproximability results, in chapter 4 we propose a modification to our cost function, for which only hardness is maintained. The modification leaves us with a maximization problem in which we are rewarded for covering segments of the template, but again are penalized for a lack of fit between pieces of the deformed bandeau and the segment of the template they cover and covering areas of the template multiple times by bone. We formulate both (CRPR) and (CRPRPS) as an optimization problem over an independence system with this modification. We then use the standard greedy approach and modify it to improve performance when applied to instances of (CRPR). We also reduce (CRPRPS) to a resource allocation problem, for which we can apply the tools developed by the authors in [10] to get a $\frac{1}{2}$-approximation. We then offer a local search heuristic and try to improve it using primal-dual techniques.

Chapter 5 gives an experimental study which compares how our approaches perform on practical instances. Our primal-dual techniques outperform the other heuristics outputting solutions that are within 10% of optimal most of the time. We did not observe an instance with a performance ratio below 50% of optimal. Together our results suggest that our methods warrant further study.

Throughout, we highlight leads on future directions for research.

Chapter 2

Preliminaries

In this chapter, we formulate how to best perform a fronto-orbital advancement with rearrangement procedure as a mathematical problem. Also, we review the necessary concepts from optimization required for our work.

We first define some of the basic notation we will use throughout this thesis. For a positive integer n, we will use $[n]$ to denote the set of integers from 1 to n. Also, for a set S and a positive integer n, where $|S| \geq n$, we will use the notation $\binom{S}{n}$ to denote the set of all subsets of S of size n.

2.1 Basic Definitions

In section 2.2, we will use the terminology defined in this section to construct a formal mathematical model for the fronto-orbital advancement with rearrangement procedure.

2.1.1 Curves

We will use a particular class of curves in our work, for which we provide the formal definitions here.

Definition 1 (Line Segment)**.** *A line segment is a set of points that can be expressed as a set* $L = \{(x, y) : x = \lambda x' + (1 - \lambda)x'', y = \lambda y' + (1 - \lambda)y'', \lambda \in [0, 1]\}$ *for some* $(x', y'), (x'', y'') \in \mathbb{R}^2$*.*

Definition 2 (Piecewise-Linear Curve)**.** *A piecewise-linear curve is a series of connected line segments defined by a list of points* $H = [(x_0, y_0), \ldots, (x_{|H|-1}, y_{|H|-1})]$*, where the line segments of* h *are those defined by the endpoints* (x_i, y_i) *and* (x_{i+1}, y_{i+1}) *for* $i \in \{0\} \cup [|H| - 2]$*.*

Definition 3 (Piecewise-Linear Function). *A piecewise-linear function is a piecewise-linear curve that can also be expressed as the set of points C, where $C = \{(x, y) : y = f(x), \ell \leq x \leq r\}$ for some $\ell, r \in \mathbb{R}$ where $f : \mathbb{R} \to \mathbb{R}$ is a function.*

For a piecewise linear curve h defined by a H, we call H the *discretization* of h. The *left endpoint* of a curve h is the first point in its discretization, denoted $L(h)$. Similarly, the *right endpoint* of a curve h is the last point in the discretization, denoted $R(h)$. For any point $(x, y) \in h$ we let $x((x, y)) = x$ and $y((x, y)) = y$. We say that a list of points are *consecutive* in H if they appear consecutively in H. We say that a list of line segments L of h are *consecutive* if every pair of line segments appearing next to one another in L share an endpoint.

Definition 4 (Simple Piecewise-Linear Curve). *A piecewise-linear curve is simple if only consecutive line segments intersect, and only at their endpoints.*

We now define for any curve h, a natural ordering $\prec_h$ on the points of h. Since all elements of h lie on some line segment between (x_i, y_i) and (x_{i+1}, y_{i+1}) for some $i \in \{0\} \cup [|H| - 2]$, we can express each (x, y) as

$$\lambda(x_i, y_i) + (1 - \lambda)(x_{i+1}, y_{i+1})$$

where $\lambda \in [0, 1]$. For $(x, y), (x', y') \in h$ we say that $(x, y) \preceq_h (x', y')$ in h if there exists $i, i' \in \{0\} \cup [|H| - 1]$ such that

$$(x, y) = \lambda(x_i, y_i) + (1 - \lambda)(x_{i+1}, y_{i+1})$$

and

$$(x', y') = \lambda'(x_{i'}, y_{i'}) + (1 - \lambda')(x_{i'+1}, y_{i'+1}),$$

where $i' < i$ or where $0 \leq \lambda \leq \lambda' \leq 1$ and $i = i'$. We denote this ordering by $\prec_h$.

Note that a curve h is a subset of points in $\mathbb{R}^2$, and hence we can use standard subset notation writing $h_1 \subseteq h_2$ if h_1 contains only points that are also contained in h_2.

Definition 5 (Curve Equality). *We say that two curves h_1, h_2 are equal if $h_1 \subseteq h_2$ and $h_2 \subseteq h_1$. We denote this $h_1 = h_2$.*

Note that if h_1 and h_2 are two equal curves, they may still have different discretizations.

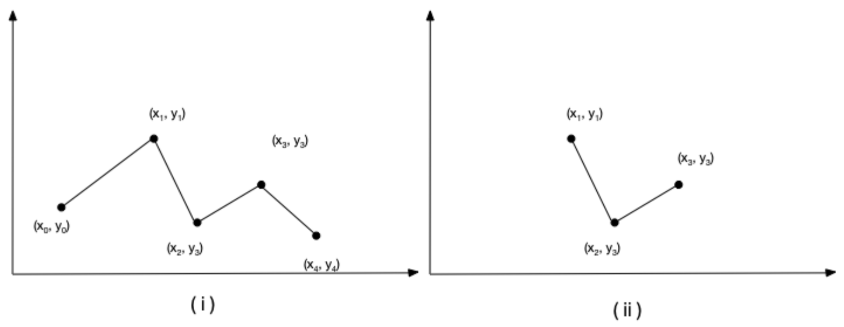

Figure 2.1: (i) Visualization of a curve, and (ii) a corresponding curve segment

2.1.2 Curve Segments

In this thesis, for a curve h we will often be interested in a subsets of h that take a particular form. We will call a subset of a curve that takes our desired form a *curve segment*. Here we provide the formal definition of curve segment and some related definitions and notation.

Definition 6 (Curve Segment)**.** *Given a curve h, with discretization $H = [(x_0, y_0), \ldots, (x_{|H|-1}, y_{|H|-1})]$, a* curve segment *$h'$ of h is any curve with discretization H' such that*

$$H' = [(x, y), (x_i, y_i), \ldots, (x_{i'}, y_{i'}), (x', y')],$$

where $(x, y), (x', y') \in h$, $i, i' \in [|H| - 2]$ satisfy

$$(x, y) \prec_h (x', y'),$$

$$(x_{i-1}, y_{i-1}) \preceq_h (x, y) \prec_h (x_i, y_i)$$

and

$$(x_{i'}, x_{i'}) \prec_h (x', y') \preceq_h (x_{i+1}, y_{i+1}).$$

Please see Figure 2.1(i) for a visualization of a curve, and Figure 2.1(ii) for a visualization of a curve segment.

Note that given h only two elements of h are necessary to fully define any curve segment h', namely the left and right endpoints of h' respectively. If $L(h') = (x, y), R(h') = (x', y')$ we call h' the curve segment of h restricted to (x, y) and (x', y'), denoted $h|_{(x,y),(x',y')}$.

Definition 7 (Adjacent Curve Segments)**.** *Suppose h is a curve, and h' and h'' are curve segments of*

h. We say that h' and h'' are adjacent *in h if $h' = h|_{(x,y),(x',y')}$ and $h'' = h|_{(x',y'),(x'',y'')}$ for some $(x,y),(x',y'),(x'',y'') \in h$ such that $(x,y) \prec_h (x',y') \prec_h (x'',y'')$.*

In Figure 2.1(i), the curve segments $h_{(x_0,y_0),(x_1,y_1)}$ and $h_{(x_1,y_1),(x_4,y_4)}$ are adjacent curve segments, because the right endpoint of $h_{(x_0,y_0),(x_1,y_1)}$ equals the left endpoint of $h_{(x_1,y_1),(x_4,y_4)}$.

We will next define an operation which takes a curve, and creates curve segments.

Definition 8. *Let h be a curve and $p_1, \ldots, p_k \in h$ such that $p_1 \prec \cdots \prec p_k$. If we* separate *$h$ at $p_1 \prec \cdots \prec p_k$ we create segments*

$$h_{L(h),p_1}, h_{p_1,p_2}, \ldots, h_{p_k,R(h)}.$$

In the case that $p_1 = L(h)$ we exclude $h_{L(h),p_1}$, and in the case that $p_k = R(h)$ we exclude the element $h_{p_k,R(h)}$.

We say that curve segments $h_{a,b}, h_{c,d}$ of h can be formed using the *same separation operation* if $b \preceq_h c$ or $d \preceq_h a$. In otherwords there is no curve segment of h contained in both $h_{a,b}$ and $h_{c,d}$.

For a curve h and a list $\bar{H}$ of points in h ordered with respect to $\prec_h$, we let $h_{/\bar{H}}$ denote the curve segments formed by separating h at $\bar{H}$. We call the elements of $h_{/\bar{H}}$ *unit segments* or *unit intervals* of h with respect to $\bar{H}$.

For an example take h to be the curve in Figure 2.1(i). If we set $\bar{H} = \{(x_0,y_0),(x_1,y_1),(x_4,y_4)\}$, then $h_{/\bar{H}}$ would contain the curves $h_{(x_0,y_0),(x_1,y_1)}$ and $h_{(x_1,y_1),(x_4,y_4)}$.

In future sections we will use i to denote the i^{th} element of $\bar{H}$ for $i \in \{0\} \cup [|\bar{H}|-1]$. We will use $x(i), y(i)$, if we wish to discuss the specific x or y coordinates of point i. Using this notation we can use $h|_{i,i+1}$ to denote the curve segment in $h_{/\bar{H}}$ obtained by restricting h to points i and $i+1$ of $\bar{H}$ for $i \in \{0\} \cup [|\bar{H}|-2]$. To simplify our notation even further, we will also use the set $\{0\} \cup [|\bar{H}|-2]$ to refer to element $h|_{i,i+1}$ of $h_{/\bar{H}}$. It will be clear from context if we using an index i to refer a point or to a unit segment.

In our above example we would then use the integers $0, 1$ and 2, respectively, to refer to the elements $(x_0,y_0), (x_1,y_1)$, and (x_4,y_4) of $\bar{H}$, and we would also use the integers 0 and 1, respectively, to refer to the curve segments $h_{(x_0,y_0),(x_1,y_1)}$ and $h_{(x_1,y_1),(x_4,y_4)}$.

2.1.3 Moving Curves

In addition to separating curves we will be interested in operations that move curve segments in the plane. In this section we give the formal definitions of these operations.

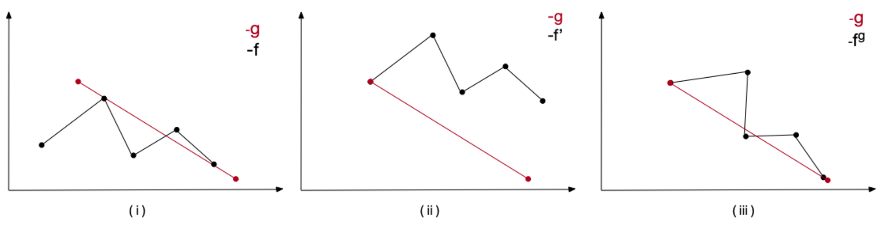

Figure 2.2: (i) Curves f and g, (ii) curves f' and g, (iii) curves f^g and g

Definition 9 (Curve Translation)**.** *Given a curve h, with discretization H, the translation of h by $(t_x, t_y) \in \mathbb{R}^2$, is the curve h' defined by discretization $H' = [(x + t_x, y + t_y) : (x, y) \in H]$.*

If we translate h *to* a point (x, y), we mean the translation of h by (t_x, t_y), where (t_x, t_y) is chosen so that the left endpoint of h' is (x, y).

Definition 10 (Curve Rotation)**.** *Given a curve h, with discretization H, the rotation of h by $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, is the curve h' defined by discretization $H' = [r((x, y)) : (x, y) \in H]$, where $r((x, y))$ returns the point obtained by rotating (x, y) by θ radians around $L(h)$.*

Definition 11 (Curve Matching)**.** *Suppose we are given one fixed curve g, and another curve f, with discretizations G and F respectively. Matching f to g is the operation by which we obtain the curve f^g through the following sequence of operations. First translate f to $L(g)$ to obtain f'. From there let $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ be the angle obtained by rotating the vector $R(f') - L(f')$ onto the vector $R(g) - L(g)$. We obtain f^g by rotating f' by θ around $L(f')$. See Figure 2.2 for a visualization. We may also say that f is* placed over *g.*

2.2 Mathematical Formulation

We now provide a mathematical formulation with which we study the fronto-orbital advancement with rearrangement procedure introduced in Section 1; we will refer to this framework as the *Curve Reshaping Problem with Rearrangement* (CRPR).

As a thin segment of bone characterizes the bandeau, we can describe its shape by a curve. In an instance of (CRPR), we are given as input an *ideal* curve g with discretization G and a subset $\bar{G}$ of G. The curve g will play the role of the template used in the fronto-orbital advancement procedures. We are also given a *deformed* curve f with discretization F and subset $\bar{F}$ of F, which represents the shape of the affected infant's bandeau. We will model the notion of making incisions along the infant's bandeau by separating the curve f into segments. We will model the notion of placing the resulting pieces of bone along the template by matching a subset of the resulting segments of f to segments of g. We will be interested only in segments of f that can be formed by separating f along a subset of points in $\bar{F}$ and matching to segments of g that can be formed by separating g at a subset of points in $\bar{G}$.

As previously mentioned, it is desirable to limit the number of cuts we are permitted to make. Hence we are given a parameter $k \geq 0$, indicating that we will be allowed to match at most k segments of f to g. Since at most $2k$ cuts are required to create any k feasible segments of the deformed bandeau, the two decisions are correlated. Given a parameter k we will select a subset of points to separate f along forming a set of curve segments from which we will select k' pieces, for some $k' \leq k$ and place a subset of these on our ideal curve. As a result, our first decision is to select k' pairs of points

$$\{(a_0, b_0), \ldots, (a_{k'-1}, b_{k'-1})\} \subseteq \binom{\bar{F}}{2} \tag{2.1}$$

such that we can apply the same separation operation to obtain the pieces in equation (2.2).

$$\{f|_{a_0,b_0}, f|_{a_1,b_1} \ldots, f|_{a_{k'-1},b_{k'-1}}\} \tag{2.2}$$

The curve segments in equation 2.2, will be indexed by the set $\{0\} \cup [k'-1]$, which we will refer to as J. The choice of J is made so that we must not always list the elements in equation (2.1), when the decision of where to cut is made. To complete the solution we must select $J^* \subseteq J$ and match each deformed segment in J^* to a segment of g. This corresponds to selecting pairs of elements of $\bar{G}$ as in equation (2.3).

$$(\ell_j, r_j) \in \binom{\bar{G}}{2}, \text{ for each } j \in J^*. \tag{2.3}$$

The elements in equations 2.1 and 2.3 are sufficient to describe a solution S to an instance of (CRPR). In this thesis, we will also study the problem where the choice of where to separate f has already been made. This problem will be called *Curve Reshaping Problem with Rearrangement and Pre-cut Segments* (CRPRPS). A solution to (CRPRPS) requires only the elements in equation 2.3.

When discussing a solution S, and a piece $j \in J$ the choice to place $f|_{a_j,b_j}$ over $g|_{\ell_j,r_j}$ can be described by the tuple (j, ℓ_j, r_j) or (a_j, b_j, ℓ_j, r_j). We call such tuples *placements*, as they represent the choice of placing $f|_{a_j,b_j}$ over $g|_{\ell_j,r_j}$. If S includes the placement (j, ℓ_j, r_j) we say $S_j = (\ell_j, r_j)$, or equivalently $(j, \ell_j, r_j) \in S$. If piece j is not included in S, we say $S_j = \emptyset$.

We introduced the notions of fit and coverage in section 1 when we discussed the desired qualities our remodeled skull has. We will introduce the cost function c to evaluate fit and the cost function μ to evaluate coverage.

The fit function c takes as input a curve segment of f and a curve segment of g and evaluates how well they would fit together if they were matched. As previously mentioned, such a measure should take into account the similarity of size and form of a pair of segments. Formally,

$$c : \binom{\bar{F}}{2} \times \binom{\bar{G}}{2} \to \mathbb{R}_{\geq 0} \cup \{\infty\},$$

where for $(a, b) \in \binom{\bar{F}}{2}, (\ell, r) \in \binom{\bar{G}}{2}$, $c(a, b, \ell, r) = 0$ would indicate a perfect fit of $f|_{a,b}$ to $g|_{\ell,r}$, while $c(a, b, \ell, r) = \infty$ would indicate that $f|_{a,b}$ cannot be fit to $g|_{\ell,r}$.

To formalize the definition of μ, we first introduce some terminology. For a piece $j \in J$ and $(\ell, r) \in \binom{\bar{G}}{2}$ we say that the placement (j, ℓ, r) *covers* segments $\ell, \ldots, r-1$ of $g_{/\bar{G}}$. The coverage function will take the form,

$$\mu : g_{/\bar{G}} \times \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0} \cup \{\infty\},$$

where $\mu(i, z)$ returns a quantity evaluating how undesirable it is to cover interval $i \in g_{/\bar{G}}$ by z segments of f. In particular μ should satisfy $\mu(i, 1) = 0$ for all $i \in g_{/\bar{G}}$, so that it is not undesirable to cover an interval one time. In addition, for $z' > z \geq 1$, we should have $\mu(i, z') \geq \mu(i, z)$ as it is undesirable for our application to cover an interval multiple times. Since $\mu(i, 0) \geq \mu(i, 1)$ for all $i \in g_{/\bar{G}}$ we say that we *collect the reward* $\mu(i, 0)$ for covering interval i. Note that μ does not capture the idea that the area of consecutive unit segments left exposed should not be too large as mentioned. Instead μ only penalizes the total area left exposed. We will see in chapter 3 that even with this simplifying assumption (CRPR) and (CRPRPS) are hard.

For a solution S to (CRPR) or (CRPRPS) we let $cov_S(i)$ be the number of pieces in J^* that cover $i \in g_{/\bar{G}}$. If two placements both cover some $i \in g_{/\bar{G}}$ we say they *overlap* with one another. If we wish to be more specific we may say they overlap at i. If two placements overlap with one another or are placements of the same piece we say that they *interfere* with one another.

The cost of S is given in equation (2.4).

$$C_{min} = \sum_{j \in J^*} c(a_j, b_j, \ell_j, r_j) + \sum_{i \in g_{/\bar{G}}} \mu(i, cov_S(i)). \tag{2.4}$$

We may equivalently express the quantity $c(a_j, b_j, \ell_j, r_j)$ as $c(j, \ell_j, r_j)$.

Amongst all solutions S, we wish to pick one that minimizes the quantity in (2.4). See section 2.2.2 for a description of the cost functions we use in practice. In the future for each piece $j \in J$ we will let $\mathcal{P}_j$ be the set of all placements that piece j can be fit to. Formally,

$$\mathcal{P}_j = \{(\ell, r) \in \binom{\bar{G}}{2} : c(j, \ell, r) < \infty\}.$$

2.2.1 Representing Inputs and Problem Size

We will assume that the values for c or μ for a given instance will be rationals. The functions c and μ are presented as arrays. It is clear that the input size of an instance of (CRPR) or (CRPRPR) is $O(b(|G|^2|F|^2))$, where b is the largest bit size of any element of in G, F or the outputs of c and μ. We may assume that $k \leq |\bar{F}| - 1$, because we cannot cut more than $|\bar{F}| - 1$ pieces from f. As a result, the quantity $O(b(|F|))$ bounds the encoding size of any solution S. With the assumption that $k \leq |\bar{F}| - 1$, any algorithm that allows us to find the optimal solution for matching exactly k segments of f to g will result in an algorithm for matching at most k segments of f to g with the same run-time aside from a factor of $|F|$. Thus when suitable, we may choose to assume that the number of pieces cut by a solution, k' equals k.

2.2.2 Proposed Cost Functions

In this section we will discuss different cost and fit functions used throughout this thesis.

Cost Functions for Experimental Study

We will use the cost functions c, μ given in this section for our experimental studies.

For $(a, b) \in \binom{\bar{F}}{2}, (\ell, r) \in \binom{\bar{G}}{2}$, we want $c(a, b, \ell, r)$ to give us a measure of how well the section of the deformed skull corresponding to $f|_{a,b}$ models the section of the template corresponding to $g|_{\ell,r}$. If $f|_{a,b}$ and $g|_{\ell,r}$ are not similar in size, then the two segments are clearly not a good match. To model this we define the quantity $m(a, b, \ell, r)$ as follows. Let $E_2(x, y)$ denote the standard Euclidean distance between x and y. We will call the Euclidean distance of a curve the Euclidean distance between its endpoints. We define

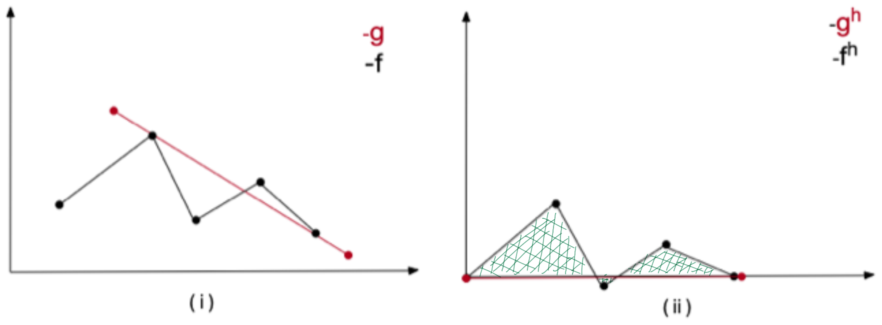

Figure 2.3: Taking $f = f|_{a,b}$ and $g = g|_{\ell,r}$, (i) Curves f and g, (ii) curves f^h and g^h, with the green shaded area being $c(a, b, \ell, r)$

$m(a, b, \ell, r)$ in equation (2.5).

$$m(a, b, \ell, r) = \frac{E_2(a, b)}{E_2(\ell, r)}. \tag{2.5}$$

If $m(a, b, \ell, r) > 1 + \delta$ or $m(a, b, \ell, r) < 1 - \delta$, for some fixed $\delta \geq 0$ we say $f|_{a,b}$ cannot be fit to $g|_{\ell,r}$ and we set $c(a, b, \ell, r) = \infty$. The motivation for this choice of m is that we wish only to cover areas of the template with segments of the deformed bandeau that are similar in size.

Otherwise $1 - \delta \leq m(a, b, \ell, r) \leq 1 + \delta$, and we are satisfied that $f|_{a,b}$ and $g|_{\ell,r}$ are sufficiently similar in size. In order to measure of how well the section of the deformed skull corresponding to $f|_{a,b}$ models the section of the template corresponding to $g|_{\ell,r}$, we must look at how similar the curves $f|_{a,b}$ and $g|_{\ell,r}$ are over their length. We propose to approximate the volume difference that would arise between the curve $f|_{a,b}^{g|_{\ell,r}}$ obtained by matching $f|_{a,b}$ to $g|_{\ell,r}$. Recall that we use f^g to denote the curve obtained by matching f to g.

To approximate this quantity we do the following. Let h be the curve with discretization $H = [(0, 0), (M, 0)]$ for some $M > 0$. Then we match $f|_{a,b}$ and $g|_{\ell,r}$ to h to get $f|_{a,b}^h$ and $g|_{\ell,r}^h$. We will only allow instances of input curves f and g, such that for any curve segments f' and g' of f and g, if we match f' and g' to h, to obtain f'^h and g'^h, that f'^h and g'^h are piecewise-linear functions.

Let t be the smallest x coordinate among $R(f|_{a,b}^h)$ and $R(g|_{\ell,r}^h)$. Since $f|_{a,b}^h$ and $g|_{\ell,r}^h$ are functions the value $c(a, b, \ell, r)$ in equation (2.6) is well defined

$$c(a, b, \ell, r) = \int_{0,t} abs(f|_{a,b}^h(x) - g|_{\ell,r}^h(x))dx, \tag{2.6}$$

where $abs(\cdot) = |\cdot|$. Please see Figure 2.3 for an illustration of $c(a, b, \ell, r)$.

We now construct our choice of function μ. We will do this in such a way as to penalize solutions proportional to the size of the total area they leave exposed. Formally, let (x_i, y_i) and $(x_{i'}, y_{i'})$ be two consecutive elements of $\bar{G}$, where the indices i, i' are chosen with respect to the index of the points in G. We take μ as defined by equation (2.7). This function gives a measure of the distance traveled if one traverses g from (x_i, y_i) to $(x_{i'}, y_{i'})$, which is reasonable for our application as we wish to cover as much area as possible.

$$\mu(i,z) = \begin{cases} \alpha \cdot \sum_{p=0}^{i'-i-1} E_2((x_{i+p}, y_{i+p}), (x_{i+p+1}, y_{i+p+1})), & \text{if } z = 0 \\ 0, & otherwise. \end{cases} \tag{2.7}$$

Cost Functions for Algorithms and Heuristics

The algorithms and heuristics we provide do not require the specific properties of cost functions given in section 2.2.2. It would be interesting in future research to determine if any properties of the cost functions given in section 2.2.2 could be exploited in some way to prove or improve guarantees or to modify our approaches. In general our approached will assume that the cost and fit functions follow the structure given in section 2.2, with the added restriction that for any interval $i \in g_{/\bar{G}}$ $\mu(i, 0)$ takes on any non-negative real value. This corresponds to rewarding coverage without insisting that the interval i be covered. When discussing algorithms and heuristics, we will restrict our discussion to situations in which $\mu(i, z) = 0$ for all $z \geq 1$. In this case, μ only differentiates between a solution that covers an interval and one that does not. We feel this choice of μ is most relevant for our application because once the surgeon has placed the bone segments, they may trim overlapping pieces with little additional time requirement. It should be noted that in some of the algorithms and heuristics given in chapter 4, one could modify this assumption to allow for other variations.

2.2.3 Summary of Notation

In this section we provide a summary of the notation we use to describe solutions of an instance $\mathcal{I}$ of (CRPR) or (CRPRPS).

Table 2.1: Relevant Notation

Symbol	Definition
J	The set of segments of the deformed curve cut by a solution S to (CRPR) or (CRPRPS).
J^*	The subset of segments of the deformed curve placed by a solution S to (CRPR) or (CRPRPS).
$\mathcal{P}_j$	The set of placements of a piece $j \in J$ that have finite cost.

2.3 Review

In this section, we review the necessary but widely known background knowledge.

2.3.1 Complexity

We will use this section to provide the necessary background material to show that we likely cannot find an efficient algorithm to solve instances of (CRPR) or (CRPSPS). We will provide an informal overview of the required definitions and theorems from complexity theory.

It will be important for us to describe a class of problems called *number problems*. We will illustrate this concept through examples. The definitions will be taken from [11]. First we consider the Hamilton Cycle Problem. In an instance of Hamiltonian cycle we are given a graph $G = (V, E)$, we ask whether or not there is an ordering $v_1, \ldots, v_n$ of V, the vertices of G, where $n = |V|$ such that $v_n v_1 \in E$ and $v_i v_{i+1} \in E$ for each $i \in [n-1]$. The second problem we consider is Partition. In an instance of Partition, we are given a finite set A and function $s : A \to \mathbb{Z}_{\geq 0}$, and ask whether there is some $A' \subseteq A$ such that

$$\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a).$$

The following is taken from [12]. Given an instance of a problem we will be interested in describing or encoding it using *bits*, which computers operate on. Bits take on values 0 and 1 and any number of N can be encoded in binary using a sequence of $log(N)$ bits. If a number N is encoded in unary it can be encoded in N bits. For a number N, we will use $poly(N)$ to mean some polynomial in N.

We return to our discussion of problems. In order to encode an instance $\mathcal{I}$ of Hamiltonian cycle we need only describe the set of vertices using the integers from 1 to n, where $n = |V|$, and since there are at most $\binom{n}{2}$ edges in E we require at most $O(n^2)$ numbers to encode $\mathcal{I}$, where each number is also bounded by n. On the other hand for an instance $\mathcal{I}'$ of Partition, we can encode A using n' numbers, where $n' = |A|$. However, $s(a)$ need not be bounded by $O(poly(n'))$.

Definition 12 (Number Problem)**.** *A problem is called a number problem if the numbers used to describe a generic instance can be arbitrarily large compared to the number of bits required to encode the instance in binary.*

Given a problem Π, we refer to the *input-size* of an instance $\mathcal{I}$ of Π as the number of bits n required to encode $\mathcal{I}$ in binary. We will also refer to the *numeric-size* of an instance $\mathcal{I}$ of Π as the number of bits

N required to encode $\mathcal{I}$ in unary. An algorithm taking an instance $\mathcal{I}$ of Π as input can be completed in *polynomial-time* if the number of steps required to complete the algorithm is $O(poly(n))$. An algorithm taking an instance $\mathcal{I}$ of Π as input can be completed in *pseudo-polynomial-time* if the number of steps required to complete the algorithm is $O(poly(N))$.

We now introduce the class of problems $\mathcal{NP}$.

Definition 13 (Decision Problem)**.** *A problem Π is a decision problem if for any instance $\mathcal{I}$ of Π, solving $\mathcal{I}$ can be posed as asking whether or not $\mathcal{I}$ is a* yes *or* no *instance of Π.*

Definition 14 (Efficient Certifier)**.** *A decision problem Π, with input-size n has an efficient certifier B, if B satisfies the following two conditions. First, B is algorithm takes as input an instance $\mathcal{I}$ of Π and a proof p and runs in $O(poly(n))$ number of steps. Second, $\mathcal{I}$ is a yes-instance of Π if and only if there exists a proof p, that is $O(poly(n))$, for which $B(\mathcal{I}, p)$ returns yes.*

The problems belonging to the class $\mathcal{NP}$, are precisely those decision problems with an efficient certifier. Next, we wish to classify the problems, which are $\mathcal{NP}$-complete.

Definition 15 (Polynomial-Time Reduction)**.** *A polynomial-time reduction from decision problem Π to a decision problem Π', is an algorithm that can solve instances $\mathcal{I}$ of Π with input-size n in $O(poly(n))$ steps, plus $O(poly(n))$ calls of a black-box algorithm, that solves any instance $\mathcal{I}'$ of Π' of input-size n' in $O(poly(n'))$ steps.*

For such Π and Π', we say that Π is polynomial-time reducible to Π'. The class of $\mathcal{NP}$-complete problems are those problems Π' that are members of $\mathcal{NP}$, and for all problems $\Pi \in \mathcal{NP}$, Π is polynomial-time reducible to Π'. It is widely accepted that there do not exist polynomial-time algorithms to solve $\mathcal{NP}$-complete problems.

The following is taken from [11]. A problem is said to be *strongly* $\mathcal{NP}$-complete if it remains $\mathcal{NP}$-complete, for instances $\mathcal{I}$, with numeric size N, that is bounded by some polynomial of its input size n. In other words, strongly $\mathcal{NP}$-complete problems remain $\mathcal{NP}$-complete even when they are encoded in unary.

One strongly $\mathcal{NP}$-complete problem is 3-Partition defined next.

Definition 16 (3-Partition)**.** *In an instance of 3-Partition the input is a set A of $3m$ elements for a positive integer m. We will also be presented with a weight function $s : A \to \mathbb{Z}_{\geq 0}$ such that $\frac{B}{4} < s(a) < \frac{B}{2}$ for all $a \in A$, such that $\sum_{a \in A} s(a) = mB$ where B is a positive integer. We wish to determine if there exists a partition of A into m sets $A_1, \dots, A_m$ such that for each $i \in \{1, \dots, m\}$ we have that $\sum_{a \in A_i} s(a) = B$. Note* *that the restriction $\frac{B}{4} < s(a) < \frac{B}{2}$ for all $a \in A$ implies that $|A_i| = 3$ for all $i \in [m]$. If such a partition exists, we say the instance is a yes-instance of 3-Partition, and a no-instance otherwise.*

Theorem 1. *3-Partition is strongly $\mathcal{NP}$-complete.*

To apply Theorem 1, we require the notion of a *pseudo-polynomial reduction*, which is formally defined in Definition 17.

Definition 17 (Pseudo-Polynomial-Time Reduction)**.** *A pseudo-polynomial-time reduction from decision problem Π to a decision problem Π', is an algorithm that can solve instances $\mathcal{I}$ of Π with numeric-size N in $O(poly(N))$ steps, plus $O(poly(N))$ calls of a black-box algorithm, that solves any instance $\mathcal{I}'$ of Π' of input-size n' in $O(poly(n'))$ steps.*

Lemma 1. *If Π is strongly $\mathcal{NP}$-complete, and $\Pi' \in \mathcal{NP}$, and there exists a pseudo-polynomial transformation from Π to Π', then Π' is strongly $\mathcal{NP}$-complete.*

Taking Π to be 3-Partition, we may then apply Lemma 1 to show that another problem Π' is strongly $\mathcal{NP}$-complete. We now provide an intuitive explanation. Note that one can scale the input and numeric sizes of instances by constants without affecting the above definitions. We see that a suitable quantity for the input-size n of an instance of 3-Partition is $mlog(B)$, while a suitable quantity for the numeric size N of an instance of 3-Partition is mB. Since 3-Partition, restricted to instances where $mB = O(poly(mlog(B))$ remains $\mathcal{NP}$-complete, to prove that a problem Π' is strongly $\mathcal{NP}$-complete it suffices to give a reduction from 3-Partition to Π' that is polynomial in quantity mB. In chapter 3 we will use these results to show that (CRPR) and (CRPRPS) are strongly $\mathcal{NP}$-complete.

2.3.2 Greedy Algorithms and Independence Systems

Definition 18 (Independence System)**.** *An independence system is a system $(E, \mathcal{I})$ where E is a set of elements referred to as the ground set, and $\mathcal{I} \subseteq 2^E$ satisfying the properties, $\emptyset \in \mathcal{I}$, and if $I \in \mathcal{I}$ and $I' \subseteq I$, then $I' \in \mathcal{I}$. The elements of $\mathcal{I}$ are referred to as independent sets.*

An element $I \in \mathcal{I}$ is called maximal if there is no element $e \in E \setminus I$ such that $I \cup e \in \mathcal{I}$.

Definition 19 (Basis)**.** *For $E' \subset E$, a basis of E' is a maximal independent set contained in E'.*

Definition 20 (Rank)**.** *For $E' \subset E$, the rank of E' is size of the largest basis of E', denoted $r(E')$.*

For $E' \subset E$, we let the quantity $\rho(E')$ denote the size of the smallest basis of E'

In this thesis we will reference the standard maximization problem on independence systems. In such a setting we are given an independence system $(E, \mathcal{I})$ and a weight function

$$w : E \to \mathbb{R},$$

and we wish to find an independent set $I \in \mathcal{I}$ that maximizes the quantity

$$w(I) = \sum_{e \in I} w(e).$$

Definition 21 (Rank Quotient)**.** *An independence system $(E, \mathcal{I})$ is said to have rank quotient*

$$k = \min_{E' \subset E, r(E') \neq 0} \frac{\rho(E')}{r(E')}.$$

The standard greedy algorithm, referred to as the *Best-In-Greedy Algorithm* in [13] is given in Algorithm 1.

Algorithm 1 $Best - In - Greedy((E, \mathcal{I}), w)$

```
1: Let $E = \{e_1, \ldots, e_n\}$, where $w(e_1) \geq \cdots \geq w(e_k)$
2: $I \leftarrow \emptyset$
3: for $i \in [n]$ do
4:     if $I \cup e_i \in \mathcal{I}$ then
5:         $I \leftarrow I \cup e_i$
   return $I$
```

Theorem 2. *For an independence system $(E, \mathcal{I})$ with rank quotient k, Algorithm 1 returns a solution I such that*

$$\frac{w(I)}{w(O)} \geq k,$$

where O is the optimal solution [14, 15]. Clearly such an algorithm can be implemented in polynomial-time, assuming that independence can be checked in polynomial time.

Note that in the case that an independence system has rank-quotient 1, Algorithm 1 will return an optimal solution. Such independence systems are called *Matroids*.

2.3.3 Local Search and Primal-Dual Algorithms

Local search and primal-dual algorithms are standard optimization techniques. In each iteration, local search algorithms explore a subset of a problem instance's solution space by moving from one solution to another that is "close." Defining solutions that are close to one another requires the concept of a *neighbourhood*. The neighbourhood of a solution S denoted $N(S)$ is usually defined by all the feasible solutions obtained from S by making some small changes. One way to ensure a local search algorithm terminates in polynomial time, is to require that each feasible solution must have a neighbourhood with polynomial size. One begins such an algorithm with a feasible solution S and checks if there is some $S' \in N(S)$, that has a better objective value. Once there is no local update to be made that can improve S, the algorithm terminates [12]. It is often necessary to ensure that the objective function's improvement is substantial after each update step [16]. If no such requirement is made, we may be faced with a situation where the number of iterations of local search is not bounded by any polynomial function in the problem's input-size.

Primal-dual algorithms are based on the fundamental concept of *complementary slackness*. For general instances of an $\mathcal{NP}$-complete problem, and any known practical integer programming formulation of it, one cannot obtain a solution to such an integer programming formulation that satisfies complementary slackness with a solution to the dual of its linear relaxation. When designing primal-dual algorithms, the general approach is to instead construct an integer feasible solution and a dual solution which satisfy a relaxed form of complementary slackness. One choice of relaxation is for the solutions to satisfy primal complementary slackness, where each non-zero primal variable has its corresponding dual constraint satisfied at equality [17].

Although local search algorithms are generally easy to design, as one only needs to define a neighbourhood, it is often difficult to prove any performance guarantees. In these situations, one cannot say anything about the quality of the local optimum that is attained. In section 4.3, we propose a local search heuristic, and in section 4.4 we offer an additional primal-dual heuristic aimed at reducing the likelihood that we reach a poor local optimum.

Chapter 3

Complexity

This section aims to prove that for a general instance of (CRPR), deciding whether of not there is a solution of cost M for any $M \geq 0$ is a strongly NP-complete problem. We will call this the decision version of (CRPR). We define the decision version of (CRPRPS) analogously. This result will also show that this problem is difficult to approximate. As a corollary, we will show that the same results apply to (CRPRPS).

We will give a pseudo-polynomial reduction from 3-Partition to (CRPR). For ease of understanding, we first offer an outline of the proof. Suppose we are given an instance $\mathcal{I}$ of 3-Partition with input parameters A, s, m, B, as in definition 16. We construct an instance $\mathcal{I}'$ of (CRPR) as follows. The deformed curve f will be created by translating and connecting curve segments at their endpoints with the following structure. For each element $z \in A$, we will include in f a curve segment that is a straight line of length $s(z)$. We will also include in f some additional segments that are not straight lines. The ideal curve g is constructed in a similar but modified fashion. It will include m copies of a straight line of length B. The additional segments used to construct g will be translations of the same set of additional segments used to construct f. The way we arrange our curve segments in each of f and g and assign the number of pieces we may select, k, will give us the property that any 0 cost solution for the instance $\mathcal{I}'$ will map directly to a 3-Partition of $\mathcal{I}$. Conversely, we will be able to map any 3-Partition of $\mathcal{I}$ to a 0 cost solution for $\mathcal{I}'$.

We now proceed with a formal description of the instance $\mathcal{I}'$. Fix an arbitrary ordering on A the elements of the 3-Partition instance. For $p \in [3m]$, let z_p be the p^{th} element of A. For simplicity add an additional element z_0 to A such that $s(z_0) = 0$. Select suitable rationals ϵ_1, ϵ_2 such that

$$1 < E_2((0,0),(1,\epsilon_1)) < E_2((0,0),(1,\epsilon_2)) < 2.$$

Throughout this construction the reader will find Figure 3.1 helpful.

We create a function η_f defined over $\{0\} \cup [m(B+8)]$ that we will use to define our discretization f. For a visualization of η_f, see the points in Figure 3.1 (i). Essentially, we assign each element of $\{0\} \cup [m(B+8)]$ in order a value by doing the following. We iterate through our set A, and when considering an element $z \in A$ we define η_f for the next $s(z)+2$ integers as follows. The first $s(z)+1$ integers will have value 0, and the final one will have value ϵ_1. Once we have finished iterating through A we continue to define η_f, alternating between setting $\eta_f(d)$ to ϵ_2 and setting $\eta_f(d)$ to 0 for the remaining $2m$ integers d over which η_f is defined.

Formally, let $\eta_f : \{0\} \cup [m(B+8)] \to \{0, \epsilon_1, \epsilon_2\}$ be defined in equation (3.1).

$$\eta_f(d) = \begin{cases} \epsilon_1, & \text{if } d = \sum_{p'=0}^{p} s(z_{p'}) + 2p - 1 \text{ for some } p \in [3m] \\ \epsilon_2, & \text{if } d = m(B+6) + 2q - 1 \text{ for some } q \in [m] \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

Take f to be the curve defined by the discretization

$$F = [(d, \eta_f(d)) : d \in \{0\} \cup [m(B+8)]. \tag{3.2}$$

See Figure 3.1 (i) for a visualization of the resulting deformed curve f, defined by F. Note that $\eta_f(d)$ is uniquely defined for each $d \in \{0\} \cup [m(B+8)]$.

We create a function η_g defined over $\{0\} \cup [m(B+8)]$ that we will use to define our discretization g. For a visualization of η_g, see the points in Figure 3.1 (ii). We assign each element of $\{0\} \cup [m(B+8)]$ in order a value by doing the following. First we complete the following process m times. We assign the next $B+1$ integers the value 0 after which the next integer receives the value ϵ_2. After this we continue to define η_g, alternating between setting $\eta_g(i)$ to ϵ_2 and setting $\eta_g(i)$ to 0 for the remaining $6m$ integers i over which η_g is defined. Formally, let $\eta_g : \{0\} \cup [m(B+8)] \to \{0, \epsilon_1, \epsilon_2\}$ be defined in equation (3.3).

$$\eta_g(i) = \begin{cases} \epsilon_2, & \text{if } i = qB + 2q - 1 \text{ for some } q \in [m] \\ \epsilon_1, & \text{if } i = m(B+2) + 2p - 1 \text{ for some } p \in [3m] \\ 0, & \text{otherwise.} \end{cases} \tag{3.3}$$

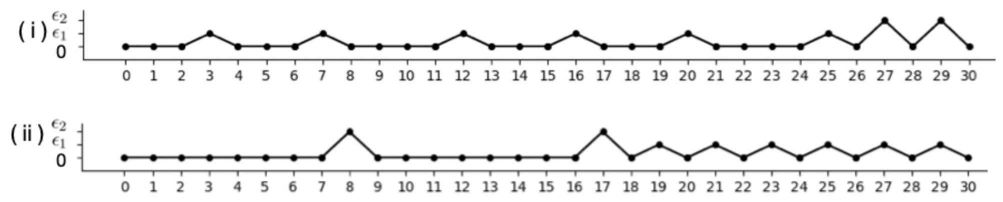

Figure 3.1: For an instance of 3-Partition with $m = 2, B = 7$ and a set A containing four elements of size 2 and two elements of size 3. (i) f, (ii) g

Take g to be the curve defined by the discretization

$$G = [(i, \eta_g(i)) : i \in \{0\} \cup [m(B+8)]]. \tag{3.4}$$

See 3.1 (ii) for a visualization of the resulting deformed curve g, defined by G. Finally, take our sets $\bar{G}$ and $\bar{F}$ that we use to define our unit segments to be $\bar{F} = F, \bar{G} = G$. We take $k = 7m$ to be the number of segments we may cut from the deformed curve. We take the fit and coverage functions described in Section 2.2.2, choosing parameter $\delta = 0$ for c, and some $\alpha > 0$, where $\alpha = O(mB)$ for μ. In particular when matching a curve segment f' of f to a curve segment g' of g, we take as our fit function c a volume difference estimation of the area between the curve obtained by matching f' to g' and g' itself. Also our coverage function μ will only penalize for unit segments $i \in g_{/\bar{G}}$ left uncovered by a solution, and it does so by scaling the Euclidean distance of the line segments left uncovered by α.

Following our notation from Section 2.1, we call the sets of curve segments $f_{/\bar{F}}$ and $g_{/\bar{G}}$ the unit segments of the deformed curve and the unit segments of the ideal curve, respectively. Recall that we have chosen to rename the elements of $f_{/\bar{F}}$ and $g_{/\bar{G}}$, by their indices. As a result we will we will select integers

$$d \in \{0\} \cup [m(B+8)-1]$$

to denote the d^{th} curve segment of $f_{/\bar{F}}$. Similarly, we will select integers

$$i \in \{0\} \cup [m(B+8)-1]$$

to denote the i^{th} curve segment of $g_{/\bar{G}}$.

We now partition $f_{/\bar{F}}$ and $g_{/\bar{G}}$ according to their length. The reader will find Figure 3.2 helpful throughout. For an element $d \in f_{/\bar{F}}$ we place d into one of $f_{\mathcal{P}}, f_{\mathcal{E}_1}$ and $f_{\mathcal{E}_2}$ as follows. If the endpoints $d, d+1 \in \bar{F}$

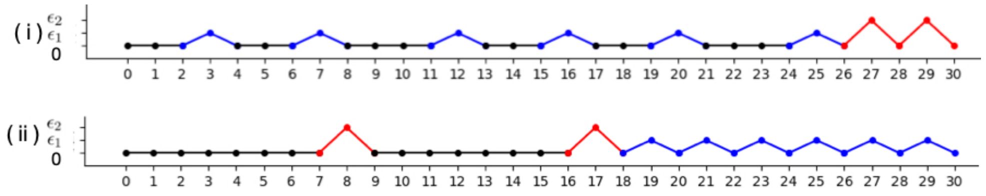

Figure 3.2: (i) $f_{\mathcal{P}}$ in black, $f_{\mathcal{E}_1}$ in blue, $f_{\mathcal{E}_2}$ in red, (ii) $g_{\mathcal{P}}$ in black, $g_{\mathcal{E}_1}$ in blue, $g_{\mathcal{E}_2}$ in red

of the curve segment $d \in f_{/\bar{F}}$ are such that $\eta_f(d) = \eta_f(d+1) = 0$ we put curve segment d into $f_{\mathcal{P}}$. If the endpoints $d, d+1 \in \bar{F}$ of the curve segment $d \in f_{/\bar{F}}$ are such that either $\eta_f(d) = \epsilon_1$ or $\eta_f(d+1) = \epsilon_1$ we put d into $f_{\mathcal{E}_1}$. Otherwise either $\eta_f(d) = \epsilon_2$ or $\eta_f(d+1) = \epsilon_2$ and we put d into $f_{\mathcal{E}_2}$. The analogous choices are made for $g_{\mathcal{E}_1}, g_{\mathcal{E}_2}, g_{\mathcal{P}}$. The formal equations are given in (3.5).

$$\begin{aligned}
f_{\mathcal{P}} &= \{d \in f_{/\bar{F}} : \eta_f(d) = \eta_f(d+1) = 0\} \\
f_{\mathcal{E}_1} &= \{d \in f_{/\bar{F}} : \eta_f(d) = \epsilon_1 \text{ or } \eta_f(d+1) = \epsilon_1\} \\
f_{\mathcal{E}_2} &= \big\{d \in f_{/\bar{F}} : \eta_f(d) = \epsilon_2 \text{ or } \eta_f(d+1) = \epsilon_2\big\} \\
g_{\mathcal{P}} &= \big\{i \in g_{/\bar{G}} : \eta_g(i) = \eta_g(i+1) = 0\big\} \\
g_{\mathcal{E}_1} &= \big\{i \in g_{/\bar{G}} : \eta_g(i) = \epsilon_1 \text{ or } \eta_g(i+1) = \epsilon_1\big\} \\
g_{\mathcal{E}_2} &= \big\{i \in g_{/\bar{G}} : \eta_g(i) = \epsilon_2 \text{ or } \eta_g(i+1) = \epsilon_2\big\}
\end{aligned} \tag{3.5}$$

At this point it may be helpful to the reader that in section 2.1.1 we defined for each curve h an ordering $\prec_h$ on the points in h. Since f and g are functions, for $h \in \{f, g\}$ we will have $(x, y) \prec_h (x', y')$ whenever $x < x'$.

For this proof, we say that a solution S to $\mathcal{I}'$ *maps* the curve segment f'' to the curve segment h if there is some curve segment f' that S matches to some curve segment g' of g that satisfies the following two properties. The first property is that there exists $(u, v), (u', v') \in f'$ such that $(u, v) \prec_f (u', v')$ and $f'' = f'|_{(u,v),(u',v')}$. To describe the second property we first need some notation. For any point $(u, v) \in f'$, we let $T((u, v))$ be the point (u, v) is transformed to after matching f' to g'. Using this notation, our final property states that $f'^{g'}|_{T((u,v)),T((u',v'))} = h$. To summarize, the curve segment f'' was transformed by S so that it aligns perfectly with h.

We first prove Lemma 2, which follows by our choice of δ and simple calculus. It argues that a 0-cost solution S will always map curve segments of f to curve segments of g.

Lemma 2. *Suppose that S is a 0-cost solution to $\mathcal{I}'$. Suppose that S matches a curve segment f' of f to a* *curve segment g' of g. Then we have that $f'^{g'}$ equals g'. In particular for any curve segment of g' there is a curve segment of f' that is mapped exactly to g'.*

Proof. Suppose that $f' = f|_{a,b}$ and $g' = g|_{\ell,r}$ for some $(a,b) \in \binom{\bar{F}}{2}, (\ell, r) \in \binom{\bar{G}}{2}$. Let h be the curve segment with discretization $[(0,0),(M,0)]$ for some $M > 0$. Since $\delta = 0$, we know that $E_2(a,b) = E_2(\ell, r)$ and thus when we match f' to h to get f'^h and g' to h to get g'^h we get that the left endpoints of f'^h and g'^h are equal. The same holds of the right endpoints of f'^h and g'^h. Our choice of cost function will in this case dictate that $c(a,b,\ell,r)$ is precisely the area between f'^h and g'^h. Since S is a 0-cost solution of $\mathcal{I}$, we know that $(c,b,\ell,r) = 0$ and hence $f'^h = g'^h$. Since $E_2(a,b) = E_2(\ell,r)$, the same transformation can be applied to $f'^{g'}$ and to g' to match them to h. The resulting curves are indeed f'^h and g'^h. As a result $f'^{g'} = g'$, and for any curve segment of g' there is a curve segment of f' that is mapped exactly to g'. □

It will be helpful for the reader to recall that in addition to using i to denote segment i of the deformed curve, we will also use i to denote point i in $\bar{G}$. It will be clear from the context which one we are using. When we wish to get the x and y-coordinates of point i we will call $x(i)$ and $y(i)$ respectively.

We now prove Lemma 3, which argues that any 0-cost solution will cover each element $i \in g_{\mathcal{E}_1}$, by mapping an element in $f_{\mathcal{E}_1}$ to i.

Lemma 3. *Suppose S is a 0-cost solution to $\mathcal{I}'$ and $i \in g_{\mathcal{E}_1}$. Then there exists some interval $d \in f_{\mathcal{E}_1}$, such that S maps interval d to interval i.*

Proof. Since S is a 0 cost solution to $\mathcal{I}'$, there is no contribution from the term

$$\sum_{i \in g_{/\bar{G}}} \mu(i, cov_{S'}(i)).$$

Since $\alpha > 0$, we conclude that $cov_S(i) \geq 1$ for all $i \in g_{/\bar{G}}$. Thus there exist some $(a,b) \in \bar{F}, (\ell, r) \in \bar{G}$, such that S matches $f' = f|_{a,b}$ to $g' = g|_{\ell,r}$, where $\ell \preceq_g i \prec_g i+1 \prec_g r$. By Lemma 2 we conclude that there exist $(u,v),(u',v') \in f'$ such that $f'|_{(u,v),(u',v')}$ maps to interval i.

Suppose that the lemma does not hold. Then $f'|_{(u,v),(u',v')}$ contains a curve segment of some $d' \in f_{\mathcal{P}} \cup f_{\mathcal{E}_2}$.

First suppose that $f'|_{(u,v),(u',v')}$ contains a curve segment of some $d' \in f_{\mathcal{E}_2}$. Since $a, b \in \bar{F}$ we know that

$$a \preceq_f d' \prec_f d' + 1 \preceq_f b.$$

Note that the Euclidean distance between the endpoints of interval i is $E_2((0,0),(1,\epsilon_1))$, but the Euclidean distance between the endpoints of interval d' is $E_2((0,0),(1,\epsilon_2))$. Since

$$E_2((0,0),(1,\epsilon_2)) > E_2((0,0),(1,\epsilon_1))$$

and the segments of g adjacent to i do not lie on the same line as i, it must be the case that either $f'|_{d',(u,v)}$ maps to a curve segment that is not a curve segment of g or $f'|_{(u',v'),d'+1}$ maps to a curve segment that is not a curve segment of g. In either case by Lemma 2 we conclude that no such d' can exist.

Now suppose that $f'|_{(u,v),(u',v')}$ contains a curve segment of some $d' \in f_{\mathcal{P}}$. If $a = d'$ and $b = d'+1$, then since the Euclidean distance between the endpoints of $f'|_{(u,v),(u',v')}$ is at most the Euclidean distance between the endpoints of curve segment d', which is 1, we conclude that $f'|_{(u,v),(u',v')}$ cannot map to the interval i. As a result either $a \preceq_f d'-1$ and $f'|_{(u,v),(u',v')}$ also contains a curve segment of $d'-1$ or $d'+2 \preceq_f b$ and $f'|_{(u,v),(u',v')}$ also contains a curve segment of $d'+1$.

In the first case, by construction of f we know that $d'-1 \in f_{\mathcal{P}} \cup f_{\mathcal{E}_1}$. If $d'-1 \in f_{\mathcal{P}}$ we observe that the Euclidean distance between the endpoints of interval i is $E_2((0,0),(1,\epsilon_1))$, but the Euclidean distance between the endpoints of $f'|_{d'-1,d'+1}$ is 2. Since

$$E_2((0,0),(1,\epsilon_1)) < 2$$

and the segments of g adjacent to i do not lie on the same line as i, it must be the case that either $f'|_{d'-1,(u,v)}$ maps to a curve that is not a curve segment of g or $f'|_{(u',v'),d'+1}$ maps to a curve that is not a curve segment of g. By Lemma 2 we conclude that $d'-1 \in f_{\mathcal{E}_1}$. Note that segment i is a straight line, and $f'|_{(u,v),(u',v')}$ is not because $f'|_{d'-1,d'+1}$ is not a straight line. Thus $a = d'$ and $d'+2 \preceq_f b$, where $f'|_{(u,v),(u',v')}$ also contains a curve segment of $d'+1$. A similar analysis shows that this also cannot be the case. Thus no such d' exists.

We conclude that $f'|_{(u,v),(u',v')}$ contains only curve segments which are elements of $f_{\mathcal{E}_1}$. If $f'|_{(u,v),(u',v')}$ contains curve segments of consecutive elements $d, d+1$ in $f_{\mathcal{E}_1}$, then $f'|_{(u,v),(u',v')}$ cannot be a straight line, again contradicting Lemma 2. We conclude that $f'|_{(u,v),(u',v')}$ only contains a curve segment of one interval $d \in f_{\mathcal{E}_1}$. Note that since $f'|_{(u,v),(u',v')}$ maps to i the endpoints of $f'|_{(u,v),(u',v')}$ are separated by a Euclidean distance of 1, which is also true of the endpoints of interval d. Thus $f'|_{(u,v),(u',v')}$ is equal to the curve segment d, as required. □

A similar argument to that in the proof of Lemma 3 can be used to to prove Lemma 4.

Lemma 4. *Suppose S is a 0-cost solution to $\mathcal{I}'$ and $i \in g_{\mathcal{E}_2}$. Then there exists some interval $d \in f_{\mathcal{E}_2}$, such* *that S maps interval d to interval i.*

As a result of Lemma 3 and Lemma 4, we get Corollary 1.

Corollary 1. *Suppose S is a 0-cost solution to $\mathcal{I}'$ and $i \in g_{\mathcal{P}}$. Then there exists some interval $d \in f_{\mathcal{P}}$, such that S maps interval d to interval i.*

Proof. By Lemma 3 each element of $g_{\mathcal{E}_1}$ has at least one segment of $f_{\mathcal{E}_1}$ mapped to it by S. Then since $|f_{\mathcal{E}_1}| = |g_{\mathcal{E}_1}|$, S cannot map any curve segment of some segment $d \in f_{\mathcal{E}_1}$ to a curve h that is not a segment of some element $i' \in g_{\mathcal{E}_1}$. An analogous statement can be made about the elements of $g_{\mathcal{E}_2}$ and $f_{\mathcal{E}_2}$. Since S is a 0-cost solution and $\alpha > 0$, there is some segment f'' of f which maps to segment i of $g_{\mathcal{P}}$. We know by our previous remarks that f'' contains exclusively elements of $f_{\mathcal{P}}$. Measuring Euclidean distances we see that S must map one segment from $f_{\mathcal{P}}$ to each segment of $g_{\mathcal{P}}$, as required. □

We now define some families of curve segments of f and g. A curve is an ϵ_1-triangle if it is a translation of a curve with discretization $[(0,0),(1,\epsilon_1),(2,0)]$. A curve is the left half of a ϵ_1*-triangle* if it is a translation of a curve with discretization $[(0,0),(1,\epsilon_1)]$. Similarly, a curve is the right half of a ϵ_1-triangle if it is a translation of a curve with discretization $[(0,\epsilon_1),(1,0)]$. We give the analogous definitions for ϵ_2-triangles.

We our now ready to prove the main result of this section, Theorem 3.

Theorem 3. *The decision version (CRPR) is strongly $\mathcal{NP}$-complete.*

Proof. Note first that for an instance (CRPR), feasibility of any solution can be verified in polynomial time because one must simply check that each pair of deformed curve segments given by the solution can be obtained using the same separation operation. Thus the decision version of (CRPR) is in $\mathcal{NP}$ as one must provide a solution with the claimed cost bound. The transformation from $\mathcal{I}$ to $\mathcal{I}'$ is computed in time $O(mB)$. As a result of our discussion in section 2.3.1, to show that (CRPR) is strongly $\mathcal{NP}$-complete it remains to show that $\mathcal{I}$ is a yes instance of 3-Partition if and only if $\mathcal{I}'$ has a zero-cost solution.

First suppose that $\mathcal{I}$ is a yes instance of 3-Partition. We create a solution S to $\mathcal{I}'$ of 0 cost. First, Let $(A_1,\ldots,A_m)$ be a 3-Partition of A. Then $\sum_{z\in A_q} s(z) = B$ for all $q \in [m]$. Now, for $q \in [m]$ let $A_i = \{z_1^q, z_2^q, z_3^q\}$. For simplicity let $s(z_0^q) = 0$ for $q \in [m]$. Assume that the ordering of A we fixed when we created $\mathcal{I}'$ was such that z_p equals z_t^q, where $p = 3(q-1)+t$ for $t \in [3]$. In other words we ordered A by first listing the elements of A_1, then the elements of A_2, and so on.

We will use $3m$ pieces from our budget to cut all possible ϵ_1-triangles from f, m pieces from our budget to cut all possible ϵ_2-triangles from f, and $3m$ pieces from our budget to cut segments that are lines of length

z for each $z \in A$. Formally, for $j \in \{0\} \cup [k-1]$ construct S by cutting f according to equations (3.6) and (3.7), and placing the separated pieces according to equations (3.8) and (3.9). The reader may again wish to consult Figure 3.2 for a visualization of how a, b, ℓ and r are selected. The pieces we cut are those that have the maximal with respect to the property of containing consecutive segments of the same colour. Our choice of ℓ and r is made so that our deformed pieces are mapped to ideal segments of the same colour.

$$a_j = \begin{cases} (\sum_{p'=0}^{\frac{j}{2}} s(z_{p'}) + j, 0) & j \text{ even }, 0 \leq j \leq 6m-1 \\ (\sum_{p'=0}^{\frac{j+1}{2}} s(z_{p'}) + j - 1, 0) & j \text{ odd }, 0 \leq j \leq 6m-1 \\ (m(B+6) + 2(j-6m), 0) & 6m \leq j \leq 7m-1 \end{cases} \tag{3.6}$$

$$b_j = \begin{cases} (\sum_{p'=0}^{\frac{j+2}{2}} s(z_{p'}) + j, 0) & j \text{ even }, 0 \leq j \leq 6m-1 \\ (\sum_{p'=0}^{\frac{j+1}{2}} s(z_{p'}) + j + 1, 0) & j \text{ odd }, 0 \leq j \leq 6m-1 \\ (m(B+6) + 2(j-6m+1), 0) & 6m \leq j \leq 7m-1 \end{cases} \tag{3.7}$$

$$\ell_j = \begin{cases} ((B+2)(q-1) + \sum_{t'=0}^{t-1} s(z_{t'}^q), 0) & j \text{ even }, 0 \leq j \leq 6m-1, z_{\frac{j}{2}} = z_t^q \\ (m(B+2) + j - 1, 0) & j \text{ odd }, 0 \leq j \leq 6m-1 \\ (B(j-6m+1) + 2(j-6m), 0) & 6m \leq j \leq 7m-1 \end{cases} \tag{3.8}$$

$$r_j = \begin{cases} ((B+2)(q-1) + \sum_{t'=0}^{t} s(z_{t'}^q), 0) & j \text{ even }, 0 \leq j \leq 6m-1, z_{\frac{j}{2}} = z_t^q \\ (m(B+2) + j + 1, 0) & j \text{ odd }, 0 \leq j \leq 6m-1 \\ (B+2)(j-6m+1), 0) & 6m \leq j \leq 7m-1 \end{cases} \tag{3.9}$$

To argue that $c(a_j, b_j, \ell_j, r_j) = 0$ for all $j \in \{0\} \cup [7m-1]$, we partition the pieces in $j \in \{0\} \cup [7m-1]$ into three groups in equation (3.10).

$$\begin{aligned} \mathcal{P} &= \{f|_{a_j,b_j} : j \text{ even }, 0 \leq j \leq 6m-1\} \\ \mathcal{E}_1 &= \{f|_{a_j,b_j} : j \text{ odd }, 0 \leq j \leq 6m-1\} \\ \mathcal{E}_2 &= \{f|_{a_j,b_j} : 6m \leq j \leq 7m-1\} \end{aligned} \tag{3.10}$$

If $j \in \mathcal{P}$ then by construction of f, $f|_{a_j,b_j}$ is a translation of the curve segment h with discretization $[(d,0) : d \in \{0\} \cup [s(z)]]$ for some $z \in A$. By construction of g, $g|_{\ell_j,r_j}$, is also a translation of h. Hence $c(a_j, b_j, \ell_j, r_j) = 0$. Now if $j \in \mathcal{E}_1$, it follows that segment j is an ϵ_1-triangle. By construction of g, the same is true of $g|_{\ell_j,r_j}$, hence $c(a_j,b_j,\ell_j,r_j)=0$. An analogous argument is made for $j\in\mathcal{E}_2$.

By our choice of S it follows that $cov_S(i)=1$ for all $i\in g_{/\bar{G}}$. Hence there is no contribution from the term

$$\sum_{i\in g_{/\bar{G}}}\mu(i,cov_S(i))$$

to the objective function. We have constructed a 0 cost solution to $\mathcal{I}'$ as required. With respect to Figure 3.2 again, S' must have selected the pieces that are maximal with respect to the property of containing consecutive segments of the same colour, and subsequently mapped them to segments of the same colour in the ideal curve.

Now suppose S' is a 0 cost solution to $\mathcal{I}'$, which places $k'\le k$ pieces. Let $\mathcal{E}_1'$ be the set of curve segments of f placed by S' that contain an element of $f_{\mathcal{E}_1}$ as a curve segment. Similarly we take $\mathcal{E}_2'$ to be the set of curve segments of f placed by S' that contain an element of $f_{\mathcal{E}_2}$ as a curve segment. Finally we take $\mathcal{P}'$ be the set of curve segments of f placed by S' that contain an element of $f_{\mathcal{P}}$ as a curve segment. By Lemma 3, Lemma 4 and Corollary 1 we know that S' maps elements of $f_{\mathcal{E}_1}$ to elements of $g_{\mathcal{E}_1}$, maps elements of $f_{\mathcal{E}_2}$ to elements of $g_{\mathcal{E}_2}$ and maps elements of $f_{\mathcal{P}}$ to elements of $g_{\mathcal{P}}$. By the way we ordered the curve segments differently in f than in g it follows that $\mathcal{E}_1'$, $\mathcal{E}_2'$ and $\mathcal{P}'$ must be disjoint. This means that any given curve segment cut by S contains curve segments from only one class.

Since $\mathcal{E}_1'$ is disjoint from both $\mathcal{E}_2'$ and $\mathcal{P}'$ and each segment of $f_{\mathcal{E}_1}$ is contained in precisely one piece of $\mathcal{E}_1'$ by our ordering of the unit segments in f we require at least $3m$ pieces in our budget to create the pieces in $\mathcal{E}_1'$. Similarly, since $\mathcal{E}_2'$ is disjoint from both $\mathcal{E}_1'$ and $\mathcal{P}'$ and each segment of $f_{\mathcal{E}_2}$ is contained in precisely one piece of $\mathcal{E}_2'$ by construction of f we require at least m pieces in our budget to create the pieces in $\mathcal{E}_2'$. Since

$$k'=|\mathcal{P}'|+|\mathcal{E}_1'|+|\mathcal{E}_1'|,$$

and $k'\le k$, $|\mathcal{E}_1'|\ge 3m$, $|\mathcal{E}_2'|\ge m$, we conclude that $|\mathcal{P}'|\le 3m$. By Corollary, 1 each segment of $f_{\mathcal{P}}$ is contained in precisely one piece of $\mathcal{P}$, and no other curve segments. By construction of f and the fact that $|\mathcal{P}'|\le 3m$, the only way this can hold is if

$$\mathcal{P}=\{f_{a,b}:a=\sum_{p'=0}^{p-1}s(z_p)+2(p-1),b=\sum_{p'=0}^{p}s(z_p)+2(p-1),p\in[3m]\}.$$

Denote the p^{th} curve segment in $\mathcal{P}$ by f_p. We now consider for $q\in[m]$ how S' covers the curve segment $g_{\ell,r}$

where

$$\ell = (q-1)(B+2), r = q(B+2) - 2.$$

We note that ℓ is the left endpoint of g or segment $\ell - 1$ is the right half of an ϵ_2-triangle. Similarly, segment r is the left half of an ϵ_2-triangle. As a result of our previous conclusion that S' can only map elements of $f_{\mathcal{P}}$ to elements of $f_{\mathcal{P}}$, we conclide any piece which covers elements a given element $\ell, \ldots, r-1$, must cover exclusively elements in $\{\ell, \ldots, r-1\}$. Since $cov_S(i) = 1$ for each $i \in \{\ell, \ldots, r-1\}$, and for each element $z \in A$, $\frac{B}{4} < s(z) < \frac{B}{2}$ there must exist some distinct $p, p', p'' \in [3m]$ such that S' matches $f_p, f_{p'}$ and $f_{p''}$ to $g_{\ell_p, r_p}, g_{\ell_{p'}, r_{p'}}$ and $g_{\ell_{p''}, r'_{p'}}$ respectively, where

$$\ell = \ell_p \prec_g r_p = \ell_{p'} \prec_g r_{p'} = \ell_{p''} \prec_g r_{p''} = r.$$

By the way we constructed $\mathcal{I}'$, if $A_q = \{z_p, z_{p'}, z_{p''}\}$ then $\sum_{z \in A_q} s(z) = x(r) - x(\ell) = B$. Hence $(A_1, \ldots, A_m)$ is a 3-Partition, and so $\mathcal{I}$ is a yes instance as required.

□

The proof of Theorem 3 implies that (CRPRPS) is also strongly $\mathcal{NP}$-hard, as even if we were presented with the pieces in $\mathcal{P} \cup \mathcal{E}_1 \cup \mathcal{E}_2$ we could show that there exists a 0 cost solution to the specified instance of (CRPRPS) if and only if $\mathcal{I}$ is a yes instance of 3-Partition. As a result we obtain Corollary 2.

Corollary 2. *The decision version of (CRPRPS) problem is strongly $\mathcal{NP}$-hard.*

In Theorem 3, we proved that $\mathcal{I}'$ has a 0-cost solution if and only if $\mathcal{I}$ was a yes instance of 3-Partition. From this, we conclude that if we could obtain any polynomially-computable approximation factor, for either (CRPR) or (CRPRPS), we could distinguish between *yes* and *no* instances of 3-Partition. Hence we also obtain Corollary 3.

Corollary 3. *There is no $\nu(|G|, |F|)$-approximation algorithm for the (CRPR) or (CRPRPS) problem unless $\mathcal{P} = \mathcal{NP}$, where ν is any function of $|G|, |F|$ that is computable in polynomial time.*

Note that when examining our proof of Theorem 3 the reader should notice the following. For choice of ϵ_1, ϵ_2 sufficiently small we constructed our deformed curve f in such a way that for any unit segments $d, d' \in f_{/\bar{F}}$ we have that $E_2(d, d+1)$ is approximately $E_2(d', d'+1)$. This is desirable as the surgeon can select any location along the bandeau to make incisions. To model this property, we want to select a set $\bar{F}$ so that the unit segments of the deformed curve are of approximately equal length and shape. The analogous property holds for our choice of g. This is also desirable because the surgeon may select any location on the template to place the pieces they have cut. We also selected $\bar{F} = F$ and $\bar{G} = G$ which again models the property that they surgeon may cut anywhere along the deformed curve, and place anywhere along the template. Another desirable property we will enforce in our instance of (CRPR) is that the Euclidean distance between the endpoints of f and the endpoints of g is the same. In practice, the normative skull shapes will be similar in size to the infant's deformed skull, and we wanted our reduction to model this. If we did not wish to capture these structures in our proof it could be simplified.

3.1 Conclusions from Hardness Results

Note that in our proof of Theorem 3, we used a specific cost function we find reasonable for our application. The same result also implies hardness for (CRPR) and (CRPRPS) with more general cost and fit functions. The specific properties required for our proof were that we set $\mu(i, 0) > 0$ for all $i \in g_{/\bar{G}}$, and that our choice of cost function gave us the property that whenever $\mathcal{I}$ was a yes instance of 3-Partition we were able to bound the value of the optimal solution to $\mathcal{I}$ from above by η, and whenever $\mathcal{I}$ was a no instance of 3-Partition we were able to bound the value of the optimal solution to $\mathcal{I}$ from below by η'. The inapproximability in Corollary 3 followed because the ratio between η' and η was unbounded.

It should be noted that in many instances that interest us, there are no curve segments of f that will fit perfectly to a curve segment of g. These (CRPR) instances emit a tighter lower bound than 0 for C_{min}. We could modify our instance $\mathcal{I}'$ used in the reduction differently. Restricted to this class of instances, the inapproximability result would depend on the value of $\frac{\eta'}{\eta}$. It would be of interest to further characterize the class of instances of (CRPR) and (CRPSPS) that are of interest for our application, to see if we can weaken the inapproximabilty result for this class. Another approach would be to assume certain properties of our ideal curve prohibiting our proof of Theorem 3 altogether. Such an example would be to assume our ideal curve is convex.

As mentioned in section 1, we are studying (CRPR) to gain an understanding of the cranial vault remodeling procedure. Given a similar modeling framework to that of (CRPR) for the cranial vault remodeling procedure, it is clear that this problem would be at least as hard. The goal of our research is to find an algorithm that performs well and runs in polynomial-time with respect to input size. In light of the inapproximability result, we decide to alter our objective function and leave a study of restricted classes of (CRPR) and (CRPSPS) to future research. In section 4 we take an affine transformation of our current objective function. In this setting, our inapproximability results do not hold, although $\mathcal{NP}$-hardness does. We will study algorithms and heuristics for (CRPR) and or (CRPRPS), concluding with an experimental study to see how these techniques perform on instances of interest.

Chapter 4

Maximization Objective

Note that we wish to design an objective function that balances our goal of covering as much area of the template, with our goal to place pieces that fit to the template as best as possible. For a solution S to an instance $\mathcal{I}$ of (CRPR) or (CRPRPS) we can view the problem of optimizing the fit and coverage of a solution as a maximization problem as follows. Recall that our coverage function μ takes as input a unit interval i of the ideal curve, and a non-negative integer z, and returns the penalty associated with covering interval i z times. Thus $\mu(i,0)$ is a measure of how undesirable it is to leave interval i exposed. When we cover an interval i, we can view the situation as collecting the reward $\mu(i,0)$. If S covers interval i multiple times, we can view the situation as collecting the reward $\mu(i,0)$, but paying a penalty $\mu(i, cov_S(i))$ for covering i too many times. Since $\mu(i,1) = 0$ for all unit intervals i in $g_{/\bar{G}}$, we can view S as collecting the reward $\mu(i,0) - \mu(i, cov_S(i))$ for each interval $i \in g_{/\bar{G}}$. We are also interested in the fit of our placements as dictated by c. For each curve segment $f|_{a,b}$ of f that S chooses to match to a curve segment $g|_{\ell,r}$ of g, we pay a penalty $c(a,b,\ell,r)$ for lack of fit. This gives us the objective function in (4.1), where the second inequality in (4.1) follows by our definition of C_{min} in equation (2.4).

$$C_{max} = \sum_{i \in \mathcal{I}} \mu(i,0) - C_{min} = \sum_{i: cov_S(i) \geq 1} (\mu(i,0) - \mu(i, cov_S(i))) - \sum_{j \in J} c(a_j, b_j, \ell_j, r_j). \tag{4.1}$$

Note that this transformation preserves optimality. This objective function can be interpreted as obtaining a reward for covering intervals, and paying penalties for over-covering and lack of fit. In the remaining sections we will assume we are optimizing over equation 4.1. We will assume our instances have cost functions fitting the description in section 2.2.2, namely that c is non-negative, for any $i \in g_{/\bar{G}}$, $\mu(i,0)$ and $\mu(i,z) = 0$

for all $z \geq 2$. For the remainder of this section, unless otherwise specified assume we are working only with instances satisfying these properties.

We summarize the example presented in [18] as a motivation for studying an affine transformations of our objective function where a strong inapproximability result may hold. Consider the Minimum Vertex Cover problem and Maximum Independent Set problem. In the Minimum Vertex Cover problem we are given a graph $G = (V, E)$ and we wish to find a subset of vertices V' of smallest cardinality such that for each edge $uv \in E$ at least one of u or v is contained in V'. In the Maximum Independent Set problem we are given a graph $G = (V, E)$ and we wish to find a subset of vertices V' of largest cardinality such that for each edge $uv \in E$ at most one of u or v is contained in V'. Both problems are $\mathcal{NP}$-complete [11]. We see from our problem definitions that if V' is a minimum vertex cover, then $V \setminus V'$ is a maximum independent set and vice versa. Despite this equivalence the approximability results for each vary, with Minimum Vertex Cover admitting constanct factor approximation ratios, while Maximum Independent Set is inapproximable for any ratio $\Omega(n^{1-\epsilon})$ for any $\epsilon > 0$, where $n = |V|$ [19].

4.1 Greedy

Recall from Section 2.3.2 that an independence system consists of a ground set E and a list of subsets $\mathcal{I}$ of E called independent sets. In this section, we will show that (CRPR) and (CRPRPS) can be reduced to an independence system with rank quotient $\frac{1}{k}$. Using the standard Best In Greedy algorithm discussed in section 2.3.2 we can get a k-approximation for both problems. We then modify this algorithm, providing a heuristic to get better performance for (CRPR) instances at the expense of run-time.

In this and the following section, we will find Lemma 5 useful.

Lemma 5. *Let S be a solution to an instance $\mathcal{I}$ of (CRPRPS). If S is optimal, then for any piece $j \in J$ used by S, we may assume there is some interval $i \in g_{/\bar{G}}$ that only piece j covers. Also, we may assume that there exist at most two pieces j', j'' used by S that interfere with the placement of j. Furthermore, pieces j' and j'' do not interfere with each other.*

Proof. Let J' be the set of pieces placed by S. Suppose that there is a piece j that does not uniquely cover any intervals $i \in g_{/\bar{G}}$. Then

$$\begin{aligned} C_{max}(S) &= \sum_{i: cov_S(i) \geq 1} (\mu(i,0) - \mu(i, cov_S(i))) - \sum_{j' \in J'} c(j, \ell_{j'}, r_{j'}) && \text{(4.2a)} \\ &\leq \sum_{i: cov_S(i) \geq 1} \mu(i,0) - \sum_{j' \in J'} c(j, \ell_{j'}, r_{j'}) && \text{(4.2b)} \\ &\leq \sum_{i: cov_S(i) \geq 1} \mu(i,0) - \sum_{j' \in J' \setminus \{j\}} c(j, \ell_{j'}, r_{j'}), && \text{(4.2c)} \end{aligned}$$

where (4.2a) follows by definition, (4.2b) follows because μ is non-negative, and (4.2c) because c is non-negative, and piece j does not uniquely cover any intervals. Thus we can remove piece j from S to get a solution at least as good.

To prove the remaining part of the lemma, suppose that there is a piece j that interferes with three pieces j', j'' and j'''. Let $(\ell, r), (\ell', r'), (\ell'', r'')$ and (ℓ''', r'''), respectively, be the locations at which S places pieces j, j', j'' and j''' along g. We may assume that each of j, j', j'', j''' uniquely covers at least one interval.

Let j_0 be a piece in $\{j, j', j'', j'''\}$ whose placement on g has the smallest left endpoint with respect to the ordering $\prec_g$. Similarly let j_3 be a piece in $\{j, j', j'', j'''\}$ whose placement in S has the largest right end point with respect to the ordering $\prec_g$. If j_0 and j_3 are not distinct, then the other three elements of $\{j, j', j'', j'''\}$ do not uniquely cover any ideal intervals, and we may delete them from S to obtain a solution that is at least as good. As a result, we assume that j_0 and j_3 are distinct. If j_0 and j_3 interfere with one another, we may delete the other two placements in $\{j, j', j'', j'''\}$ as they do not uniquely cover any unit ideal intervals. We conclude that j_0 and j_3 do not interfere with one another. Thus $j \notin \{j_0, j_3\}$. Since j interferes with j_0 and j_3, by choice of j_0 and j_3, it must be the case that the remaining piece does not uniquely cover any intervals, and so we can delete it to obtain a solution that is at least as good. □

As a corollary to Lemma 5, we obtain the same result for any instance of (CRPR) because once we have decided which pieces to cut, we are left with an instance of (CRPRPS).

Corollary 4. *Let S be a solution to an instance $\mathcal{I}$ of (CRPR). If S is optimal, then for any piece $j \in J$ used by S, we may assume there is some interval $i \in g_{/\bar{G}}$ that only piece j covers. Also, we may assume that there exist at most two pieces j', j'' used by S that interfere with piece j. Furthermore, pieces j' and j'' do not interfere with each other.*

Using these results, we will reduce both (CRPR) and (CRPRPS) as maximization problems over an independence system.

First let $\mathcal{I}$ be any instance of (CRPR) such that for any $i \in g_{/\bar{G}}$, $\mu(i,0)$ and $\mu(i,z) = 0$ for all $z \geq 2$. Suppose that $p_1 = (a_1, b_1, \ell_1, r_1)$ and $p_2 = (a_2, b_2, \ell_2, r_2)$ are two placements. If $f|_{a_1,b_1}$ and $f|_{a_2,b_2}$ contain some common portion of the deformed curve, then they cannot be formed by the same separation operation and any solution containing both p_1 and p_2 is infeasible.

Note that if a solution S covers some interval $i \in g_{/\bar{G}}$, multiple times then since S can only collect the reward $\mu(i,0)$ once we can view the situation as having a single piece take credit for covering i, when the rest do not. Now, as a result of Corollary 4, there must exist an optimal solution S to $\mathcal{I}$, such that for each piece j that S places the following holds. Suppose that S places j over curve segment $g|_{\ell,r}$ for some $(\ell, r) \in \mathcal{P}_j$. There must exist some $(\bar{\ell}, \bar{r}) \in \binom{\bar{G}}{2}$ such that $\ell \preceq_g \bar{\ell} \prec_g \bar{r} \preceq_g r$, where piece j uniquely covers all of $\bar{\ell}, \ldots, \bar{r} - 1$. Furthermore, if $\ell \prec_g \bar{\ell}$, then there is one piece j' that also covers all of $\ell, \ldots \bar{\ell} - 1$, and no other piece covers any intervals in $\{\ell, \ldots \bar{\ell} - 1\}$. Similarly, if $\bar{r} \prec_g r$, then there is one piece j'' that also covers all of $\bar{r}, \ldots r - 1$, and no other piece covers any intervals in $\{r, \ldots \bar{r} - 1\}$. As a result we may pick some point ℓ' such that $\ell \preceq_g \ell' \preceq_g \bar{\ell}$ such that piece j' receives credit for covering all the intervals in $\ell, \ldots, \ell' - 1$ and piece j receives credit for covering all the intervals in $\ell', \ldots, \bar{\ell}$. Similarly, we may pick some point r' such that $\bar{r} \preceq_g r' \preceq_g r$ such that j'' receives credit for covering all the intervals in $r', \ldots, r - 1$ and piece j receives credit for covering all the intervals in $\bar{r}, \ldots, r' - 1$. In addition piece j will receive credit for covering all intervals it uniquely covers.

Motivated by these observations we create our ground set E in equation (4.3), where element $(a, b, \ell, r, \ell', r')$ has weight

$$w((a, b, \ell, r, \ell', r')) = \sum_{i : \ell' \preceq_g i \prec_g r' - 1} \mu(i, 0) - c(a, b, \ell, r).$$

The element $(a, , b, \ell, r, \ell', r')$ corresponds to the decision of matching $f|_{a,b}$ to $g|_{\ell,r}$ but only receiving credit for covering the intervals $\ell', \ldots, r' - 1$. Let

$$E = \{(a, b, \ell, r, \ell', r') : (a, b) \in \binom{\bar{F}}{2}, (\ell, r) \in \binom{\bar{G}}{2}, \ell \preceq_g \ell' \prec_g r' \preceq_g r\}. \tag{4.3}$$

We say that two elements $(a_1, b_1, \ell_1, r_1, \ell_1', r_1'), (a_2, b_2, \ell_2, r_2, \ell_2', r_2') \in E$ are *compatible* if $f|_{a_1,b_1}$ and $f|_{a_2,b_2}$ can both be formed by the same separation operation on f , and if $g|_{\ell_1', r_1'}$ and $g|_{\ell_2', r_2'}$ can both be formed by the same separation operation on g. Then we take,

$$\mathcal{J} = \{S \subseteq 2^E : |S| \leq k, \text{ and for } e_1, e_2 \in S, e_1, e_2 \text{ compatible}\}.$$

As is standard, we let the weight of any independent set $I \in \mathcal{J}$, $w(I)$, be the sum of the weights of all elements in I.

We tie our reduction together with Claim 1. This claim is sufficient to prove our reduction from (CRPR) to Maximum Weight Independent Set because the highest weight independent set is mapped to a solution S of (CRPR), which must also be optimal.

Claim 1. *We can partition the independent sets in $\mathcal{J}$ into classes $\mathcal{C}_S$ for each $S \in \mathcal{S}$ that satisfy the following. Each independent set $I \in \mathcal{C}_S$ has weight $w(I)$ at most $C_{max}(S)$. Also for each $S \in \mathcal{S}$, there is some independent set $I_S^* \in \mathcal{C}_S$ with weight $w(I_S^*)$ equal to $C_{max}(S)$.*

Proof. Let $I \in \mathcal{J}$. Then $I = \{(a_0, b_0, \ell_0, r_0, \ell'_0, r'_0), \ldots, (a_{k'-1}, b_{k'-1}, \ell_{k'-1}, r_{k'-1}, \ell'_{k'-1}, r'_{k'-1})\}$, for some $k' \leq k$. Since I is independent we know that we can use the same separation operation to form all of the pieces $f|_{a_0,b_0}, \ldots, f|_{a_{k'-1},b_{k'-1}}$ of the deformed curve and the unit intervals $\ell'_0, \ldots, r'_0 - 1, \ldots, \ell'_{k'-1}, \ldots, r'_{k'-1} - 1$ are distinct. Consider the solution S which contains the placements $(a_0, b_0, \ell_0, r_0), \ldots, (a_{k'-1}, b_{k'-1}, \ell'_{k'-1}, r'_{k'-1})$. Since

$$\ell_j \preceq_g \ell'_j \prec_g r'_j \preceq_g r_j$$

for all $j \in \{0\} \cup [k'-1]$ we know that S covers all intervals $\ell'_0, \ldots, r'_0 - 1, \ldots, \ell'_{k'-1}, \ldots, r'_{k'-1} - 1$. Thus

$$C_{max}(S) \geq \sum_{j \in \{0\} \cup [k'-1]} \left(\sum_{i : \ell'_j \preceq_g i \prec_g r'_j - 1} \mu(i, 0) - c(a_j, b_j, \ell_j, r_j) \right) = w(I).$$

Now, let $S \in \mathcal{S}$. Suppose S uses the placements $(a_0, b_0, \ell_0, r_0), \ldots, (a_{k'-1}, b_{k'-1}, \ell'_{k'-1}, r'_{k'-1})$ for some $k' \leq k$. Assume that indices chosen for the k' pieces are non-decreasing with respect to their left placement point on the ideal curve in S. In other words, $\ell_0 \preceq_g \cdots \preceq_g \ell_{k'-1}$. Then for $j \in \{0\} \cup [k'-1]$ take

$$\ell'_j = \ell_j$$

and

$$r'_j = \begin{cases} \min(r_j, \ell_{j+1}) & j \in \{0\} \cup [k'-2] \\ r_j & j = k-1 \end{cases}.$$

Note that $\ell'_0 \prec_g r'_0 \preceq_g \ell'_1 \cdots \preceq_g \ell'_{k'-1} \prec_g r_{k'-1}$, by our choice of reindexing, and the fact that each piece uniquely covers at least one interval.

We can take I_S^* to be set

$$\{(a_0, b_0, \ell_0, r_0, \ell'_0, r'_0), \ldots, (a_{k'-1}, b_{k'-1}, \ell_{k'-1}, r_{k'-1}, \ell'_{k'-1}, r'_{k'-1})\}.$$

Since S is feasible the curve segments $f|_{a_0,b_0}, \ldots, f|_{a_{k'-1},b_{k'-1}}$ can be formed by the same separation operation. Thus each pair of elements in I_S^* is compatible by our choice of $\ell'_0, r'_0, \ell'_1 \ldots, \ell'_{k'-1}, r_{k'-1}$.

$\square$

If we wish to model an instance of (CRPRPS) using an independence system, we instead take,

$$E = \{(a, b, \ell, r, \ell', r') : a = a_j, b = b_j \text{ for } j \in J, (a, b) \in \binom{\bar{F}}{2}, (\ell, r) \in \binom{\bar{G}}{2}, \ell \preceq_g \ell' \prec_g r' \preceq_g r\},$$

and offer the analogous proofs.

The system $(E, \mathcal{J})$ is an independence system because $\emptyset \in \mathcal{I}$ and if any subset I of the ground set is such that each pair of elements is pairwise compatible, then any subset of I will also have the same property. Note $\mathcal{J}$ may contain independent sets of cardinality 1. For example, the set $\{(L(f), R(f), L(g), R(g), L(g), R(g))\}$ is maximal, as there are no other pieces that can be formed by the same separation operation as $f|_{L(f),R(f)}$. In general, one may be able to give a slightly better bound on $\rho(E')$ for some $E' \subseteq E$, by eliminating certain placements that clearly cannot be in an optimal solution. We do not assume this here. Also, in general, the rank of our independence system will be k as there will be many feasible solutions S placing k pieces. Thus for both problems, we conclude that $(E, \mathcal{J})$ is an independence system with rank quotient $\frac{1}{k}$. The above transformation is done in time $O(|\bar{F}|^2||\bar{G}|^4)$.

By theorem 2, Algorithm 1 gives a $k-approximation$ for both (CRPR) and (CRPRPS). However, suppose we have two instances of (CRPR) with identical input except the choice of k_1 and k_2, where $k_1 < k_2$. Then after k_1 rounds of Algorithm 1 for either choice of k, we will have the same solution. This is because the ground set E we construct will be identical with identical weights for each instance. We note that if we are running algorithm 1 and we match $f|_{a,b}$ to $g|_{\ell,r}$ where $r - \ell$ is particularly large we will get a large contribution from the term $\sum_{i:\ell \leq i \leq r-1} \mu(i, 0)$. Although if k is large this greedy move may not produce a good solution. To see this we illustrate with an example. Suppose we are given an instance of (CRPR) with the curves in Figure 4.1, where $k = 2$, $\bar{F} = \{0, 1, 2\}$, and $\bar{G} = \{0, 1, 2\}$. Suppose δ is large enough so that

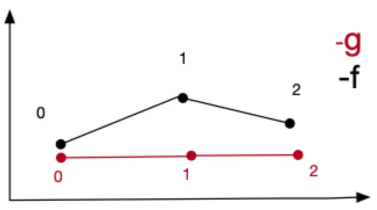

Figure 4.1: Illustration of an Instance where standard greedy algorithm performs poorly

$c(0,1,0,1) = c(1,2,1,2) = 0$ and $0 < c(0,2,0,2) < \infty$, and that $\mu(0,0), \mu(1,0)$ are chosen so that

$$\mu(0,0) + \mu(1,0) - c(0,2,0,2)$$

is greater than both $\mu(0,0) - c(0,1,0,1)$ and $\mu(1,0) - c(1,2,1,2)$. The standard greedy approach would pick the solution containing the placement $(0,2,0,2)$, even though we would suspect with the option to pick k pieces we may want to choose the solution with placements $(0,1,0,1)$ and $(1,2,1,2)$, which is indeed optimal.

We offer the heuristic in 2 to overcome this. In addition to the input parameters associated with an instance $\mathcal{I}$ of (CRPR), the algorithm will take as input a parameter $1 \leq m \leq k$. Our heuristic will run a modified greedy algorithm for each element $(q_1, q_2, \ldots, q_m) \in \binom{[|G|-1]}{m}$. Suppose we are constructing our solution, and it contains $m'-1$ elements for $m' \in [m]$. When selecting a possible m'^{th} element, we limit our choice to elements in our ground set corresponding to placements covering at most $q_{m'}$ intervals of the ideal curve. For $m' > m$, we will limit choice to elements in our ground set corresponding to placements covering at most q_m intervals of the ideal curve. We then check all solutions produced and select the best one. As m increases, we reduce the likelihood of selecting a piece that covers a large area but fits relatively poorly. The run-time of this algorithm is $O(|\bar{F}^2||\bar{G}|^{m+4})$.

4.2 Resource Allocation Problem

In [10], the authors introduce a general model for Resource Allocation problems, which we will formally define and review. We will see how we can reduce an instance of (CRPRPS) to a resource allocation problem, and then apply an algorithm in [10] to obtain a $\frac{1}{2}$-approximation for (CRPRPS). We will follow up with some suggestions for future research.

Algorithm 2 *SizeLimitingGreedy*$(\mathcal{I}, m)$

Input: An instance $\mathcal{I}$ of (CRPR) and in integer $m \leq k$.
Output: A solution S to (CRPR).

```
1: Initialize $E = \{e_1, \ldots, e_n\}$ as in equation 4.3, where $w(e_1) \geq \ldots w(e_n)$.
2: $S \leftarrow \emptyset$
3: $Q \leftarrow \{(q_1, q_2, \ldots, q_m) \in \binom{[|G|-1]}{m}\}$
4: for $(q_1, q_2, \ldots, q_m) \in Q$ do
5:     $I \leftarrow \emptyset$
6:     for $i \in [n]$ do
7:         $(a, b, \ell, r, \ell', r') \leftarrow e_i$
8:         if $|I| = k' - 1 < m$ then
9:             if $r - \ell \leq q_{k'}$ then
10:                if $I \cup e_i \in \mathcal{I}$ then
11:                    $I \leftarrow I \cup e_i$
12:        else if $r - \ell \leq q_m$ then
13:            if $I \cup e_i \in \mathcal{I}$ then
14:                $I \leftarrow I \cup e_i$
15:    if $w(I) \geq w(S)$ then
16:        $S \leftarrow I$
    return $S$
```

4.2.1 Review of Resource Allocation Problem

A Resource Allocation problem takes as input a set of n activities. Each activity $j \in [n]$ is associated with a set of time intervals $\mathcal{A}_j$, over which it can potentially be scheduled. We call each such time interval an instance of activity j. Specifically, each instance $I \in \mathcal{A}_j$ occurs over an interval of time $[s(I), e(I)]$, such that $0 \leq s(I) < e(I) \leq T$. In addition we are presented with profit and weight functions

$$p(I) : \bigcup_{j=1}^{n} \mathcal{A}_j \to \mathbb{R}$$

and

$$w(I) : \bigcup_{j=1}^{n} \mathcal{A}_j \to [0, 1],$$

respectively. We say that an instance I of activity j occurs over a time interval $[t_1, t_2] \subseteq [0, T]$, if $s(I) \leq t_1 < t_2 \leq e(I)$. We wish to schedule a subset of activity instances, such that at most one instance is chosen from each activity, and for any given time interval $t = [t_1, t_2] \subseteq [0, T]$ of positive length, the combined weight of all scheduled instances occurring over t is at most 1. Amongst all such feasible schedules S we wish to select one that maximizes

$$p(S) := \sum_{I \in S} p(I).$$

We present the primal-dual framework discussed in [10]. Let $\mathcal{T}$ be the set of time intervals contained in $[0,T]$ that are maximal with respect to the property of not having any activity instance start or end during the middle of the time interval. More formally,

$$\begin{aligned}\mathcal{T} = \{[t_1,t_2] \subseteq [0,T] : &\nexists I \in \bigcup_{j=1}^{n} \mathcal{A}_j, t_1 < s(I) < t_2 \text{ or } t_1 < e(I) < t_2,\\ &\exists I', I'' \in \bigcup_{j=1}^{n} \mathcal{A}_j \text{ such that } t_1 \in \{s(I'), e(I')\}, t_2 \in \{s(I''), e(I'')\}\}\end{aligned} \tag{4.4}$$

Using this definition of $\mathcal{T}$ we can write down the integer programming formulation in (4.5) for an instance of a Resource Allocation problem.

For each instance I of an activity we create a variable x_I such that x_I is 1 if I is chosen for our schedule, and x_I is 0 otherwise. The dual of the linear programming relaxation to (4.5) is given in (4.6). We provide an interpretation of the dual variables. Each instance $I \in \bigcup_{j\in[n]}\bigcup_{I\in A_j}$, is associated with a profit $p(I)$. We wish to assign cost shares y_j and z_t, respectively, to each activity $j \in [n]$ and interval $t \in \mathcal{T}$ with the following property. If I is an instance of activity j, with weight $w(I)$, then the cost share y_j plus the sum of the cost shares z_t for intervals over which I occurs weighted by $w(I)$ is enough to pay the cost $p(I)$.

$$\max \sum_{j\in[n]}\sum_{I\in A_j} p(I)x_I \tag{4.5a}$$

$$\text{s.t.} \sum_{I:s(I)\le t_1<t_2\le e(I)} w(I)\cdot x_I \le 1 \quad \forall t=[t_1,t_2]\in\mathcal{T} \tag{4.5b}$$

$$\sum_{I\in\mathcal{A}_j} x_I \le 1 \quad \forall j\in[n] \tag{4.5c}$$

$$x_I \in \{0,1\} \quad \forall I\in\mathcal{A}_j \quad \forall j\in[n] \tag{4.5d}$$

$$\min \sum_{j\in[n]} y_j + \sum_{t\in\mathcal{T}} z_t \tag{4.6a}$$

$$\text{s.t.} \quad y_j + \sum_{t=[t_1,t_2]\in\mathcal{T}:s(I)\le t_1<t_2\le e(I)} w(I)\cdot z_t \ge p(I) \quad \forall I\in\mathcal{A}_j \quad \forall j\in[n] \tag{4.6b}$$

$$y, z \ge 0 \tag{4.6c}$$

Definition 22 (Interferes)**.** *We say that instance I interferes with instance I' if $[s(I), e(I)] \cap [s(I'), e(I')] \neq \emptyset$.*

The reader should note how the definition of interference is analogous to the notion of interference of placements in a solution to (CRPR) or (CRPRPS) given in Section 2.2.

Define w_{min}, w_{max} to be the smallest and largest weight of any instance of any activity in $\bigcup_{j \in [n]} \mathcal{A}_j$. Take

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

for $\alpha > 0$. The parameter α is preselected to maximize the quantity r. We then provide α as an input to Algorithm 3. The algorithm from [10] that we will present gives an r-approximation for a given instance of the resource allocation problem.

In addition for an instance $I \in \bigcup_{j \in [n]} \mathcal{A}_j$, let $\mathcal{A}(I)$ be the activity to which I belongs, and let $\mathcal{I}(I)$ be the set of instances not belonging to $\mathcal{A}(I)$, but interfering with I.

The authors provide Algorithm 3, which proceeds in two phases. In the first phase a stack L is created, which stores a subset of instances that we may include in our final schedule S. At initialization we begin with the infeasible dual solution $y = 0, z = 0$ and an empty stack. At each iteration we examine an instance $I \in \bigcup_{j \in [n]} \mathcal{A}_j$ with smallest end-time $e(I)$, among all instances in $\bigcup_{j \in [n]} \mathcal{A}_j$ violating constraint (4.6b). Let δ be the quantity by which (4.6b) is violated for the selected instance I. We update our dual variables by adding δ to y_j and $\alpha \cdot \delta$ to z_t, where j is the activity $\mathcal{A}(I)$ and t is the time interval in $\mathcal{T}$ ending at $e(I)$. Note that by our construction of $\mathcal{T}$, t exists. Also, we note that constraint (4.6b) is no longer violated. At the end of the iteration, we add I to the top of our stack L. Once (y, z) is feasible, we proceed to the algorithm's second phase. Here we construct a schedule S by removing elements from the top of L and adding them to S if feasibility is maintained and discarding them otherwise. Recall that at each iteration in the first phase of the algorithm we chose to examine an instance $I \in \bigcup_{j \in [n]} \mathcal{A}_j$ with smallest end-time $e(I)$, among all instances in $\bigcup_{j \in [n]} \mathcal{A}_j$ violating constraint (4.6b). We will see that this choice allows us to conclude the primal objective of the solution S that we constructed is at least a proportion r of the objective value of our dual solution (y, z), giving us the r approximation we alluded to earlier.

Theorem 4. *Algorithm 3 is an r-approximation algorithm to the resource allocation problem.*

Proof. Suppose that $I_1, \ldots, I_q$ are the instances from $\bigcup_{j \in [n]} \mathcal{A}_j$ that our stack L is composed of at the end of the first phase of our algorithm, presented in the order we added them to L. For $q' \in [q]$ and $j \in [n]$, let $y_j^{q'}$ be the amount by which we increased variable y_j during iteration q' of the first phase of our algorithm. Note that $y_j^{q'} > 0$ if and only if j is the activity $\mathcal{A}(I_{q'})$. Also for $t = [t_1, t_2] \in \mathcal{T}$, let $z_t^{q'}$, be the amount by

Algorithm 3 Resource-Allocation-Primal-Dual($\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}, p, w$)

Input: Set of activity instances $\mathcal{A}_j$, for each activity $j \in [n]$. Profit and weight functions p and w, respectively.

Output: A schedule S.

```
    Phase 1: Creating the Stack
 1: $L \leftarrow \emptyset$
 2: while $y, z$ is infeasible do
 3:     Let $I$ be an instance with smallest end-time $e(I)$, among all instances in $\bigcup_{j\in[n]} \mathcal{A}_j$ violating constraint (4.6b).
 4:     $\delta \leftarrow p(I) - y_j - w(I) \sum_{t=[t_1,t_2]\in T: s(I)\le i < i' \le e(I)} z_t$
 5:     $y_j \leftarrow y_j + \delta$, $z_t \leftarrow z_t + \alpha\delta$, where $t$ is the last time interval during which $I$ occurs.
 6:     Append $I$ to $L$
    Phase 2: Constructing the Solution
 7: $S \leftarrow \emptyset$
 8: while $L \neq \emptyset$ do
 9:     Take the instance $I$ off the top of $L$
10:     if $S \cup I$ is feasible then
11:         $S \leftarrow S \cup I$
    return $S$
```

which we increased variable z_t during iteration q' of the first phase of our algorithm. Note that $z_t^{q'} > 0$ if and only if $e(I_{q'}) = t_2$. Suppose we can express the primal objective of our solution S as the quantity

$$\sum_{q'\in[q]} \left(\sum_{j\in[n]} \beta_j^{q'} y_j^{q'} + \sum_{t\in\mathcal{T}} \eta_j^{q'} z_t^{q'} \right)$$

for $\beta, \eta \geq 0$, where the following holds. For each $q' \in [q]$, if j is the activity $\mathcal{A}(I_{q'})$, and $t = [t_1, t_2]$ is the interval in $\mathcal{T}$ such that $e(I_{q'}) = t_2$, then either $\beta_j^{q'} \geq 1$ or $\eta_t^{q'} \geq \max(w_{min}, 1 - w_{max})$, then we will have shown that Algorithm 3 is indeed an r-approximation algorithm to the resource allocation problem. To see this note that increase of the dual objective in iteration q' is $y_j^{q'} + z_t^{q'}$, where $y_j^{q'} = \delta$ and $z_t^{q'} = \alpha\delta$ for some $\delta > 0$. If $\beta_j^{q'} \geq 1$, then the term

$$\frac{\beta_j^{q'} y_j^{q'} + \eta_j^{q'} z_t^{q'}}{y_j^{q'} + z_t^{q'}} \geq \frac{\delta}{\delta + \alpha\delta} = \frac{1}{1+\alpha}.$$

If $\eta_t^{q'} \geq (w_{min}, 1 - w_{max})$, then the term

$$\frac{\beta_j^{q'} y_j^{q'} + \eta_j^{q'} z_t^{q'}}{y_j^{q'} + z_t^{q'}} \geq \frac{\max(w_{min}, 1 - w_{max})\alpha\delta}{\delta + \alpha\delta} = \frac{(w_{min}, 1 - w_{max})\alpha}{1+\alpha}.$$

Since we began with initial dual solution $y = 0, z = 0$, recalling that $r = \frac{\min(1, \alpha \max(w_{min}, 1-w_{max}))}{1+\alpha}$, and summing over all q' we would get that the profit of our schedule S is at least r times that of the objective function value of our dual.

Now we proceed by giving such β, η. For each $q^* \in [q]$, we know that

$$p(I_{q^*}) = y_j^{q^*} + \sum_{q''=1}^{q^*-1} \left(y_j^{q''} + w(I) \sum_{t=[t_1,t_2] \in \mathcal{T} : s(I) \leq t_1 < t_2 \leq e(I)} z_t^{q''} \right) \tag{4.7}$$

because at the time we considered I_{q^*} in the first phase of our algorithm we had that

$$y_j = \sum_{q''=1}^{q^*-1} y_j^{q''}$$

and

$$z_t = \sum_{q''=1}^{q^*-1} z_t^{q''}$$

for $t = [t_1, t_2] \in \mathcal{T} : s(I) \leq t_1 < t_2 \leq e(I)$, and that constraint (4.6b) for instance I_{q^*} was violated by δ, which is equal to our increase in y_j from this iteration, namely $y_j^{q^*}$.

Then the primal objective value of our schedule S is the quantity.

$$\sum_{q^* \in [q] : I_{q^*} \in S} p(I_{q^*}) = \sum_{q^* \in [q] : I_{q^*} \in S} \left(y_j^{q^*} + \sum_{q''=1}^{q^*-1} \left(y_j^{q''} + w(I) \sum_{t=[t_1,t_2] \in \mathcal{T} : s(I) \leq t_1 < t_2 \leq e(I)} z_t^{q''} \right) \right). \tag{4.8}$$

For each $q^* \in [q]$ and activity $j \in [n]$, let $\beta_j^{q^*}$ be the coefficient of $y_j^{q^*}$ in equation (4.8). Similarly, for any $q^* \in [q]$ and time interval $t \in \mathcal{T}$, let $\eta_t^{q^*}$ be the coefficient of $z_t^{q^*}$ in equation (4.8). Clearly, $\beta \geq 0, \eta \geq 0$ because the equation only includes terms with non-negative contributions from each variable of interest.

To prove that for each $q' \in [q]$, if j is the activity $\mathcal{A}(I_{q'})$, and $t = [t_1, t_2]$ is the interval in $\mathcal{T}$ such that $e(I_{q'}) = t_2$, then either $\beta_j^{q'} \geq 1$ or $\eta_t^{q'} \geq \max(w_{min}, 1 - w_{max})$, there are three cases to consider.

In the first case, we included $I_{q'}$ in S during the second phase of our algorithm. Then we see that terms from equation (4.7) for $p(I_{q'})$ are included in our expression from (4.8) of our primal objective. Since $y_j^{q'}$ has a coefficient of 1 in equation (4.7) for $p(I_{q'})$, we conclude that $\beta_j^{q'} \geq 1$ as required.

In the second case, we do not include $I_{q'}$ in S because at the time when we took $I_{q'}$ off of our stack L, there was $\bar{q} \in [q]$ such that element $I_{\bar{q}} \in S$ and $I_{\bar{q}} \in \mathcal{A}_j$. Since we constructed S by greedily taking elements off the top of our stack L we know that $\bar{q} > q'$. Thus $y_j^{q'}$ has a coefficient of 1 in equation (4.7), and we conclude that $\beta_j^{q'} \geq 1$ as required.

Finally, suppose that none of the first two cases hold. Note that we do not include $I_{q'}$ in S, but there is no instance of activity j in S at the point in the second phase when we considered $I_{q'}$. As a result, it must be that there exists some $\mathcal{X} \subseteq S$ of instances all interfering with $I_{q'}$ during some $t \in \mathcal{T}$, such that $w(\mathcal{X}) + w(I) > 1$. Again since we constructed S by greedily taking elements off the top of our stack L we know that if $I_{\bar{q}} \in \mathcal{X}$ then $\bar{q} > q'$. Take such an element $I_{\bar{q}} \in \mathcal{X}$. Recall that $\bar{t} = [t_1, t_2]$ is the unique interval in $\mathcal{T}$ such that $e(I_{q'}) = t_2$. Since we considered $I_{\bar{q}}$ after $I_{q'}$ in the first phase of the algorithm we know that $e(I_{q'}) \leq e(I_{\bar{q}})$. Thus $I_{q'}$ and $I_{\bar{q}}$ interfere with one another during t. Hence for all $\bar{q} \in [q]$ such that $I_{\bar{q}} \in \mathcal{X}$ since $\bar{q} \geq q'$ and we have a coefficient of $w(I_{q'})$ in front of $z_t^{q'}$ in (4.7) for $p(I_{\bar{q}})$. Since for each such $\bar{q}$ the terms from (4.7) for $p(I_{\bar{q}})$ appear in the expression (4.8) we conclude that the coefficient $\eta_t^{q'}$ is at least $w(\mathcal{X})$. Note that $\mathcal{X}$ is non-empty for otherwise $w(\mathcal{X}) + w(I_{q'}) = w(I_{q'}) \leq 1$ a contradiction. Thus $w(\mathcal{X})$ is at least w_{min}, the smallest weight of any instance in $\bigcup_{j\in[n]} \mathcal{A}_j$. If $w(\mathcal{X}) \leq 1 - w_{max}$, then clearly $w(\mathcal{X}) + w(I_{q'}) \leq 1 - w_{max} + w_{max} = 1$ a contradiction as w_{max}, is the largest weight of any instance in $\bigcup_{j\in[n]} \mathcal{A}_j$. We conclude that

$$\eta_t^{q'} \geq \max(w_{min}, 1 - w_{max})$$

concluding our proof.

□

4.2.2 Reduction from (CRPRPS) to Resource Allocation Problem

Suppose we are given such an instance $\mathcal{I}$ of (CRPRPS) fitting the description in section 2.2.2. Our reduction will be very similar to that in Section 4.1 and the proofs of correctness are very similar. As a result, we provide only the reduction itself.

For each segment $j \in J$ and pair of endpoints on the ideal curve, $(\ell, r) \in \mathcal{P}_j$ we create a set of instances $\mathcal{A}_{j,\ell,r}$ to be added to $\mathcal{A}_j$. The instances in $\mathcal{A}_{j,\ell,r}$ are as follows.

Let the units of time $\mathcal{T}$ in equation 4.4 be the elements of $g_{/\bar{G}}$, ordered increasingly with respect to their placement along the interval $[0, T]$. Since each pair of time intervals in $\mathcal{T}$ are non-intersecting except possibly at an endpoint, this ordering is non-ambiguous. For each $\ell', r' \in \binom{\bar{G}}{2}$ such that $\ell \preceq_g \ell' \prec_g r' \preceq_g r$ add the instance $I = (j, \ell, r, \ell', r')$ to $\mathcal{A}_j$, with

$$s(I) = \ell', e(I) = r', p(I) = \sum_{i:\ell' \leq i \leq r'-1} \mu(i, 0) - c(j, \ell, r), w(I) = 1.$$

Instance I will correspond to placing j over (ℓ, r) and taking credit only for the intervals in $\{\ell', \dots, r'-1\}$.

Note that since all intervals $I \in \bigcup_{j\in[n]}$ have the property that $w(I) = 1$ our approximation guarantee r

for Algorithm 3 is $\frac{\min(1,\alpha)}{1+\alpha}$ for our choice of $\alpha > 0$. Since this quantity is maximized for $\alpha = 1$, Algorithm 3 gives us a $\frac{1}{2}$-approximation.

4.2.3 Unanswered Questions

The approach in this section does not provide us any intuition about how to form the pieces of the deformed bandeau. A similar reduction from (CRPR) to the resource allocation problem does not apply because it is not clear how to construct the activities. It would be of interest to see if the techniques can be modified or built upon to choose good pieces.

We mentioned in chapter 1 that we are studying the fronto-orbital advancement with rearrangement procedure to gain an intuition for how to perform cranial vault remodeling procedures best. We now discuss problems in applying a similar method to an analogous mathematical model for the cranial vault remodeling procedure. Our models (CRPR) and (CRPRPS) used curves to represent our bandeau and its template. If two placements $(j, \ell_j, r_j) \in \mathcal{P}_j, (j', \ell_{j'}, r_{j'}) \in \mathcal{P}_{j'}$ interfere with one another it is clear that the two placements interfere with one another at $r_j - 1$ assuming that $\ell_j \prec_g \ell_{j'}$. Similarly in the corresponding resource allocation instance if we have interfering instances $e_j = (j, \ell_j, r_j, \ell'_j, r'_j), e_{j'} = (j, \ell_j, r_j, \ell'_j, r'_j)$ where $\ell_j \preceq_g \ell'_j$ is is clear that e_j and $e_{j'}$ interfere with one another at the time interval corresponding to interval $\ell'_j \in g_{/\bar{G}}$. This observation allows the authors in [10] to capture at least half of the objective function value of the constructed feasible dual solution. It is not clear whether or not there is a suitable modification that will give us an algorithm for use in deciding where to place pieces during a cranial vault remodeling procedure.

4.3 Local Search

In this section, we propose a local search heuristic for any instance of (CRPRPS). Let $\mathcal{I}$ be an instance of (CRPRPS).

For two solutions S and S' of $\mathcal{I}$, we say that S' *changes the placement* of a piece j from that of S if one of the following hold. Only one of S and S' places piece j, or S assigns piece j to cover a different curve segment of g than S'.

We first recall some notation from section 2.2. If a solution S places piece j over curve segment g_{ℓ_j, r_j} for some $j \in J$ and $(\ell_j, r_j) \in \binom{\bar{G}}{2}$ we say $S_j = (\ell_j, r_j)$, and $(j, \ell, r) \in S$. Also, for each $j \in J$, we let $\mathcal{P}_j$ be the set of all placements of j of finite cost.

As mentioned in section 2.3.3, to give a local search heuristic, it is sufficient to specify the local search neighbourhood, $N(S)$ for any solution S. We will specify our local search neighbourhood by limiting the number and types of changes that may be made to a solution S. In section 4.4 we will find it useful to restrict our neighbourhood so that we only change the locations of pieces in some set $J^c \subseteq J$. In addition we will also find it useful to restrict our search so that we do not consider all placements in $\mathcal{P}_j$ for all $j \in J$, but rather placements in T_j for some $T_j \subseteq \mathcal{P}_j$. With this in mind, we define a family of neighbourhoods. For each fixed $k' \leq k$, each choice of

$$(T_1, \ldots, T_k) \subseteq \mathcal{P}_1 \times \cdots \times \mathcal{P}_k,$$

each set of indices $J^c \subseteq J$ and feasible solution S, define $N_{k'}(S, J^c, \{T_j : j \in J^c\})$ to be the set containing all solutions S' that can be obtained from S by changing the placement of at most k' pieces in J^c, so that for piece $j \in J$, we have $S_j \in \{\emptyset\} \cup T_j$. Formally $S' \in N_{k'}(S, T_1, \ldots, T_k)$, if

$$|\{j \in J : S'_j \neq S_j\}| \leq k'$$

and $S_j \in \{\emptyset\} \cup T_j$ for all pieces $j \in J$, with the added condition that $S_j = S'_j$ for all $j \notin J^c$.

As mentioned in section 2.3.3, it is often difficult to prove good performance guarantees for local search algorithms. Suppose we have a solution S, that uses placements $(j_1, \ell_1, r_1), \ldots, (j_t, \ell_t, r_t)$, all of which interfere with some placement $(j, \ell, r) \notin S$. Suppose that the solution S' obtained from S by removing all placements $(j_1, \ell_1, r_1), \ldots, (j_t, \ell_t, r_t)$, and adding (j, ℓ, r), has a higher objective value. If the set $\{j, j_1, \ldots, j_t\}$ has cardinality greater than k', then $S' \notin N_{k'}(S)$, as we can only obtain S' from S by changing the location of more than k' pieces. We can easily construct instances where the difference between the objective values of such a pair S and S' is large. To avoid this we may construct a neighbourhood $N'(S, J^c, \{T_j : j \in J^c\})$, which for a solution S contains all solutions S' obtained by fixing the placements of some piece j at some $(\ell, r) \in T_j$ and deleting all placements of pieces in J^c, that interfere with (j, ℓ, r). Note that $|N'(S)| = O(k|G|^2)$, and $|N_{k'}(S)| = O(\binom{k}{k'}|G|^{2k'})$. As a result, for any choice of k', checking $N'(S)$ for local moves will be dominated by checking $N_{k'}(S)$ for local moves.

In practice to obtain a polynomial algorithm, we will fix a constant choice of k' and take

$$N(S, J^c, \{T_j : j \in J^c\}) = N_{k'}(S, J^c, \{T_j : j \in J^c\}) \cup N'(S, J^c, \{T_j : j \in J^c\})$$

for each feasible solution S. This alone may not be sufficient to guarantee polynomial runtime, as our improvement in the objective function may be too small. A possible strategy for avoiding this is to require that for some appropriate choice of $\epsilon > 0$ we make the local move from S to $S' \in N(S)$ only if

$$C_{max}(S') \geq C_{max}(S) + \epsilon.$$

For example if we take

$$\epsilon = O\left(\frac{\sum_{i \in g_{/\bar{G}}} \mu(i,0)}{poly(|G|)}\right),$$

then as

$$\sum_{i \in g_{/\bar{G}}} \mu(i,0)$$

is an upper bound on the value of any feasible solution we can have at most $O(poly(|G|))$ rounds of local search. Without any intuition on how we can filter each set $\mathcal{P}_j$ to obtain a suitable choice of T_j it will be practical for us to implement a version of the proposed local search heuristic for $k' = 1$.

Algorithm 4 *LocalSearch*$(\mathcal{I}, S, J^c, \{T_j : j \in J^c\})$

Input: An instance $\mathcal{I}$, a feasible solution S of $\mathcal{I}$, a subset J^c of pieces indices, and a subset T_j of $\mathcal{P}_j$ for all $j \in J^c$ **Output:** A solution S^*.

```
1: $S^* \leftarrow S$
2: while $C_{max}(S') \geq C_{max}(S^*) + \epsilon$ for some $S' \in N(S, J^c, \{T_j : j \in J^c\})$ do
3:     $S^* \leftarrow S'$
   return $S^*$
```

4.4 Alternative Primal-Dual Approaches

In this section we offer a pair of heuristics to (CRPRPS) that call local search heuristic 4 as a subroutine, for a fixed choice of neighbourhood size k'. Let $\mathcal{I}$ be an instance of (CRPRPS).

4.4.1 Motivation

Our heuristics are motivated by two similar observations, for which the following definitions will be helpful.

Definition 23 (Individual Profit). *For a piece $j \in J$ and a placement $(\ell, r) \in \mathcal{P}_j$ the individual profit of (j, ℓ, r) is the quantity*

$$p(j,\ell,r) = \sum_{i:\ell \preceq_g i \prec_g r} \mu(i,0) - c(j,\ell,r).$$

Definition 24 (Average Cost)**.** *For a piece $j \in J$ and a placement $(\ell, r) \in \mathcal{P}_j$ the average cost of (j, ℓ, r) is the quantity*

$$\frac{c(j,\ell,r)}{|\{\ell,\dots,r-1\}|}.$$

For each interval $i \in g_{/\bar{G}}$ let $c_{min}(i)$ be the lowest average cost of any placement covering interval i. We observe that if a solution S of $\mathcal{I}$ satisfies the following two conditions, we may conclude that it is indeed relatively close to optimal. The first is that S leaves little of the possible coverage reward unclaimed. The second is that the sum of the costs of all of the used placements is close to $\sum_{i:cov_S(i)\geq 1} c_{min}(i)$. Since

$$\sum_{i:cov_S(i)\geq 1} (\mu(i,0) - c_{min}(i))$$

is the largest objective possible for any solution covering at least the intervals that S covers, our claim follows. Note that, in general, the optimal solution may not even satisfy this property. Our observation is only made to motivate our algorithm.

We recall from Section 2.3.3 that one can attempt to improve the outcome of a local search heuristic by running it multiple times with different initial solutions. Our first approach uses this and the above observation by at first calling heuristic 4 with neighbourhoods containing only placements of low average cost. The approach then slowly relaxes our threshold allowing placements of higher average cost for intervals not currently covered by our solution into the search neighbourhoods, for later calls of heuristic 4.

Now we make a similar observation, which will motivate an alternative approach. For a piece $j \in J$ let $p_{max}(j)$ be the cost of a placement in $\mathcal{P}_j$, that maximizes individual profit. If a solution S has an objective value close to

$$\sum_{j\in J} p_{max}(j),$$

then we may conclude that S is close to optimal as this gives an upper bound on our objective value. This observation motivates our second approach, where we will again call heuristic 4 with multiple starting solutions. This time, however, we will begin the algorithm by restricting our search neighbourhood to placements of high profit. We then slowly relax our search neighbourhoods to include placements of lower profit to encourage more coverage in our solution. We note that in general, the optimal value of may be much smaller than

$$\sum_{j\in J} p_{max}(j),$$

and so we again only use our observation as a guide.

In our approaches, we will use primal-dual techniques to motivate how we relax the restrictions on our neighbourhoods. To this end, in section 4.4.2 we introduce an integer programming formulation for the restricted class of instances of (CRPRPS) we study in this chapter.

It should be noted that we do not claim that our heuristics accomplish the goal of balancing coverage and fit in the best way possible. The discussion above is simply a guide for what we are trying to accomplish in this section. Whether we can improve upon this approach is a topic for further study.

4.4.2 Integer Programming Formulation

We first write the integer program in (4.9) that models our restricted class of (CRPRPS) instances. For $j \in J$ and $(\ell, r) \in \mathcal{P}_j$ we have a decision variable $x_{j,\ell,r}$ which takes on the value 1 if we place piece j over $g|_{\ell,r}$ and 0 otherwise. For interval $i \in g_{/\bar{G}}$ we have a decision variable y_i which equals 1 if we have covered interval i and 0 otherwise. Since we have $\mu(i,z) = 0$ for all $z \geq 2$, we are only concerned with whether or not a solution S has $cov_S(i) \geq 1$, and so (4.9a) is indeed the correct objective function. Since we can only collect a reward for covering interval $i \in g_{/\bar{G}}$, for a solution S if $cov_S(i) \geq 1$ we impose (4.9b). In addition each reward can be collected at most once, so we impose (4.9d). Lastly, as each deformed piece can be used at most once, we require (4.9c). The dual of the linear programming relaxation of (4.9) is given in (4.10). Note that in our integer program we label the primal constraints with their corresponding dual variables.

$$
\begin{aligned}
\max \quad & \sum_{i \in g_{/\bar{G}}} \mu(i,0) y_i - \sum_{j \in J, (\ell,r) \in \mathcal{P}_j} c(j,\ell,r) x_{j,\ell,r} && \text{(4.9a)} \\
\text{s.t.} \quad & y_i \leq \sum_{j \in J, (\ell,r) \in \mathcal{P}_j : \ell \preceq_g i \prec_g r} x_{j,\ell,r} \quad \forall i \in g_{/\bar{G}} \qquad [\pi_i] && \text{(4.9b)} \\
& \sum_{(\ell,r) \in \mathcal{P}_j} x_{j,\ell,r} \leq 1 \quad \forall j \in J \qquad [\gamma_j] && \text{(4.9c)} \\
& y_i \leq 1 \quad \forall i \in g_{/\bar{G}} \qquad [\beta_i] && \text{(4.9d)} \\
& 0 \leq y_i \leq 1, \text{ integer} \quad \forall i \in g_{/\bar{G}} && \text{(4.9e)} \\
& x_{j,\ell,r} \in \{0,1\} \quad \forall j \in J, (\ell,r) \in \mathcal{P}_j. && \text{(4.9f)}
\end{aligned}
$$

$$\min \quad \sum_{j \in J} \gamma_j + \sum_{i \in g_{/\bar{G}}} \beta_i \tag{4.10a}$$

$$\text{s.t.} \quad \pi_i + \beta_i \geq \mu(i,0) \quad \forall i \in g_{/\bar{G}} \tag{4.10b}$$

$$\sum_{i:\ell \leq i \leq r-1} \pi_i - \gamma_j \leq c(j,\ell,r) \quad \forall j \in J, (\ell,r) \in \mathcal{P}_j \tag{4.10c}$$

$$\pi, \gamma, \beta \geq 0. \tag{4.10d}$$

When running our heuristics we will maintain an integral primal solution S and a feasible dual solution π, β, γ in our algorithms, such that whenever $(j,\ell,r) \in S$ for some $j \in J, (\ell,r) \in \mathcal{P}_j$, we have that

$$\sum_{i:\ell \preceq_g i \prec_g r} \pi_i - \gamma_j = c(j,\ell,r).$$

In other words, constraint (4.10c) for placement (j,ℓ,r) is tight.

We observe that if we have a dual solution π, β, γ where for some interval $i \in g_{/\bar{G}}$ constraint (4.10b) is not tight we can improve our dual solution by setting $\beta_i = \mu(i,0) - \pi_i$. As a result we will only construct dual solutions for which constraint (4.10b) is always tight for each $i \in g_{/\bar{G}}$. Under this restriction the condition

$$\sum_{i:\ell \preceq_g i \prec_g r} \pi_i - \gamma_j = c(j,\ell,r)$$

is equivalent to

$$\sum_{i:\ell \preceq_g i \prec_g r} \beta_i + \gamma_j = p(j,\ell,r).$$

This can be seen by subtracting both sides of the first equation from

$$\sum_{i:\ell \preceq_g i \prec_g r} \mu(i,0)$$

and simplifying.

If S and π, β, γ are a primal-dual pair satisfying the above conditions, the difference between the primal objective of S and the objective of the dual solution can be computed by equation (4.11). To see this, first let J' be the set of pieces placed by S, and for each $j \in J'$, let $S_j = (\ell_j, r_j)$. Then since $\mu(i,z) = 0$ for all

$z \geq 1$ it must hold that

$$C_{max}(S) = \sum_{j \in J'} p(j, \ell_j, r_j) - \sum_{i \in g_{/\bar{G}} : cov_S(i) > 1} (cov_S(i) - 1)\mu(i, 0),$$

as we can only collect the coverage reward $\mu(i, 0)$ one time per interval i.

By our previous remarks, we conclude that

$$C_{max}(S) = \sum_{j \in J'} (\gamma_j + \sum_{i : \ell \preceq_g i \prec_g r} \beta_i) \sum_{i : cov_S(i) > 1} (cov_S(i) - 1)\mu(i, 0).$$

Since $\beta_i = \mu(i, 0) - \pi_i$, for all $i \in g_{/\bar{G}}$, we can simplify to get

$$C_{max}(S) = \sum_{j \in J'} \gamma_j + \sum_{i : cov_S(i) \geq 1} \beta_i - \sum_{i \; cov_S(i) > 1} (cov_S(i) - 1)\pi_i.$$

Since the dual objective is

$$\sum_{j \in J} \gamma_j + \sum_{i \in g_{/\bar{G}}} \beta_i,$$

we can conclude that the difference between the primal objective of S and the objective of our dual solution is indeed given in (4.11).

$$\sum_{i \in g_{/\bar{G}} : cov_S(i) = 0} \beta_i + \sum_{i \in g_{/\bar{G}} : cov_S(i) > 1} (cov_S(i) - 1)\pi_i + \sum_{j \in J, S_j = \emptyset} \gamma_j \tag{4.11}$$

Both of our algorithms will start with and maintain an integral primal solution S and a dual solution π, β, γ, at each stage modifying the solutions in such a way as to close the gap in equation (4.11).

At any given point in the following discussion, for a piece $j \in J$ we will take T_j to be the set of placements of piece j for which constraint (4.10c) is tight. Thus, in this section, when calling the local search heuristic 4, once we specify a set J^c of piece indices whose placements we can change, we will take our set of possible new locations for each piece $j \in J^c$ to be the set of tight constraints T_j. In our discussion we will only specify the initial solution S and set of piece indices J^c, whenever we call heuristic 4.

4.4.3 Decreasing Primal-Dual Approach

We provide a brief outline of how our algorithm will proceed. We will initialize by taking $\beta_i = \mu(i, 0)$ for each $i \in g_{/\bar{G}}$, and $\gamma = 0$. Note that the only placements in T_j for some j will be those with perfect fit at initialization, or in other words, placements of average cost 0. We then take our primal solution S to be the solution outputted by running heuristic 4 with $J^c = J$ and the empty solution as input.

At any given iteration, we will use linear programming to determine a system of rates $\Delta^\gamma, \Delta^\beta$ which we will use to update our dual variables. We have previously noted that we will always assume $\beta_i = \mu(i, 0) - \pi_i$, for all $i \in g_{/\bar{G}}$. For this reason, we have only specified a rate for how Δ^β is changing.

We will determine the rates by solving a sequence of linear programs, which aim to update our dual solution in such a way as to decrease the quantity in equation (4.11) as much as possible with respect to our current solution S. When $\Delta^\gamma = 0, \Delta^\beta = 0$ is an optimal solution to our first system of linear equations, the algorithm does not expect to further decrease the gap between the primal objective of S and the objective of our dual solution, so we will terminate. Otherwise, at each iteration, there will exist some positive quantity $v > 0$ such that we can update our dual variables according to equation (4.12). If in this process there is a non-empty set of placements whose dual constraint (4.10c) becomes tight, we will consider updating our solution S to include them. Next, we provide the specific choices we have made for how each of these steps are completed.

$$\beta \leftarrow \beta - \Delta^\beta \cdot v, \gamma \leftarrow \gamma + \Delta^\gamma \cdot v \tag{4.12}$$

Since we wish to decrease the quantity computed in equation (4.11), we choose to optimize the objective function (4.13a). Since S is the best solution we have found thus far, we currently have no incentive to move away from S, and so we ensure all placements in S remain tight (4.13b). We also wish to maintain dual feasibility for some positive change in the dual variables ((4.13c)). Putting this together and normalizing

Δ^β we solve for an optimal solution of system 4.13.

$$\max \sum_{i \in g_{/\bar{G}}} (1 - cov_S(i))\Delta^\beta - \sum_{j \in J, S_j = \emptyset} \Delta^\gamma_j \tag{4.13a}$$

$$\text{s.t.} \quad \Delta^\gamma_j = \sum_{i:\ell \le i \le r} \Delta^\beta_i \quad \forall j \in J, \text{ if } (\ell, r) \in S_j \tag{4.13b}$$

$$\Delta^\gamma_j \ge \sum_{i:\ell' \le i \le r'} \Delta^\beta_i \quad \text{for } (\ell,' r') \in T_j \setminus S_j \tag{4.13c}$$

$$\Delta^\beta_i = 0 \quad \forall i \text{ s.t. } \pi_i = \mu_i \tag{4.13d}$$

$$0 \le \Delta^\beta \le 1 \tag{4.13e}$$

$$\Delta^\gamma \ge 0 \tag{4.13f}$$

Even though our choice of rates objective (4.13a) was motivated by equation (4.11), we offer an intuitive explanation of this choice. At each iteration, we aim to increase the quality of our solution S. Recall this is dictated by the fit and coverage functions c and μ. We begin with a search neighbourhood containing only 0-cost placements. While we suspect our search neighbourhood does not contain a set of placements that optimizes the tradeoff between c and μ, we would like to consider including placements of increasing cost to our neighbourhood in the hopes that these placements will allow us to cover a broader range on unit intervals in $g_{/\bar{G}}$. Since we only consider placements whose dual constraints are tight with respect to (4.10c), it is advantageous for us to increase π_i for intervals not covered by S at as fast a rate as possible so that more placements covering these intervals become tight. On the other hand, if a piece $j \in J$ is not used by S, we would want more options for this piece, and by the same reasoning, we would wish to keep γ_j constant. If an interval $i \in g_{/\bar{G}}$ is covered multiple times by S, it does not need any more options, so we wish to keep π_i constant. If the optimal solution to system (4.13) has objective value 0, we terminate our algorithm. Since our objective function was selected to decrease the gap between our primal and dual solution, this condition means that our algorithm does not expect to be able to decrease this gap further.

It should be noted that there may be many optimal solutions to system (4.13). We will provide a heuristic for selecting a particular optimal solution to system (4.13) by solving two additional linear programs. To do this, we make some observations about system (4.13). Suppose that S is a local optimum for heuristic (4) at the end of the last iteration.

Suppose that a piece (j, ℓ, r) is not included in our set of tight constraints. If $cov_S(i) \geq 1$ for all $i \in \{\ell, \dots, r-1\}$, then under the assumption that piece (j, ℓ, r) has higher average cost than the tight placements already placed by S, we have no motivation to include it in our solution. If we pick an optimal solution to (4.13) with a small value for the quantity

$$\sum_{i:\ell \preceq_g i \prec_g r-1} \Delta_i^\beta,$$

we will not be encouraging such placements (j, ℓ, r) to become tight. To capture this we chose to pick an optimal solution that minimizes the quantity

$$\sum_{i \in g_{/\bar{G}}} \Delta_i^\beta,$$

so that whenever possible we are encouraged to set $\Delta_i^\beta = 0$ for all i such that $cov_S(i) = 0$.

By constraint (4.13b) and the fact that any optimal solution to (4.13) will prefer to increase the rates of intervals i with $cov_S(i) = 1$ over intervals with $cov_S(i) > 1$, we choose to model this by instead picking a solution that minimizes the quantity

$$\sum_{j \in J : S_j \neq \emptyset} \Delta_j^\gamma.$$

Using linear programming to solve for these properties, we chose an optimal solution to system (4.14), where P is the feasible region to system (4.13) and OPT_1 is the optimal objective value of system (4.13). Note that for simplicity we have chosen to minimize

$$\sum_{j \in J} \Delta_j^\gamma$$

rather than

$$\sum_{j \in J : S_j \neq \emptyset} \Delta_j^\gamma.$$

Let OPT_2 be the optimal objective value of system (4.14). In addition we choose to solve a third linear program (4.15), which amongst all optimal solutions to (4.14) minimizes the total slack created in the tight constraints. For many instances that we are interested in, this approach experimentally appears to reduce the number of iterations until termination. Whether or not we have bad the best choices it a topic for future investigation.

$$\min \sum_{j \in J} \Delta^{\gamma}_{j} \tag{4.14a}$$

$$\text{s.t.} \sum_{i \in g_{/\bar{G}}} (1 - cov_S(i))\Delta^{\beta} - \sum_{j \in J,\text{ unused}} \Delta^{\gamma}_{j} = OBJ_1 \tag{4.14b}$$

$$(\Delta^{\beta}, \Delta^{\gamma}) \in P \tag{4.14c}$$

$$\min \sum_{j \in J} \sum_{(\ell,r) \in T_j} z_{j,\ell,r} \tag{4.15a}$$

$$\text{s.t.} \sum_{i \in g_{/\bar{G}}} (1 - cov_S(i))\Delta^{\beta} - \sum_{j \in J,\text{ unused}} \Delta^{\gamma}_{j} = OBJ_1 \tag{4.15b}$$

$$\sum_{j \in J} \Delta^{\gamma}_{j} = OBJ_2 \tag{4.15c}$$

$$z_{j,\ell,r} = \Delta^{\gamma}_{j} - \sum_{i:\ell \leq i \leq r} \Delta^{\beta}_{i} \quad \forall j \in J, (\ell, r) \in \mathcal{P}_j \tag{4.15d}$$

$$(\Delta^{\beta}, \Delta^{\gamma}) \in P \tag{4.15e}$$

Once we have selected our rates $\Delta^{\gamma}, \Delta^{\beta}$, we will compute v in equation (4.12) as follows. Since system 4.13 does not consider placements whose dual constraint 4.10c is not tight, for each such placement (j, ℓ, r) there is some $v^{\gamma}_{j,\ell,r} \in \mathbb{R} \cup \{\infty\}$ such that

$$v^{\gamma}_{j,\ell,r} = \frac{c(j,\ell,r) + \gamma_j - \sum_{i:\ell \leq i \leq r-1} \pi_i}{\sum_{i:\ell \leq i \leq r-1} \Delta^{\beta}_{i} - \Delta^{\gamma}_{j}}.$$

Note that only placements for which $v^{\gamma}_{j,\ell,r} \in \mathbb{R}_{>0}$ can become tight if we change our dual solution at rates $\Delta^{\gamma}, \Delta^{\beta}$. Hence we take

$$v^{\gamma} = \min_{(\ell,r) \in \binom{\bar{G}}{2}, v_{j,\ell,r} > 0} v_{j,\ell,r}.$$

We are also limited by the fact that $\beta \geq 0$ and so we take

$$v^{\beta} = \min_{i \in g_{/\bar{G}}} \frac{\beta_i}{\Delta^{\beta}_{i}}.$$

Then v is given in equation (4.16).

$$v = \min(v^{\gamma}, v^{\beta}). \tag{4.16}$$

After selecting rates $\Delta^{\gamma}, \Delta^{\beta}$, computing our increase limit v and updating our dual variable accordingly, we will be in at least one of the following situations. The first situation is that there is some set of $I \subseteq g_{/\bar{G}}$ of intervals such that for $i \in I$ $\beta_i > 0$ before the current iteration, but currently $\beta_i = 0$. The second situation is that there is some set of placements N that were not tight in the previous round but are currently tight. In the latter case, for each placement $(j, \ell, r) \in N$ we compute the solution S' taken by fixing the placement of piece j to be (ℓ, r) and applying the local search algorithm in 4.3, with $J^c = J \setminus \{j\}$. If all such S' satisfy $C_{max}(S') \leq C_{max}(S) + \epsilon$ we do not update S, otherwise we take S to be best new solution computed. The choice of how to use algorithm in 4 as a subroutine is motivated by the following two observations. First, if we choose to update our solution S in a given iteration, we will only do so if we include at least one placement from N because otherwise, we would have done so in a previous iteration. Thus if $|N|$ is small, it makes sense to run the local search algorithm $|N|$ times where each time we fix one element of N to be in our solution. When testing this algorithm, we have found that $|N|$ is usually small in our instances of interest, including one or two placements. Whether or not this is the best choice is a topic to be explored in future research.

The complete algorithm is given in Algorithm 5.

We make a final note that connects us back to our motivation for studying this approach given at the beginning of section 4.4.1. When we select our rates $\Delta^{\gamma}, \Delta^{\beta}$, we are only motivated to assign positive values to these rates if doing so allows us to change β_i for some i, where i is uncovered by S. For each piece j, we are encouraging placements of higher average cost that cover area of the ideal curve not currently covered by S.

4.4.4 Increasing Primal-Dual Approach

This approach will be similar to that of Section 4.4.3, so we will be more brief.

In this approach we will initialize the algorithm by taking $\beta_i = 0$ for each $i \in g_{/\bar{G}}$, and $\gamma_j = p_{max}(j)$. Note that at initialization, the only constraints in T_j for some j will be those that maximize individual profit. We then take our primal solution S to be the solution outputted by running the local search algorithm in section 4.3 with $J^c = J$ and the empty solution as input. After initialization, the quality of S will depend on how much overlap is present. If there is lots of overlap, we wish to relax the placements allowed in T_j for

Algorithm 5 Primal-Dual Decrease ($\mathcal{I}$)

Input: An instance $\mathcal{I}$ of (CRPRPS)
Output: A solution S to $\mathcal{I}$.

```
 1: $S \leftarrow \emptyset$
 2: $\beta_i = \mu(i, 0), \pi_i = 0$ for all $i \in g_{/\bar{G}}$, $\gamma = 0$, $T_j \leftarrow \{(\ell, r) \in \mathcal{P}_j : c(j, \ell, r) = 0\}$ for $j \in J$
 3: $S \leftarrow LocalSearch(\mathcal{I}, \emptyset, J, \{T_j : j \in J\})$
 4: Solve (4.15) to get $\Delta^\gamma, \Delta^\beta$
 5: while Equation (4.13a) is positive do
 6:     $v^\gamma = \min_{(\ell,r)\in\binom{\bar{G}}{2}, v_{j,\ell,r}>0} v_{j,\ell,r}$
 7:     $v^\beta = \min_{i\in g_{/\bar{G}}} \frac{\beta_i}{\Delta_i^\beta}$
 8:     $v = \min(v^\gamma, v^\beta)$
 9:     $T'_j \leftarrow \{(\ell, r) \in \mathcal{P}_j : \sum_{i:\ell\leq i\leq r-1} \pi_i - \gamma_j = c(j, \ell, r)\}$ for each $j \in J$
10:     $\beta \leftarrow \beta - v \cdot \Delta^\beta, \pi \leftarrow \pi + v \cdot \Delta^\beta, \gamma \leftarrow \gamma + v \cdot \Delta^\gamma$
11:     $T_j \leftarrow \{(\ell, r) \in \mathcal{P}_j : \sum_{i:\ell\leq i\leq r-1} \pi_i - \gamma_j = c(j, \ell, r)\}$ for each $j \in J$
12:     $N_j \leftarrow T_j \setminus T'_j$ for each $j \in J$
13:     $S^* \leftarrow S$
14:     for $j \in J$ do
15:         for $(\ell, r) \in N_j \setminus T_j$ do
16:             $S' \leftarrow S$
17:             $S'_j \leftarrow (\ell, r)$
18:             $S' \leftarrow LocalSearch(\mathcal{I}, S', J \setminus \{j\}, \{T_j : j \in J \setminus \{j\}\})$
19:             if $C_{max}(S') \geq C_{max}(S^*) + \epsilon$ then
20:                 $S^* \leftarrow S'$
21:     if $C_{max}(S^*) \geq C_{max}(S) + \epsilon$ then
22:         $S \leftarrow S^*$
23:     Solve (4.15) to get $\Delta^\gamma, \Delta^\beta$
    return $S$
```

each $j \in J$ to include options of lower individual profit but cover areas not currently covered by S.

We will again update our dual solution by decreasing the difference between the current dual objective and the primal value of S. This equation is again given by equation (4.11). In some sense, we have chosen the opposite choice of initialization conditions to that in the decreasing approach from Section 4.4.3. If we wish to minimize the gap between primal and dual objective, we should increase β_i for intervals i that are covered multiple times. Intuitively we cannot collect a reward $\mu(i, 0)$ more than once, and increasing β_i symbolizes this. To see this, if we maintain feasibility as we alter β, we must simultaneously decrease γ_j for each piece j covering interval i in S. This process will encourage new placements of each such piece j to become tight covering intervals not currently covered by S. Using this logic we formulate system (4.18), and subsequently we solve system (4.19) and (4.20), where P is the feasible region to system 4.18 and OPT_1 is the optimal objective value of system 4.18 and OPT_2 is the optimal objective value of system 4.19. The logic behind our choices of (4.19) and (4.20) is similar to that of (4.14) and (4.15) in Section 4.4.3.

Again we can only change our dual variables at the rates Δ^{γ} and Δ^{β} for some $v > 0$ while maintaining a feasible dual solution. The formula for v is the same as in the other approach and is given in equation (4.16) in Section 4.4.3 for the other approach. The equation for the dual variables at the end of a given iteration is given by (4.17). We note that the dual-variables that were increasing in our other approach are decreasing in this approach, and the dual-variables that were decreasing in our other approach are increasing in this approach.

$$\beta \leftarrow \beta + \Delta^{\beta} \cdot v, \pi \leftarrow \pi - \Delta^{\beta} \cdot v, \gamma \leftarrow \gamma - \Delta^{\gamma} \cdot v \tag{4.17}$$

$$\max \sum_{i \in g_{/\bar{G}}} (cov_S(i) - 1)\Delta^{\beta} + \sum_{j \in J, S_j = \emptyset} \Delta_j^{\gamma} \tag{4.18a}$$

$$\text{s.t.} \quad \Delta_j^{\gamma} = \sum_{i:\ell \leq i \leq r} \Delta_i^{\beta} \quad \forall j \in J, \text{ if } (\ell, r) \in S_j \tag{4.18b}$$

$$\Delta_j^{\gamma} \leq \sum_{i:\ell' \leq i \leq r'} \Delta_i^{\beta} \quad \text{for } (\ell,' r') \in T_j \setminus S_j \tag{4.18c}$$

$$\Delta_i^{\beta} = 0 \quad \forall i \text{ s.t. } \beta_i = \mu_i \tag{4.18d}$$

$$0 \leq \Delta^{\beta} \leq 1 \tag{4.18e}$$

$$\Delta^{\gamma} \geq 0 \tag{4.18f}$$

$$\min \sum_{j\in J} \Delta^{\gamma}_{j} \tag{4.19a}$$

$$\text{s.t.} \sum_{i\in g_{/\bar{G}}} (cov_S(i)-1)\Delta^{\beta} + \sum_{j\in J,\ \text{unused}} \Delta^{\gamma}_{j} = OBJ_1 \tag{4.19b}$$

$$(\Delta^{\beta}, \Delta^{\gamma}) \in P \tag{4.19c}$$

$$\min \sum_{j\in J} \sum_{(\ell,r)\in T_j} z_{j,\ell,r} \tag{4.20a}$$

$$\text{s.t.} \sum_{i\in g_{/\bar{G}}} (cov_S(i)-1)\Delta^{\beta} + \sum_{j\in J,\ \text{unused}} \Delta^{\gamma}_{j} = OBJ_1 \tag{4.20b}$$

$$\sum_{j\in J} \Delta^{\gamma}_{j} = OBJ_2 \tag{4.20c}$$

$$z_{j,\ell,r} = \Delta^{\gamma}_{j} - \sum_{i:\ell\leq i\leq r} \Delta^{\beta}_{i} \quad \forall j \in J, (\ell,r)\in \mathcal{P}_j \tag{4.20d}$$

$$(\Delta^{\beta}, \Delta^{\gamma}) \in P \tag{4.20e}$$

After selecting rates $\Delta^{\gamma}, \Delta^{\beta}$, computing our increase limit v and updating our dual variable accordingly, we will be in at least one of the following situations. The first situation is that there is some set of $I \subseteq g_{/\bar{G}}$ of intervals such that for $i \in I$ $\beta_i < \mu(i,0)$ before the current iteration, but currently $\beta_i = \mu(i,0)$. The second situation is that there is some set of placements $\mathcal{P}$ that were not tight in the previous round but are currently tight. In the latter case, we check if we can improve our solution S as in the first approach.

The complete algorithm is given in Algorithm 6.

We make a final note that connects us back to our motivation for studying this approach given at the beginning of section 4.4.1. When we select our rates $\Delta^{\gamma}, \Delta^{\beta}$, we are only motivated to assign positive values to these rates if doing so allows us to change β_i for some i, where i is covered multiple times by S. For each piece j, we are encouraging placements of lower individual profit that cover area of the ideal curve not currently covered by S.

Algorithm 6 Primal-Dual Increase ($\mathcal{I}$)

Input: An instance $\mathcal{I}$ of (CRPRPS)
Output: A solution S to $\mathcal{I}$.

```
 1: $S \leftarrow \emptyset$
 2: $\beta_i = 0, \pi_i = \mu(i,0)$ for all $i \in g_{/\bar{G}}$, $\gamma_j = p_{max}(j)$ for all $j \in J$, $T_j \leftarrow \{(\ell,r) \in \mathcal{P}_j : p(j,\ell,r) = p_{max}(j)\}$
    for $j \in J$
 3: $S \leftarrow LocalSearch(\mathcal{I}, \emptyset, J, \{T_j : j \in J\})$
 4: Solve (4.20) to get $\Delta^\gamma, \Delta^\beta$
 5: while Equation (4.18a) is positive do
 6:     $v^\gamma = \min_{(\ell,r)\in\binom{\bar{G}}{2}, v_{j,\ell,r}>0} v_{j,\ell,r}$
 7:     $v^\beta = \min_{i\in g_{/\bar{G}}} \frac{\beta_i}{\Delta^\beta_i}$
 8:     $v = \min(v^\gamma, v^\beta)$
 9:     $T'_j \leftarrow \{(\ell,r) \in \mathcal{P}_j : \sum_{i:\ell\le i\le r-1} \pi_i - \gamma_j = c(j,\ell,r)\}$ for each $j \in J$
10:     $\beta \leftarrow \beta - v\cdot\Delta^\beta, \pi \leftarrow \pi + v\cdot\Delta^\beta, \gamma \leftarrow \gamma + v\cdot\Delta^\gamma$
11:     $T_j \leftarrow \{(\ell,r) \in \mathcal{P}_j : \sum_{i:\ell\le i\le r-1} \pi_i - \gamma_j = c(j,\ell,r)\}$ for each $j \in J$
12:     $N_j \leftarrow T_j \setminus T'_j$ for each $j \in J$
13:     $S^* \leftarrow S$
14:     for $j \in J$ do
15:         for $(\ell,r) \in N_j \setminus T_j$ do
16:             $S' \leftarrow S$
17:             $S'_j \leftarrow (\ell,r)$
18:             $S' \leftarrow LocalSearch(\mathcal{I}, S', J\setminus\{j\}, \{T_j : j \in J\})$
19:             if $C_{max}(S') \ge C_{max}(S^*) + \epsilon$ then
20:                 $S^* \leftarrow S'$
21:     if $C_{max}(S^*) \ge C_{max}(S) + \epsilon$ then
22:         $S \leftarrow S^*$
23:     Solve (4.20) to get $\Delta^\gamma, \Delta^\beta$
    return $S$
```

4.4.5 Notes on primal-dual Approaches

In practice, after termination of our primal-dual algorithm, one could run the local search algorithm with the initial solution outputted by the primal-dual approach and expanding T_j to include even the placements are not tight with respect to 4.10c. At this point, we hypothesize that, on average, the local optimum we terminate with would be better than if we ran local search without the primal-dual techniques. Our hypothesis is made because of our approach of considering only placements low average cost or high profit in our search neighbourhood, then slowly relaxing this. This is because of the notes made in our motivation for studying these primal-dual techniques. This hypothesis is explored in an experimental study in chapter 5.

4.4.6 Unanswered Questions

As previously noted, our main goal in this chapter is to provide an algorithm that performs well and in polynomial time. We note that we can construct (CRPRPS) instances such that both of our primal-dual approaches give a solution with an asymptotic value of half of the optimal value. It would be interesting for future research to see if one can prove any approximation guarantees for these heuristics. It is also unclear if we can provide a worst-case run-time bound for these heuristics. If not, we would also be curious to see if appropriate modifications can be made to the algorithms to improve run-time or performance.

For the sake of future research and our experimental study, we define a measure associated with run-time. When we run our primal-dual approaches, for a given iteration, we say that combinatorial progress was made if either the objective function increased by some ϵ sufficiently large, or if the size of the set U which we will define for each approach separately decreases. For the decreasing primal-dual approach from section 4.4.3, the set U is given by

$$\{i \in g_{/\bar{G}} : cov_S(i) = 0, \beta_i > 0\}.$$

The intuition behind this choice is that in the decreasing approach, there is effectively no incentive to cover intervals i with $\beta_i = 0$. To see this, we recall that we are limited to placements (j, ℓ, r) tight with respect to constraint (4.10c). This means that the individual profit of a tight placement (j, ℓ, r) is $\gamma_j + \sum_{i:\ell \preceq_g i \prec_g r} \beta_i$. We could only increase our profit by using placement (j, ℓ, r) if some subset of the relevant dual variables has a positive value. Thus, if $\beta_i = 0$, there is no incentive to cover interval i.

For the increasing primal-dual approach from section 4.4.4, the set U is define as

$$\{i \in g_{/\bar{G}} : cov_S(i) > 1, \beta_i < \pi_i\}.$$

The intuition behind this choice is that in the increasing approach when we increase β_i for some $i \in g_{/\bar{G}}$, we do so when we have several options with which to cover interval i, and want to encourage the pieces which correspond to these options to explore other placements. This in effect disincentivizes covering interval i. When $\beta = \pi_i$ we can no longer disincentivize pieces to cover interval i.

For either approach, when $|U|$ decreases, either we have obtained more coverage, obtained less overlap, or the dimension of our rates system has decreased, making this a suitable quantity to track. For an appropriate choice of ϵ, we can bound the number of iterations in which combinatorial progress is made by a polynomial function in $|\bar{G}|$. As a result, one method to bound the run-time of either approach is to bound the number of consecutive iterations in which no combinatorial progress is made. In our experimental study in chapter 5, we measure this quantity to provide an intuition of whether or not studying these algorithms further is a good direction for the future of this project.

Chapter 5

Experimental Results

In this chapter, we conduct an experimental study to test the algorithms and heuristics given in Chapter 4. Recall that our heuristics from Chapter 4 optimize over equation (4.1). Thus in this section we will continue to frame (CRPR) and (CRPRPS) as maximization problems. First, we will examine the approaches targeted toward instances of (CRPRPS) in 5.1, and finally in Section 5.2, we examine our modified greedy approach for instances of (CRPR). In this experimental study, we used curves created to match the shape of a patient's preoperative bandeau, as well as the template the surgeon would have used. This is done by taking a CT scan of a patient's head and extracting data about the outer layer of bone. To create the deformed curve we place points along the bandeau, rotating them to a 2-dimensional plane and using them to define our curve. We have a library of ideal curves developed by the authors in [2], from which we took the ideal curve we used for this study. In practice the surgeons choose a template that can be placed onto the infant's head touching or almost touching the head at the midpoint and endpoints of the template. Analogously, our ideal curve is selected so that the deformed curve meets or is close to the ideal curve at the endpoints and midpoint. Taking discretizations with about 200 points captures enough detail to model the patient bandeaus effectively. Thus our curves f and g have the property that

$$|F| = |G| = 201.$$

We took $\bar{G} = G$ and $\bar{F}$ to contain every eighth point in F, beginning with the first point and including the last. This choice was made for the following two reasons. When testing our heuristics for (CRPRPS), if we took $\bar{F} = F$ for most choices of piece limit k we would be left with too many possible cut patterns to test in our time frame. In this case we would anyway select a subset of the possible cut patterns to test. The second reason is that when testing our modified greedy heuristics for (CRPR) we wanted to test instances which we could solve optimally to observe how our heuristics perform in practice. With our computing power we were unable to solve instances of (CRPR) using $\bar{F} = F, \bar{G} = G$ for any k. However, taking $\bar{F}$ to be one eighth the size of F, we can quickly solve instances of (CRPR) using our computing power.

We mentioned that the surgeons would like to limit the number of cuts made to the deformed skull in order to limit the length of the procedure. In practice, the surgeons typically make up to 7 cuts, resulting in up to 8 bandeau pieces. Two cuts will typically be too few. As a result, we elected to test values of k, ranging from 3 to 8. We used the choice of c and μ defined in Section 2.2.2, with parameters $\alpha = 1$ and $\delta = 0.005$.

For the following tests we have implemented our heuristics using Python.

5.1 Experiments for Instances of CRPRPS

We will first explain how our study of (CRPRPS) will proceed. To motivate our choice of dataset we consider the following quantity. For a given set of pieces J, and for each piece $j \in J$, we let m_j be the maximum number of intervals covered by any placement of piece j with a non-infinite cost. The formal equation for m_j is given in (5.1), where we recall that $\mathcal{P}_j$ is the set of placements of piece j with a non-infinite cost. We call the quantity $\sum_{j \in J} m_j$, the maximum possible coverage for piece set J because it provides an upper bound on the number of intervals any solution to (CRPRPS) specified by J covers.

$$m_j = \max_{(\ell, r) \in \mathcal{P}_j} |\{\ell, \dots, r-1\}| \tag{5.1}$$

We create our data set based on the following intuition. For our application, we are interested in covering most of the unit intervals of the ideal curve. We note that if the maximum possible coverage of a set of pieces is too low, it cannot lead to a suitable solution. Since our heuristics for (CRPRPS) do not provide intuition on generating cut patterns best, we will be interested in how our algorithms perform on cut patterns that have been preselected to be suitable. For this reason, we have chosen to favour testing on instances of (CRPRPS) with high values of maximum possible coverage. The data set is meant to filter further by selecting (CRPRPS) instances whose optimal solution uses all inputted pieces. We experiment on the this dataset to see how our algorithms perform on instances with optimal solutions that cover almost the entire ideal curve.

We next describe how the data set is created. For a given k, we made test instances using the following procedure. We first enumerated all sets of k pieces that can be obtained using the same separation operation. We iterate through our list, and for each piece set J, we compute the optimal solution using an integer programming formulation for (CRPRPS) implemented using Gurobi's Python interface. If the optimal solution places all pieces in J, we place it into the dataset. We limit each data set to 25000 examples.

We will use tables to present our results. For each heuristic, we will compute the ratio between the objective value of the solution produced by the heuristic and the optimal objective value, which we will call the primal performance ratio. Some of our approaches use primal-dual techniques. In addition to testing the primal performance ratio, for such approaches, we will be interested in the ratio between the value of the solution produced by the heuristic and the objective value of the dual solution produced by the heuristic, which we will call the dual performance ratio. Specifically these approaches are the resource allocation approach in Section 4.2, and the primal-dual local search approaches from Section 4.4.

For each choice of k, we will provide two tables, one measuring the primal performance ratios for our data set and one measuring the dual performance ratios for our data set whenever relevant. In each table's caption, we will specify the value k, the size of our data set, and which ratio we are testing. The rows of the table will correspond to the heuristic or algorithm we are testing. We will use the titles Greedy, LS, RA, PD DEC, PD DEC LS, PD INC, and PD INC LS respectively to present our results for the standard greedy approach in section 4.1, the local search approach in section 4.3, the resource allocation approach in section 4.2, the decreasing primal-dual approach in section 4.4.3, the decreasing primal-dual approach in section 4.4.3 with an additional round of local search at the end where we expand T_j to $\mathcal{P}_j$ for each piece $j \in J$, the increasing primal-dual approach in section 4.4.4, and the increasing primal-dual approach in section 4.4.4 with an additional round of local search at the end where we expand T_j to $\mathcal{P}_j$ for each piece $j \in J$, respectively. Ranges will label the columns of our charts. The entry for a particular column will be the percentage of the test data whose performance ratio was in the specified range for the specified heuristic or algorithm. In essence, our charts are meant as appropriate substitutes for histograms. Note that the entries of our tables were rounded to two, sometimes three decimal places, allowing for some error.

We first summarize our data by describing how each heuristic performed for each choice of k as well as over all. The reader should refer to Tables 5.1 and 5.2 for performance statistics on our heuristics for $k = 3$, Tables 5.3 and 5.4 for performance statistics on our heuristics for $k = 4$, Tables 5.5 and 5.6 for performance statistics on our heuristics for $k = 5$, Tables 5.7 and 5.8 for performance statistics on our heuristics for $k = 6$, Tables 5.9 and 5.10 for performance statistics on our heuristics for $k = 7$, and Tables 5.11 and 5.12

for performance statistics on our heuristics for $k = 8$. For the reader's convenience we provide a full analysis as well as a summary of it afterward.

5.1.1 Full Analysis of Performance

We examine each heuristic one by one.

Primal Performance Ratios Greedy Heuristic We first examine the performance of the greedy heuristic. For $k = 3$, the heuristic gave an optimal solution on 28.78% of the instances, 65.84% of the cases had a primal performance ratio of at least 0.9 and 85.92% of the cases had a primal performance ratio of at least 0.8. For $k = 4$, the heuristic gave an optimal solution on 15.93% of the instances, 71.02% of the instances had a primal performance ratio of at least 0.9 and 94.55% of the instances had a primal performance ratio of at least 0.8. For $k = 5$, the heuristic gave an optimal solution on 5.53% of the instances, 70.77% of the instances had a primal performance ratio of at least 0.9 and 97.67% of the instances had a primal performance ratio of at least 0.8. For $k = 6$, the heuristic gave an optimal solution on 1.52% of the instances, 73.62% of the instances had a primal performance ratio of at least 0.9 and 98.78% of the instances had a primal performance ratio of at least 0.8. For $k = 7$, the heuristic gave an optimal solution on 1.28% of the instances, 84.92% of the instances had a primal performance ratio of at least 0.9 and 99.84% of the instances had a primal performance ratio of at least 0.8. For $k = 8$, the heuristic gave an optimal solution on 0.21% of the instances, 93.53% of the instances had a primal performance ratio of at least 0.9 and 99.96% of the instances had a primal performance ratio of at least 0.8. For our test data, the likelihood that the algorithm gives the optimal solution appears to decrease as k increases. Over our entire data set, approximately 78.96% of the instances had a primal performance ratio of at least 0.9, and 98.17% of the instances had a primal performance ratio of at least 0.8, suggesting a strong over all performance.

Primal Performance Ratios Local Search Heuristic Next, we examine the performance of the local search heuristic. For $k = 3$, the heuristic gave an optimal solution on 44.93% of the instances, 68.33% of the instances had a primal performance ratio of at least 0.9 and 86.34% of the instances had a primal performance ratio of at least 0.8. For $k = 4$, the heuristic gave an optimal solution on 26.46% of the instances, 74.34% of the instances had a primal performance ratio of at least 0.9 and 95.57% of the instances had a primal performance ratio of at least 0.8. For $k = 5$, the heuristic gave an optimal solution on 9.68% of the instances, 74.28% of the instances had a primal performance ratio of at least 0.9 and 98.1% of the instances had a primal performance ratio of at least 0.8. For $k = 6$, the heuristic gave an optimal solution on 3.32% of the instances, 76.42% of the instances had a primal performance ratio of at least 0.9 and 98.97% of the instances had a primal performance ratio of at least 0.8. For $k = 7$, the heuristic gave an optimal solution on 2.6% of the instances, 88.25% of the instances had a primal performance ratio of at least 0.9 and 99.92% of the instances had a primal performance ratio of at least 0.8. For $k = 8$, the heuristic gave an optimal solution on 1.1% of the instances, 97.06% of the instances had a primal performance ratio of at least 0.9, and almost all of the instances had a primal performance ratio of at least 0.8. Again, for our test data, the likelihood that the local search heuristic gives the optimal solution appears to decrease as k increases. Again the over all performance remained strong as k increased since about 82.26% of the instances had a primal performance ratio of at least 0.9 and 97.89% of the instances had a primal performance ratio of at least 0.8.

Primal Performance Ratios Resource Allocation Algorithm Next, we examine the performance of the resource allocation algorithm. For $k = 3$, the heuristic gave an optimal solution on 12.22% of the instances, 70.4% of the instances had a primal performance ratio of at least 0.9 and 80.13% of the instances had a primal performance ratio of at least 0.8. For $k = 4$, the heuristic gave an optimal solution on 2.67% of the instances, 57.61% of the instances had a primal performance ratio of at least 0.9 and 80.81% of the instances had a primal performance ratio of at least 0.8. For $k = 5$, the heuristic gave an optimal solution on 0.44% of the instances, 47.78% of the instances had a primal performance ratio of at least 0.9 and 81.72% of the instances had a primal performance ratio of at least 0.8. For $k = 6$, the heuristic gave an optimal solution on 0.06% of the instances, 42.02% of the instances had a primal performance ratio of at least 0.9 and 85.91% of the instances had a primal performance ratio of at least 0.8. For $k = 7$, the heuristic gave an optimal solution on 0.01% of the instances, 64% of the instances had a primal performance ratio of at least 0.9 and 96.19% of the instances had a primal performance ratio of at least 0.8. For $k = 8$, the heuristic gave an optimal solution on 0.01% of the instances, 59.51% of the instances had a primal performance ratio of at least 0.9 and 97.83% of the instances had a primal performance ratio of at least 0.8. Again, for our test data, the likelihood that the resource allocation algorithm gives the optimal solution appears to decrease as k increases. Although in general not as strong at the greedy and local search ratios, over all performance was strong and consistent, since about 54.06% of the instances had a primal performance ratio of at least 0.9 and 88.69% of the instances had a primal performance ratio of at least 0.8

Primal Performance Decreasing Primal-Dual Heuristic Next, we examine the performance of the decreasing primal-dual heuristic. For $k = 3$, the heuristic gave an optimal solution on 44.93% of the instances, 96.28% of the instances had a primal performance ratio of at least 0.9 and all of the instances had a primal performance ratio of at least 0.8. For $k = 4$, the heuristic gave an optimal solution on 34.69% of the instances, 97.46% of the instances had a primal performance ratio of at least 0.9, and almost all of the instances had a primal performance ratio of at least 0.8. For $k = 5$, the heuristic gave an optimal solution on 34.47% of the instances, 97.97% of the instances had a primal performance ratio of at least 0.9, and almost all of the instances had a primal performance ratio of at least 0.8. For $k = 6$, the heuristic gave an optimal solution on 34.55% of the instances, 97.76% of the instances had a primal performance ratio of at least 0.9, and almost all of the instances had a primal performance ratio of at least 0.8. For $k = 7$, the heuristic gave an optimal solution on 47.26% of the instances, 99.76% of the instances had a primal performance ratio of at least 0.9 and all of the instances had a primal performance ratio of at least 0.8. For $k = 8$, the heuristic gave an optimal solution on 43.06% of the instances, 99.94% of the instances had a primal performance ratio of at least 0.9 and all of the instances had a primal performance ratio of at least 0.8. This heuristic performed well and consistently, giving an optimal solution on 38.99% of the instances, 98.53% of the instances had a primal performance ratio of at least 0.9, and almost all of the instances had a primal performance ratio of at least 0.8. We note that this heuristic performed better than the previously mentioned heuristics, increasing the percentage of instances falling into each of the three classes, namely those giving a primal performance ratio of $1, 0.9$, and of 0.8.

Now we discuss the performance of the decreasing primal-dual heuristic with the additional round of local search with an expanded neighbourhood and compare it to that of the the standard local search heuristic. For $k = 3$, the class of instances for which we were able to obtain the optimal solution included an additional 36.85% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 30.43% of the instances, and the class of instances for which we were able to obtain a performance ratio of at least 0.8 included and additional 13.67% of the instances. For $k = 4$, the class of instances for which we were able to obtain the optimal solution included an additional 39.15% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 23.74% of the instances, and the class of instances for which we were able to obtain a performance ratio of at least 0.8 included and additional 4.38% of the instances. For $k = 5$, the class of instances for which we were able to obtain the optimal solution included an additional 43.81% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 23.84% of the instances, and the class of instances for which we were able to obtain a performance ratio of at least 0.8 included and additional 1.85% of the instances. For $k = 6$, the class of instances for which we were able to obtain the optimal solution included an additional 43.28% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 21.44% of the instances, and the class of instances for which we were able to obtain a performance ratio of at least 0.8 included and additional 1.03% of the instances. For $k = 7$, the class of instances for which we were able to obtain the optimal solution included an additional 53.44% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 11.57% of the instances, and the class of instances for which we were able to obtain a performance ratio of at least 0.8 included and additional 0.08% of the instances. For $k = 8$, the class of instances for which we were able to obtain the optimal solution included an additional 48.11% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 2.89% of the instances, and the class of instances for which we were able to obtain a performance ratio of at least 0.8 was not changed as all instance for both heuristics achieved this value. In summary, over all tested instances the addition of the primal-dual steps allowed us to increase the class of instances for which we were able to obtain the optimal solution to include an additional 45.78% of the total instances. We were able to obtain a performance ratio of at least 0.9 on an additional 16.45% of the instances, with little change in the class of instances for which we were able to obtain a performance ratio of at least 0.8 as both heuristics gave performance ratios of at least 0.8 on most instances.

Next we discuss the performance of the decreasing primal-dual heuristic with the additional round of local search and compare it to that of the the standard decreasing primal-dual heuristic. For $k = 3$, the class of instances for which we were able to obtain the optimal solution included an additional 36.85% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 2.46% of the instances, and both heuristics obtained a performance ratio of at least 0.8 for all instances. For $k = 4$, the class of instances for which we were able to obtain the optimal solution included an additional 30.92% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 0.62% of the instances, and both heuristics obtained a performance ratio of at least 0.8 for almost all instances. For $k = 5$, the class of instances for which we were able to obtain the optimal solution included an additional 19.02% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 0.15% of the instances, and both heuristics obtained a performance ratio of at least 0.8 for almost all instances. For $k = 6$, the class of instances for which we were able to obtain the optimal solution included an additional 12.05% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 0.1% of the instances,, and both heuristics obtained a performance ratio of at least 0.8 for almost all instances. For $k = 7$, the class of instances for which we were able to obtain the optimal solution included an additional 8.78% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 0.03% of the instances, and both heuristics obtained a performance ratio of at least 0.8 for almost all instances. For $k = 8$, the class of instances for which we were able to obtain the optimal solution included an additional 6.15% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 0.01% of the instances, and the class of instances for which we were able to obtain a performance ratio of at least 0.8 was not changed as all instance for both heuristics achieved this value. In summary, the addition of extra round of local search allowed us to increase the class of instances for which we were able to obtain the optimal solution to include an additional 14.83% of the total instances. There was little change in the classes of instances for which we were able to obtain a performance ratio of at least 0.9 and 0.8, respectively as both heuristics gave similar performance in these categories with each heuristic admitting a ratio of at least 0.8 in most instances.

Primal Performance Increasing Primal-Dual Heuristic Lastly, we examine the performance of the increasing primal-dual heuristic. For $k = 3$, the heuristic gave an optimal solution on 85.3% of the instances, 98.96% of the instances had a primal performance ratio of at least 0.9 and almost all of the instances had a primal performance ratio of at least 0.8. For $k = 4$, the heuristic gave an optimal solution on 73.22% of the instances, 97.57% of the instances had a primal performance ratio of at least 0.9, and almost all of the instances had a primal performance ratio of at least 0.8. For $k = 5$, the heuristic gave an optimal solution on 58.97% of the instances, 96.67% of the instances had a primal performance ratio of at least 0.9, and almost all of the instances had a primal performance ratio of at least 0.8. For $k = 6$, the heuristic gave an optimal solution on 46.39% of the instances, 95.72% of the instances had a primal performance ratio of at least 0.9, and almost all of the instances had a primal performance ratio of at least 0.8. For $k = 7$, the heuristic gave an optimal solution on 45.14% of the instances, 98.81% of the instances had a primal performance ratio of at least 0.9 and all of the instances had a primal performance ratio of at least 0.8. For $k = 8$, the heuristic gave an optimal solution on 35.96% of the instances, 99.53% of the instances had a primal performance ratio of at least 0.9 and all of the instances had a primal performance ratio of at least 0.8. This heuristic performed well and consistently, giving an optimal solution on 51.16% of the instances, 97.58% of the instances had a primal performance ratio of at least 0.9, and almost all of the instances had a primal performance ratio of at least 0.8.

Now we discuss the performance of the increasing primal-dual heuristic with the additional round of local search and compare it to that of the the standard local search heuristic. For $k = 3$, the class of instances for which we were able to obtain the optimal solution included an additional 45.34% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 30.64% of the instances, and the class of instances for which we were able to obtain a performance ratio of at least 0.8 included and additional 13.67% of the instances. For $k = 4$, the class of instances for which we were able to obtain the optimal solution included an additional 54.11% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 30.58% of the instances, and the class of instances for which we were able to obtain a performance ratio of at least 0.8 included and additional 4.21% of the instances. For $k = 5$, the class of instances for which we were able to obtain the optimal solution included an additional 56.22% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 22.74% of the instances, and the class of instances for which we were able to obtain a performance ratio of at least 0.8 included and additional 1.72% of the instances. For $k = 6$, the class of instances for which we were able to obtain the optimal solution included an additional 48.16% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 19.86% of the instances, and the class of instances for which we were able to obtain a performance ratio of at least 0.8 included and additional 1.01% of the instances. For $k = 7$, the class of instances for which we were able to obtain the optimal solution included an additional 50.53% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 10.93% of the instances, and the class of instances for which we were able to obtain a performance ratio of at least 0.8 included and additional 0.08% of the instances. For $k = 8$, the class of instances for which we were able to obtain the optimal solution included an additional 43.14% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 to include an additional 2.74% of the instances, The class of instances for which we were able to obtain a performance ratio of at least 0.8 was not changed as all instance for both heuristics achieved this value. In summary, over all tested instances the addition of the primal-dual steps allowed us to increase the class of instances for which we were able to obtain the optimal solution to include an additional 50.22% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 16.87% of the instances, with little change in the class of instances for which we were able to obtain a performance ratio of at least 0.8 as both heuristics gave performance ratios of 0.8 on most instances. Next we discuss the performance of the increasing primal-dual heuristic with the additional round of local search and compare it to that of the the standard increasing primal-dual heuristic. For $k = 3$, the class of instances for which we were able to obtain the optimal solution included an additional 4.97% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 0.01% of the instances, and both heuristics obtained a performance ratio of at least 0.8 for all instances. For $k = 4$, the class of instances for which we were able to obtain the optimal solution included an additional 7.35% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 0.26% of the instances, and both heuristics obtained a performance ratio of at least 0.8 for all instances. For $k = 5$, the class of instances for which we were able to obtain the optimal solution included an additional 6.93% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 0.35% of the instances, and both heuristics obtained a performance ratio of at least 0.8 for all instances. For $k = 6$, the class of instances for which we were able to obtain the optimal solution included an additional 5.09% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 4.62% of the instances, and both heuristics obtained a performance ratio of at least 0.8 for all instances. For $k = 7$, the class of instances for which we were able to obtain the optimal solution included an additional 7.99% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 0.37% of the instances, and both heuristics obtained a performance ratio of at least 0.8 for all instances. For $k = 8$, the class of instances for which we were able to obtain the optimal solution included an additional 8.28% of the total instances, the class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 0.27% of the instances, and the class of instances for which we were able to obtain a performance ratio of at least 0.8 was not changed as all instance for both heuristics achieved this value. In summary, over all tested instances the additional of round of local search allowed us to increase the class of instances for which we were able to obtain the optimal solution to include 7.1% more of the total instances, with little change in the classes of instances for which we were able to obtain a performance ratio of at least 0.9 and 0.8, respectively as both heuristics gave similar performance in these categories with each heuristic admitting a ratio of at least 0.8 on most instances.

Dual Performance Ratios We examine the dual performance ratios given in our various primal-dual approaches.

Dual Performance Resource Allocation Algorithm First, we examine the performance of the resource allocation algorithm. For $k = 3$, the heuristic never gave an optimal pair of solutions, 30.85% of the instances had a dual performance ratio of at least 0.9 and 60.04% of the instances had a dual performance ratio of at least 0.8. For $k = 4$, the heuristic never gave an optimal pair of solutions, 12.94% of the instances had a dual performance ratio of at least 0.9 and 43% of the instances had a dual performance ratio of at least 0.8. For $k = 5$, the heuristic never gave an optimal pair of solutions, 2.84% of the instances had a dual performance ratio of at least 0.9 and 25.56% of the instances had a dual performance ratio of at least 0.8. For $k = 6$, the heuristic never gave an optimal pair of solutions, 0.03% of the instances had a dual performance ratio of at least 0.9 and 10.2% of the instances had a dual performance ratio of at least 0.8. For $k = 7$, the heuristic never gave an optimal pair of solutions, 0.02% of the instances had a dual performance ratio of at least 0.9 and 15.64% of the instances had a dual performance ratio of at least 0.8. For $k = 8$, the heuristic never gave an optimal pair of solutions, 0.01% of the instances had a dual performance ratio of at least 0.9 and 10.65% of the instances had a dual performance ratio of at least 0.8. Comparing these results with the primal performance ratios of this algorithm we see that solutions are typically much better than the dual bound offered. We also observe that on our test set the dual performance ratio appears to drop as k increase.

Dual Performance Decreasing Primal-Dual Heuristic First, we examine the performance of the decreasing primal-dual heuristic. For $k = 3$, the heuristic gave an optimal pair of solutions 39.13% of the time, 79.71% of the instances had a dual performance ratio of at least 0.9 and 98.34% of the instances had a dual performance ratio of at least 0.8. For $k = 4$, the heuristic gave an optimal pair of solutions 24.6% of the time, 89.13% of the instances had a dual performance ratio of at least 0.9 and 99.78% of the instances had a dual performance ratio of at least 0.8. For $k = 5$, the heuristic gave an optimal pair of solutions 26.78% of the time, 95.68% of the instances had a dual performance ratio of at least 0.9 and 99.93% of the instances had a dual performance ratio of at least 0.8. For $k = 6$, the heuristic gave an optimal pair of solutions 31% of the time, 96.68% of the instances had a dual performance ratio of at least 0.9 and 100% of the instances had a dual performance ratio of at least 0.8. For $k = 7$, the heuristic gave an optimal pair of solutions 42.46% of the time, 99.76% of the instances had a dual performance ratio of at least 0.9 and 100% of the instances had a dual performance ratio of at least 0.8. For $k = 8$, the heuristic gave an optimal pair of solutions 38.9% of the time, 99.93% of the instances had a dual performance ratio of at least 0.9 and 100% of the instances had a dual performance ratio of at least 0.8. Recalling our observations from examining our primal performance ratios, we obtain the following. Over all cases, the heuristic gave an optimal pair of solutions 33.07% of the time while giving the optimal primal solution 38.96% of the time. The heuristic gave a dual performance ratio of at least 0.9 96.37% of the time and a primal performance ratio of at least 0.9 98.53% of the time. The heuristic gave a dual and primal performance ratios of at least 0.8 on almost all instances. We conclude that on our test set the decreasing primal-dual heuristic produces very good quality dual solutions.

Dual Performance Increasing Primal-Dual Heuristic First, we examine the performance of the decreasing primal-dual heuristic. For $k = 3$, the heuristic gave an optimal pair of solutions 57.35% of the time, 97.93% of the instances had a dual performance ratio of at least 0.9 and 99.79% of the instances had a dual performance ratio of at least 0.8. For $k = 4$, the heuristic gave an optimal pair of solutions 32.45% of the time, 95.99% of the instances had a dual performance ratio of at least 0.9 and 99.78% of the instances had a dual performance ratio of at least 0.8. For $k = 5$, the heuristic gave an optimal pair of solutions 28.62% of the time, 96.08% of the instances had a dual performance ratio of at least 0.9 and 99.8% of the instances had a dual performance ratio of at least 0.8. For $k = 6$, the heuristic gave an optimal pair of solutions 29.44% of the time, 95.38% of the instances had a dual performance ratio of at least 0.9 and 99.95% of the instances had a dual performance ratio of at least 0.8. For $k = 7$, the heuristic gave an optimal pair of solutions 19.73% of the time, 98.71% of the instances had a dual performance ratio of at least 0.9 and 99.99% of the instances had a dual performance ratio of at least 0.8. For $k = 8$, the heuristic gave an optimal pair of solutions 15.73% of the time, 99.5% of the instances had a dual performance ratio of at least 0.9 and 100% of the instances had a dual performance ratio of at least 0.8. Looking at the data, as k increases there appears to be a drop in the percentage of instances for which the heuristic produces a primal dual pair. Again, recalling our observations from examining our primal performance ratios, we obtain the following. Over all cases, the heuristic gave an optimal pair of solutions 25.01% of the time while giving the optimal primal solution 51.16% of the time. The heuristic gave a dual performance ratio of at least 0.9 97.1% of the time and a primal performance ratio of at least 0.9 97.58% of the time. The heuristic gave a dual and primal performance ratios of at least 0.8 on almost all instances. We conclude that on our test set the decreasing primal-dual heuristic produces very good quality dual solutions.

This heuristic performed well and consistently, giving an optimal solution on 51.16% of the instances, 97.58% of the instances had a primal performance ratio of at least 0.9, and almost all of the instances had a primal performance ratio of at least 0.8. We note that this heuristic performed similarly to the decreasing primal-dual heuristic, giving the optimal solution over 10% more often.

5.1.2 Summary of Performance Analysis

For our test data, the likelihood that the greedy algorithm gives the optimal solution appears to decrease as k increases. For $k = 3, 4$ we achieved an optimal solution for 28.78% and 15.93% of the instances, respectively, whereas for $k = 7, 8$ we achieved an optimal solution for 1.28% and 0.21% of the instances respectively. Over our entire data set, approximately 78.96% of the instances had a primal performance ratio of at least 0.9, and 98.17% of the instances had a primal performance ratio of at least 0.8, suggesting that the greedy algorithm performed well.

For our test data, the likelihood that the local search heuristic gives the optimal solution also appears to decrease as k increases. For $k = 3, 4$ we achieved an optimal solution for 44.93% and 26.46% of the instances respectively, whereas for $k = 7, 8$ we achieved an optimal solution for 2.6% and 1.1% of the instances respectively. Over our entire data set, approximately 82.26% of the instances had a primal performance ratio of at least 0.9, and 97.89% of the instances had a primal performance ratio of at least 0.8. This suggests that overall the local search heuristic performed well.

For our test data, the likelihood that the resource allocation algorithm gives the optimal solution also appears to decrease as k increases. For $k = 3, 4$ we achieved an optimal solution for 12.22% and 2.67% of the instances respectively, whereas for $k = 7, 8$ we achieved an optimal solution for 0.01% of the instances for both k. Over our entire data set, approximately 54.06% of the instances had a primal performance ratio of at least 0.9, and 88.69% of the instances had a primal performance ratio of at least 0.8. The observed performance is not as strong as the previous two approaches.

For our test data, the likelihood that the decreasing primal-dual approach gives the optimal solution appears to remain more consistent than the previous approaches. For $k = 3, 4$ we achieved an optimal solution for 44.93% and 34.69% of the instances, respectively, whereas for $k = 7, 8$ we achieved an optimal solution for 47.26% and 43.06% of the instances respectively. Over our entire data set, the approach gave an optimal solution on 38.99% of the instances. Whereas 98.53% of the instances had a primal performance ratio of at least 0.9, and almost all of the instances had a primal performance ratio of at least 0.8. Overall this approach performed stronger than the above three, increasing the number of instances in which we obtained an optimal or close to optimal solution. We also tested how this heuristic fared with an additional round of local search at the end, where we expanded our neighbourhood to include all possible placements for each piece. Comparing the solutions we obtained from doing only the standard local search, we obtained the following results. The decreasing primal-dual approach with the extra round of local search gave the optimal solution for an additional 45.78% of the total instances. The class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 16.45% of the instances, with little change in the class of instances for which we were able to obtain a performance ratio of at least 0.8 as both heuristics gave performance ratios of at least 0.8 on most instances. Also, we compared how the additional round of local search at the end improved upon the decreasing primal-dual approach. Over all tested instances, the addition of an extra round of local search allowed us to increase the class of instances for which we were able to obtain the optimal solution to include an additional 14.83% of the total instances. There was little change in the classes of instances for which we were able to obtain a performance ratio of at least 0.9 and 0.8, respectively as both heuristics gave a similar performance in these categories with each heuristic admitting a ratio of at least 0.8 on most instances.

The increasing primal-dual approach also performed well, giving an optimal solution in 51.16% of the instances, 97.58% of the instances had a primal performance ratio of at least 0.9, and almost all of the instances had a primal performance ratio of at least 0.8. Comparing the solutions we obtained from doing only the standard local search, we obtained the optimal solution included an additional 50.22% of the total instances. The class of instances for which we were able to obtain a performance ratio of at least 0.9 included an additional 16.87% of the instances, with little change in the class of instances for which we were able to obtain a performance ratio of at least 0.8 as both heuristics gave performance ratios of 0.8 on most instances. Comparing how the additional round of local search at the end improved upon the decreasing primal-dual approach, we were able to obtain the optimal solution in 7.1% more of the total instances. There was little change in the classes of instances for which we were able to obtain a performance ratio of at least 0.9 and 0.8, respectively as both heuristics gave a similar performance in these categories with each heuristic admitting a ratio of at least 0.8 on most instances.

We also analyzed the dual performance ratios. We found that the resource allocation algorithm typically provides a much better primal solution than dual solution. In contrast, in most cases for our primal-dual local search heuristics, the dual and primal performance ratios were much similar. In particular, over all cases, the decreasing heuristic gave an optimal pair of solutions 33.07% of the time while giving the optimal primal solution 38.96% of the time. The decreasing heuristic gave a dual performance ratio of at least 0.9 96.37% of the time and a primal performance ratio of at least 0.9 98.53% of the time. The decreasing heuristic gave dual and primal performance ratios of at least 0.8 on almost all instances. Similarly, over all cases, the increasing heuristic gave an optimal pair of solutions 33.07% of the time while giving the optimal primal solution 38.96% of the time. The increasing heuristic gave a dual performance ratio of at least 0.9 96.37% of the time and a primal performance ratio of at least 0.9 98.53% of the time. The increasing heuristic gave dual and primal performance ratios of at least 0.8 on almost all instances.

We did not find an example where any of our heuristics returned a solution worth less than half the optimum, with most of our examples concentrated at at least 80% of the optimum. On our dataset, the strongest performing heuristics were the primal-dual local search approaches.

5.1.3 Worst-Case Run-Time Analysis

Since it is still unclear what the worst-case run-time is for our primal-dual approaches, we also took note of the maximum observed number of consecutive iterations without combinatorial progress, as defined in section 4.4.6 when we ran these heuristics. Over all choices of k, the worst observed value for this quantity in the decreasing primal-dual approach from section 4.4.3 was 75, while the increasing primal-dual approach from section 4.4.4 had a worst observed value for this quantity of 27975. We conclude that for out test data the decreasing primal-dual approach takes less iterations to make combinatorial progress than the increasing approach. For more complete statistics, please see Table 5.13, where for each k and each primal-dual method, we record the maximum observed value and the average over all test cases. It appears from this table that as k increases, the number of iterations also increases and that for our test set, the increasing algorithm requires significants more iterations on average. We also computed the CPU seconds required to run each instance in our test set, and in Table 5.14 recorded the average value for each k. It appears from this table that as k increases, that the number of CPU required to run either primal-dual local search heuristic also increases. Although we concluded that the increasing approach required more iterations, it appears that, on average, the number of CPU seconds required to run an instance of the increasing approach is less than that of the decreasing approach.

Table 5.1: k=3, 483 instances, Primal Rerformance Ratios

	[0.5- 0.6)	[0.6- 0.7)	[0.7- 0.8)	[0.8- 0.9)	[0.9- 1)	1
Greedy	-	1.66	12.42	20.08	37.06	28.78
LS	-	1.66	12.01	18.01	23.4	44.93
RA	0.21	4.76	14.91	9.73	58.18	12.22
PD DEC	-	-	-	3.73	51.35	44.93
PD DEC LS	-	-	-	1.24	16.98	81.78
PD INC	-	-	0.21	0.83	13.66	85.3
PD INC LS	-	-	-	1.04	8.7	90.27

Table 5.2: k=3, 483 instances, Dual Performance Ratios

	[0.5- 0.6)	[0.6- 0.7)	[0.7- 0.8)	[0.8- 0.9)	[0.9- 1)	1
RA	4.14	17.39	18.42	29.19	30.85	-
PD DEC	-	-	1.66	18.63	40.58	39.13
PD INC	-	-	0.21	6.21	36.23	57.35

Table 5.3: k=4, 20107 instances, Primal Performance Ratios

	[0.5- 0.6)	[0.6- 0.7)	[0.7- 0.8)	[0.8- 0.9)	[0.9- 1)	1
Greedy	0.02	0.21	5.22	23.53	55.09	15.93
LS	0.02	0.14	4.27	21.23	47.88	26.46
RA	0.1	4.77	14.32	23.2	54.94	2.67
PD DEC	-	-	0.04	2.49	62.77	34.69
PD DEC LS	-	-	0.04	1.87	32.47	65.61
PD INC	-	0.005	0.21	2.21	24.35	73.22
PD INC LS	-	0.005	0.18	1.98	17.26	80.57

Table 5.4: k=4, 20107 instances, Dual Performance Ratios

	[0.5- 0.6)	[0.6- 0.7)	[0.7- 0.8)	[0.8- 0.9)	[0.9- 1)	1
RA	7.03	23.38	26.6	30.06	12.94	-
PD DEC	-	-	0.21	10.65	64.53	24.6
PD INC	-	0.005	0.22	3.79	63.54	32.45

Table 5.5: k=5, 25000 instances, Primal Performance Ratios

	[0.5- 0.6)	[0.6- 0.7)	[0.7- 0.8)	[0.8- 0.9)	[0.9- 1)	1
Greedy	-	0.12	2.23	26.9	65.24	5.53
LS	-	0.1	1.8	23.82	64.6	9.68
RA	0.01	2.23	16.04	33.94	47.34	0.44
PD DEC	-	-	0.05	1.97	63.5	34.47
PD DEC LS	-	-	0.05	1.83	44.63	53.49
PD INC	-	0.02	0.18	3.14	37.7	58.97
PD INC LS	-	0.012	0.17	2.8	31.12	65.896

Table 5.6: k=5, 25000 instances, Dual Performance Ratios

	[0.5- 0.6)	[0.6- 0.7)	[0.7- 0.8)	[0.8- 0.9)	[0.9- 1)	1
RA	8.36	33.12	32.96	22.72	2.84	-
PD DEC	-	-	0.07	4.25	68.9	26.78
PD INC	-	0.02	0.18	3.72	67.46	28.62

Table 5.7: k=6, 25000 instances, Primal Performance Ratios

	[0.5- 0.6)	[0.6- 0.7)	[0.7- 0.8)	[0.8- 0.9)	[0.9- 1)	1
Greedy	-	0.01	1.21	25.16	72.1	1.52
LS	-	0.01	1.02	22.55	73.1	3.32
RA	-	0.79	13.3	43.89	41.96	0.06
PD DEC	-	-	0.004	2.23	63.21	34.55
PD DEC LS	-	-	0.004	2.14	51.26	46.6
PD INC	-	-	0.05	4.22	49.33	46.39
PD INC LS	-	-	0.02	3.696	44.8	51.48

Table 5.8: k=6, 25000 instances, Dual Performance Ratios

	[0.5- 0.6)	[0.6- 0.7)	[0.7- 0.8)	[0.8- 0.9)	[0.9- 1)	1
RA	5.9	44.77	39.13	10.17	0.03	-
PD DEC	-	-	0.04	3.32	65.68	31.0
PD INC	-	-	0.06	4.57	65.94	29.44

Table 5.9: k=7, 25000 instances, Primal Performance Ratios

	[0.5- 0.6)	[0.6- 0.7)	[0.7- 0.8)	[0.8- 0.9)	[0.9- 1)	1
Greedy	-	-	0.164	14.92	83.64	1.28
LS	-	-	0.09	11.67	85.65	2.6
RA	-	0.04	3.77	32.19	63.99	0.01
PD DEC	-	-	-	0.21	52.53	47.26
PD DEC LS	-	-	-	0.18	43.78	56.04
PD INC	-	-	0.004	1.19	53.67	45.14
PD INC LS	-	-	-	0.82	46.05	53.13

Table 5.10: k=7, 25000 instances, Dual Performance Ratios

	[0.5- 0.6)	[0.6- 0.7)	[0.7- 0.8)	[0.8- 0.9)	[0.9- 1)	1
RA	0.3	18.2	65.86	15.62	0.02	-
PD DEC	-	-	-	0.24	57.3	42.46
PD INC	-	-	0.004	1.28	78.98	19.73

Table 5.11: k=8, 25000 instances, Primal Performance Ratios

	[0.5- 0.6)	[0.6- 0.7)	[0.7- 0.8)	[0.8- 0.9)	[0.9- 1)	1
Greedy	-	-	0.04	6.46	93.32	0.21
LS	-	-	0.004	3.21	95.96	1.1
RA	-	0.01	2.18	38.32	59.5	0.01
PD DEC	-	-	-	0.06	56.88	43.06
PD DEC LS	-	-	-	0.04	50.74	49.21
PD INC	-	-	-	0.47	63.57	35.96
PD INC LS	-	-	-	0.2	55.56	44.24

Table 5.12: k=8, 25000 instances, Dual Performance Ratios

	[0.5- 0.6)	[0.6- 0.7)	[0.7- 0.8)	[0.8- 0.9)	[0.9- 1)	1
RA	0.08	19.14	70.13	10.64	0.01	-
PD DEC	-	-	-	0.06	61.03	38.9
PD INC	-	-	-	0.5	83.77	15.73

Table 5.13: Maximum Number of Consecutive Iterations Without Combinatorial Progress

k	Maximum DEC	Average DEC	Maximum INC	Average INC
3	21	20.19	42	28.33
4	36	35.33	8677	7876.50
5	50	47.97	12050	10899.58
6	62	61.01	29975	14337.07
7	75	73.55	9220	6115.34
8	83	81.24	4933	3850.77

Table 5.14: Average CPU Seconds to Run Primal-Dual Local Search Heuristics

k	Average DEC	Average INC
3	1.34	0.61
4	3.02	2.92
5	5.55	4.52
6	8.37	6.31
7	15.06	7.63
8	18.94	10.91

5.2 Experiments for Instances of CRPR

In this section we provide the results when we run the modified greedy algorithm from Section 4.1 in Table 5.15. We will focus our attention on choices of $m \leq 2$, as we want to study our performance for a small constant m. The standard greedy algorithm performed well with $m = 0$, but as we expected, the performance ratio drops as k increases. We recall that for the same instance of (CRPR), aside from piece limits k_1 and k_2 for $k_1 < k_2$, the first k_1 pieces picked by our algorithm in either instance will be the same. For $m = 0$, and $k = 3, 4$, we were able to obtain performance ratios above 0.9. For $m = 0$ and $k = 5, 6, 7$ we were able to obtain performance ratios above 0.8. For $m = 0$ and $k = 8$, we were able to obtain a performance ratio of 0.7. For $m = 1$ and $k = 5$ we had a performance ratio of 0.895. For all other k, we achieved a ratio of at least 0.9. For $m = 2$, for all choices of k, we had a performance ration of at least 0.9.

We note that even for very small values of m, the greedy solution performs very well, and should likely be tested for future use in algorithms for the complete cranial vault remodeling procedure.

Table 5.15: Modified Greedy

k \ m	0	1	2	3	4	5	6	7	8
3	0.991	0.991	1.0	1.0	-	-	-	-	-
4	0.973	0.973	0.997	1.0	1.0	-	-	-	-
5	0.895	0.895	0.91	0.948	0.987	0.987	-	-	-
6	0.853	0.927	0.945	0.95	0.966	0.967	0.967	-	-
7	0.814	0.917	0.939	0.954	0.956	0.956	0.964	0.964	-
8	0.775	0.904	0.936	0.946	0.935	0.951	0.954	0.956	0.956

5.3 Unanswered Questions

We see that on our observed test data, none of our heuristics gave a solution that was worse than half the optimum. Whether or not our primal-dual heuristics from section 4.4 can be proven to have an approximation guarantee of $\frac{1}{2}$ was a question we posed in section 4.4.6. This question still stands after our study. In the same section, we left as an open question the worst-case run-time of our approaches. Given our results, one may hypothesize that the increasing primal-dual approach from section 4.4.4 does not have polynomial run-time, because we had instances taking almost 30000 iterations, while our input size was $O(poly(200))$. Although we should note that we cannot conclude anything concrete without further investigation, especially considering our statistics on CPU run-time. It is also not clear whether or not the increasing primal-dual approach from section 4.4.3 runs in polynomial time. It would be interesting to understand what causes the differences in behaviour. We ask if any alterations can be made to the increasing approach to improve its performance. In particular we ask if we can make an early stopping condition that does not affect observed performance drastically.

For those methods from section 4 that can be used as heuristics for the entire cranial vault remodeling procedure, it would be an exciting avenue of study to implement and test them experimentally.

Chapter 6

Conclusions and Future Research

This chapter summarizes our results and some possible future directions for research that we mentioned throughout the thesis.

In this thesis, we studied a simplification of the cranial vault remodeling procedure called fronto-orbital advancement with rearrangement. First, we modeled the decisions involved in such a procedure as a combinatorial optimization problem called the Curve Reshaping Problem with Rearrangement (CRPR). Also, we offered an additional problem Curve Reshaping Problem with Rearrangement and Pre-cut Segments (CRPRPS) where the decision of how to separate the infant's bandeau into segments has already been made. We then proved that these problems are strongly $\mathcal{NP}$-hard, and with no additional restrictions, they are also not approximable by any polynomial function with respect to input-size. We then posed the question of whether or not our choice of cost functions μ and c could be improved, or we could restrict our class of instances to weaken our inapproximability result. To work toward algorithms that run in polynomial time and perform well on instances of (CRPR) and (CRPRPS) we took an affine transformation of our original objective function, converting our minimization problem to a maximization problem. We then gave multiple approaches to tackle instances of (CRPRPS) and one approach to tackle instances of (CRPR). When we tested our techniques on a data set of interest, it performed well, with the possibility of poor run-time for one of our unproven approaches. We noted that our heuristics do not observe any unique characteristics of the specific choice of cost and fit functions c and μ given in section 2.2.2 applied to instances of interest for our application. We wonder if these properties can explain our heuristics' performance or guide us to discover more specially designed algorithms. Other interesting questions include testing extensions of these approaches to a formulation for the entire cranial vault remodeling procedure and further exploring performance and run-time of the primal-dual strategies from section 4.4.

Bibliography

[1] "Craniofacial Center Dallas, Texas," http://thecraniofacialcenter.com/synostoses_treatment.html, (Accessed on 08/27/2020).

[2] N. R. Saber, J. Phillips, T. Looi, Z. Usmani, J. B, J. Drake, and P. C. Kim, "Generation of normative pediatric skull models for use in cranial vault remodeling procedures," *Child's Nervous System*, vol. 28, no. 3, pp. 405–410, 2012.

[3] L. G. Serletti, Joseph, "Consultation with the specialist: Pediatric approach to craniosynostosis," *Pediatr Rev*, vol. 19, no. 10, 1998.

[4] "Cranial sutures medicine plus," https://medlineplus.gov/ency/article/002320.htm, (Accessed on 11/06/2020).

[5] "Craniosynostosis Mayo Clinic," https://www.mayoclinic.org/diseases-conditions/craniosynostosis/symptoms-causes/syc-20354513, (Accessed on 11/06/2020).

[6] 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*, vol. 121, no. 4, pp. 170e–178e, 2008.

[7] "Surgical options for craniosynostosis," https://childrenswi.org/medical-care/neuroscience/conditions/craniosynostosis#Video7, (Accessed on 11/06/2020).

[8] "Craniosynostosis Surgery Children's Hospital of Philadelphia," https://www.chop.edu/treatments/surgical-treatment-craniosynostosis, (Accessed on 08/12/2020).

[9] J. Drake, M. Drygala, R. Fukasawa, J. Koenemann, A. Linhares, T. Looi, J. Phillips, D. Qian, N. Saber, J. Toth *et al.*, "Optimized cranial bandeau remodeling," *arXiv preprint arXiv:1912.10601*, 2019.

[10] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber, "A unified approach to approximating resource allocation and scheduling," *Journal of the ACM (JACM)*, vol. 48, no. 5, pp. 1069–1090, 2001.

[11] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[12] J. Kleinberg and E. Tardos, *Algorithm design.* Pearson Education India, 2006.

[13] B. Korte and Vygen, *Combinatorial optimization.* Springer, 2012, vol. 2.

[14] T. A. Jenkyns, "The efficacy of the "greedy" algorithm," in *Proceedings of the 7th Southeastern Conference on Combinatorics, Graph Theory and Computing*, 1976, pp. 341–350.

[15] B. Korte and D. Hausmann, "An analysis of the greedy heuristic for independence systems," *Algorithmic aspects of combinatorics*, vol. 2, pp. 65–74, 1978.

[16] W. Michiels, E. H. Aarts, and J. Korst, "Theory of local search," in *Handbook of heuristics.* Springer, 2018, pp. 299–339.

[17] M. X. Goemans and D. P. Williamson, "The primal-dual method for approximation algorithms and its application to network design problems," *Approximation algorithms for NP-hard problems*, pp. 144–191, 1997.

[18] T. F. Gonzalez, *Handbook of approximation algorithms and metaheuristics.* CRC Press, 2007.

[19] L. Trevisan, "Inapproximability of combinatorial optimization problems," *arXiv preprint cs/0409043*, 2004.